Fitness Landscape Analysis and Image Filter Evolution Using Functional-Level CGP

Karel Slaný and Lukáš Sekanina

Faculty of Information Technology, Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
slany@fit.vutbr.cz, sekanina@fit.vutbr.cz

Abstract. This work analyzes fitness landscapes for the image filter design problem approached using functional-level Cartesian Genetic Programming. Smoothness and ruggedness of fitness landscapes are investigated for five genetic operators. It is shown that the mutation operator and the single-point crossover operator generate the smoothest landscapes and thus they are useful for practical applications in this area. In contrast to the gate-level evolution, a destructive behavior of a simple crossover operator has not been confirmed.

1 Introduction

Cartesian Genetic Programming (CGP) was introduced by J. Miller and P. Thomson in 1999 [6]. In contrast with a standard genetic programming, CGP represents candidate programs as bounded $u \times v$-node directed graphs (rather than as trees), utilizes only a mutation operator and operates with a small population – to mention main differences.

In the connection with CGP, several issues have been discussed in the recent years, for example, the role of neutrality (which is implicit to the CGP representation) [2, 19], the role of bloat [9], modularity in CGP [17] and the usefulness of the search strategy which is based on a simple mutation [14, 8, 10]. While the standard GP benefits from crossover operators, it seems that a crossover is not useful for CGP at all (at least for the problems approached by CGP up to now). In other words, nobody has proposed a useful operator for CGP so far.

Techniques of fitness landscape analysis were utilized to obtain an information about CGP fitness landscapes in the digital circuit design task [14, 8, 16]. The structure of fitness landscapes has been studied in terms of their smoothness, ruggedness and neutrality on a time series obtained by sampling fitness values on a random walk. Easy problems are supposed to have smooth landscapes, while hard problems are supposed to be caused by rugged landscapes. It was recognized that the crossover is not a useful operator for evolving logic circuits using CGP, since the corresponding landscapes are extremely rugged. On the other hand, the mutation landscapes appear to be relatively smooth, and therefore, the mutation is more feasible for evolutionary circuit design [14].

In order to evolve more complicated digital circuits, CGP has been extended to operate at the functional level [12, 13]. Instead of simple gates, CGP works

M. Ebner et al. (Eds.): EuroGP 2007, LNCS 4445, pp. 311–320, 2007.

with high-level components, such as adders, comparators, k-bit logic functions etc. This approach is especially well suited for the evolution of image operators, such as smoothing filters and edge detectors. It was reported in several case studies that the resulting operators are human-competitive [13, 5].

Similarly to the gate-level CGP, only a mutation operator was utilized for the functional-level CGP in mentioned applications. The problem investigated in this paper is whether a crossover operator is suitable for image operator evolution at the functional level using CGP. The motivation for investigating this problem comes from the following ideas: It seems that the image filter design problem is easier for CGP than, let us say, 3-bit multiplier design problem. Experimental results show that while approx. 30k generations are needed to find a good image filter [5], 5M generations are needed to find a good multiplier [15].[1]

The reason is that in case of gate-level circuit evolution, all possible input combinations are generated for a candidate circuit in order to obtain its fitness value. If only a subset of input combinations were considered, a vast majority of resulting circuits would not be fully functional and thus useful for real-world applications. On the other hand, the fitness function used in the image operator design problem utilizes only a subset of all possible combinations of image pixels. As human eyes are not able to see all the details in images, sufficiently good (not necessarily perfect) image operators are acceptable. It also seems that the mutation operator is more destructive for the multiplier evolution than for the filter evolution. This higher flexibility in resulting acceptable circuits and less destructive mutations indicate that corresponding fitness landscapes could be much smoother for the image filter evolution than for the gate-level evolution of multipliers.

The goal of this work is to perform the fitness landscape analysis for image filter design problem which is approached by the functional-level CGP. Various mutation and crossover operators will be compared in terms of fitness landscape analysis with the aim of identifying a suitable genetic operator for this particular problem. Regarding the previous analysis, our assumption is that a crossover operator can be identified which could help to make the resulting search algorithm more efficient than the original mutation-based approach.

2 Fitness Landscapes

The metaphor of fitness landscape, introduced in biology, expresses the idea that evolution can be viewed as a population flow on a surface in which the altitude of a point indicates how well the corresponding organism is adapted to an environment [18]. In the field of evolutionary computing, fitness landscapes are investigated in order to learn how difficult a particular problem is for a given search algorithm [4, 1, 8, 11]. On the basis of results of the fitness landscape

[1] Note that although the image filter evolution requires fewer generations, it takes much more time than the multiplier evolution, because the fitness value calculation is much more time consuming (e.g. 256 times more consuming for a training image of 128×128 pixels vs. a 3-bit multiplier with 64 test cases).

analysis, an original search algorithm is usually modified in order to improve its performance. Note that for GP, some authors have shown that the landscape metaphor may be deceptive [3].

A population moving on the fitness landscape creates a path which is called a walk on a fitness landscape. A walk $\{f_t\}_{t=0}^{n}$ can be described as a time series of $n+1$ fitness values which we acquired by using a reproduction operator.

Similarly to [14, 8] and in order to represent the changes made to the fitness value, we can introduce a string of marks which represent corresponding changes to the fitness value. Let $F = f_0 f_1 \dots f_n$ be a series of $n+1$ fitness values we have sampled during n generations from the initial population 0. This series can be represented by a string of symbols $S = s_1 s_2 \dots s_n$ over the alphabet $s_i \in \Sigma = \{\bar{1}, 0, 1\}$, which can be formalized by the function

$$\Psi_{f_t}(i,\varepsilon) = \begin{cases} \bar{1}, & f_i - f_{i-1} < -\varepsilon \\ 0, & |f_i - f_{i-1}| \leq \varepsilon \\ 1, & f_i - f_{i-1} > \varepsilon \end{cases} \tag{1}$$

so that

$$s_i = \Psi_{f_t}(i,\varepsilon) \tag{2}$$

for any constant ε [8]. For the fitness landscape $\mathcal{L}$ [8], the parameter ε is a real number from the interval $\langle 0, l_f \rangle$, where l_f is the greatest value on the fitness landscape. The parameter ε is used as an magnifying glass. The function $\Psi_{f_t}(i,0)$ is very sensitive to the changes made to the fitness value during the walk and thus the string $S(0)$ is set with the maximum accuracy. Contrariwise, the string $S(l_f)$ consists only of zeros.

By using the $S(\varepsilon)$ string, we can introduce two information characteristics called *entropic measures*. While the first entropic measure (FEM) is defined as

$$H(\varepsilon) = -\sum_{p \neq q} P_{[pq]} \log_6 P_{[pq]}, \tag{3}$$

the second entropic measure (SEM) is defined as

$$h(\varepsilon) = -\sum_{p=q} P_{[pq]} \log_3 P_{[pq]}. \tag{4}$$

The first measure rates the ruggedness of a landscape; the second measure estimates the smoothness of a landscape walk. The parameter $P_{[pq]}$ denotes the probability of the occurrence of the string pq in the string $S(\varepsilon)$. Smaller values of the measures mean neater walks.

Our goal is to evaluate different genetic operators by means of FEM and SEM. For the creation of the $S(\varepsilon)$ string we use a series of fitness values, which correspond to the fitness values of the best population member in each generation. As the parameter ε is set to 0, we can obtain the best resolution for the fitness landscape analysis.

3 CGP at the Functional Level for Image Filter Evolution

3.1 Cartesian Genetic Programming

In CGP, a candidate graph (circuit) is modeled as an array of u (columns) $\times$ v (rows) of programmable elements (gates). The number of circuit inputs, n_i, and outputs, n_o, is fixed. Feedback is not allowed. Each gate input can be connected to the output of a gate placed in the previous L columns or to some of circuit inputs. The L parameter, in fact, defines the level of connectivity and thus reduces/extends the search space. For example, if L=1 only neighboring columns may be connected; if $L = u$, the full connectivity is enabled. Each gate is programmed to perform one of functions defined in the set Γ. Figure 1 shows an example and a corresponding chromosome. Every individual is encoded using $u \times v \times 3 + n_o$ integers.

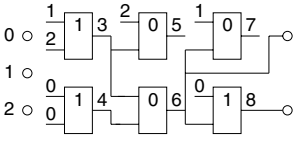

Fig. 1. An example of a 3-input circuit. CGP parameters are as follows: $L = 3$, $u = 3$, $v = 2$, Γ = {AND (0), OR (1)}. Gates 5 and 7 are not utilized. Chromosome: 1,2,1, 0,0,1, 2,3,0, 3,4,0 1,6,0, 0,6,1, 6, 8. The last two integers indicate the outputs of the circuit.

CGP operates with the population of λ individuals (typically, $\lambda = 5$). The initial population is randomly generated. Every new population consists of a parent (the fittest individual from the previous population) and its mutants. In case that two or more individuals have received the same fitness score in the previous population, the individual which did not serve as a parent in the previous population will be selected as a new parent. This strategy is used to ensure the diversity of population. The mutation operator modifies some randomly selected genes of an individual.

3.2 Image Filter Evolution

As introduced in [12, 13], every image operator will be considered as a digital circuit of nine 8-bit inputs and a single 8-bit output, which processes gray-scaled (8 bits/pixel) images. Fig. 2 shows that every pixel value of the filtered image is calculated using a corresponding pixel and its eight neighbors in the processed image. Each of circuit nodes can be programmed to perform one of functions given in Table 1. Recall that all functions operate with 8-bit operands and generate 8-bit outputs.

The goal of CGP is to propose a filter which minimizes the difference between filtered image I_2 and a reference image I_r which must exist for a particular

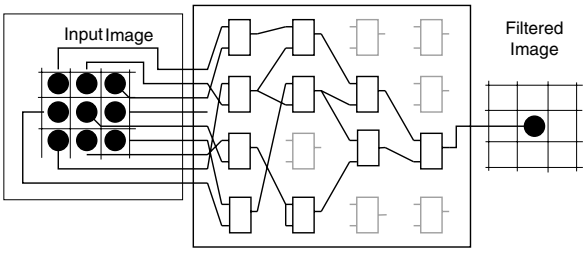

Fig. 2. Example of the image operator ($u = v = 4$). The output pixel value is calculated using the corresponding pixel and its eight neighbors in the input image.

Table 1. Functions used in circuit components. All functions have 8-bit operands and 8-bit outputs.

ID	Function	Description	ID	Function	Description
0	$x \vee y$	binary or	4	$x +_{sat} y$	addition with saturation
1	$x \wedge y$	binary and	5	$(x + y) >> 1$	average
2	$x \oplus y$	binary xor	6	$Max(x, y)$	maximum
3	$x + y$	addition	7	$Min(x, y)$	minimum

corrupted input image I_1. Suppose that the corrupted image and the reference image are of the size $K \times L$ pixels. Then the filtered image has the size of $(K - 2) \times (L - 2)$ pixels. The quality of the evolved image filter is evaluated by the fitness function

$$fitness = 255 \cdot (K - 2) \cdot (L - 2) - \sum_{i=1}^{K-2} \sum_{j=1}^{L-2} |I_2(i, j) - I_r(i, j)|. \quad (5)$$

Papers [13, 5] show that this approach leads to very good image filters even in case that only a single image is utilized in the fitness function. As suitable image sizes, $K = 128$ and $L = 128$ are considered.

4 Proposed Genetic Operators

This section briefly introduces one mutation operator and four crossover operators we tested for image filter evolution. Please notice that operators that differ only in the number of offspring inserted into the new generation have quite significant differences in entropic measures.

Mutation Operator: CGP, as it is defined, uses only a mutation as the genetic operator. This operator selects the best-scored individual from the previous population. Its copies are mutated and inserted into the new population. Elitism is enabled. Diversity of the population is maintained according to the strategy described in Section 3.1.

```
do
  copy the best member into the new population
  mutate the just copied member in the new population
until the new population is not full
```

Single-Point Crossover (1): This is a standard one-point crossover operator which operates at the level of integer chromosomes. However, only one offspring of the crossover operation is mutated and included into the new population.

```
do
  choose two different best members of the previous population
  make one point crossover over its copies
  copy one offspring into the new population and mutate it
until the new population is not full
```

Single-Point Crossover (2): In comparison with the previous operator, this one copies the both offspring into the new population and mutates them.

Multi-point Crossover (1): This operator is an example of a multi-place crossover operator. This operator selects two (different) best members from the previous population. These two chromosomes are mutually combined in the way that they switch their functional blocks wiring; however, nodes' functions remain unchanged. Only one offspring is moved into the new generation. Then it is mutated.

```
do
  choose two different best members of the previous population
  make a multi-point crossover over its copies
  copy one offspring into the new population and mutate it
until new the population is not full
```

Multi-point Crossover (2): In comparison with the previous operator, this one copies the both offspring into the new population and mutates them.

5 Experimental Results

In order to estimate the smoothness and ruggedness of fitness landscapes, we measure the entropic measures h and H for each operator and parameters setting. Note that elitism is not utilized in our experiments.

The experiments are divided into two groups. While the first group operates with 7×4-node graphs (circuits) and 8-member population, the second group operates with 8×5-node graphs and population sizes 1, 2, 4 and 8 individuals. The mutation probability is set to 3% and L-parameter is set to 1 in both cases. A 128×128-pixel Lena image containing a random shot noise is utilized as a training image.

As these experiments are very time consuming, we use only 1000 generations in the first series of experiments (denoted as the *short run* in following figures).

In order to obtain more precise results, 100000 generations are performed in the second series of experiments (denoted as the *long run* in following figures). The first series is repeated 600 times; the second series is repeated 10 times.

Figure 3 shows average values of $H(0)$ (FEM) and $h(0)$ (SEM) for five genetic operators and for CGP with 7×4 nodes, i.e. for the first group of experiments. The lower average values the neater walk on the fitness landscape. Figures 4 and 5 show average values of $H(0)$ and $h(0)$ for the same genetic operators and CGP with 8×5 nodes. These figures are parameterized by the size of population.

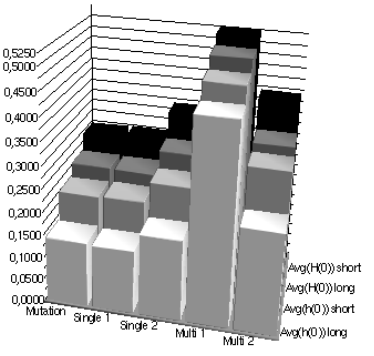

Fig. 3. Average values of $H(0)$ and $h(0)$ measures for five genetic operators and for CGP with 7×4 nodes

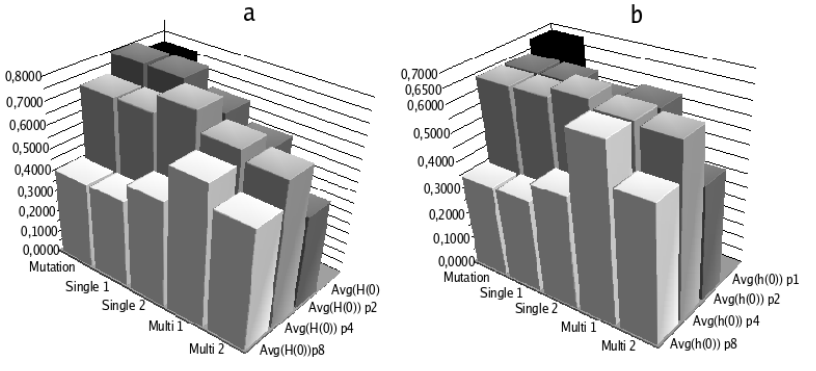

Fig. 4. Average values of $H(0)$ and $h(0)$ measures for five genetic operators and for CGP with 8×5 nodes. Results are given for *short runs* and population sizes 1, 2, 4 and 8 individuals (denoted as p1 – p8).

6 Discussion

For the first group of experiments, where the population contains 8 individuals, the mutation operator generates the smoothest fitness landscapes. Fig. 3

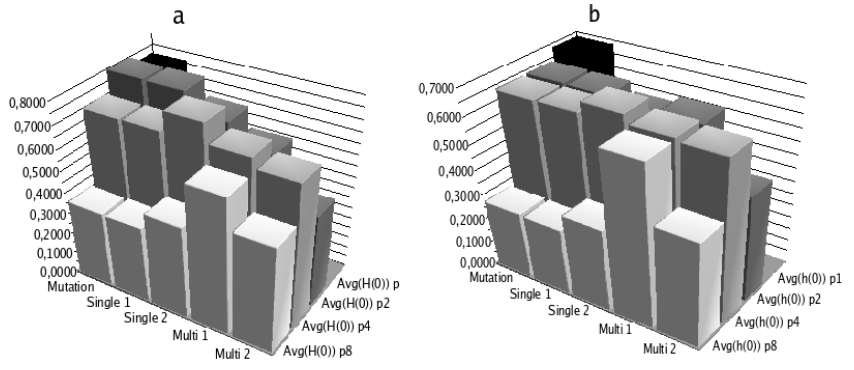

Fig. 5. Average values of $H(0)$ and $h(0)$ measures for five genetic operators and for CGP with 8×5 nodes. Results are given for *long runs* and population sizes 1, 2, 4 and 8 individuals (denoted as p1 – p8).

Fig. 6. Left – Lena image corrupted by the shot noise. Middle – resulting image for the multi-point crossover operator (1). Right – resulting image for the mutation operators.

also shows that Single-point Crossover (1) produces practically identical results. Other crossover operators generate more rugged fitness landscapes.

For the second group of experiments, we can observe that the shape of bars remains practically unchanged (for the population size of 8 individuals) when compared with the shape obtained for the first group. However, the average values of $H(0)$ and $h(0)$ are slightly higher, which is caused by using more nodes in CGP. By increasing the number of generations, the average values of the entropic measures slightly decrease; however, only when the population consists of 8 individuals. The number of generations does not influence the average values of $H(0)$ and $h(0)$ for remaining genetic operators significantly.

We can observe that the population size is the parameter which significantly contributes to the average values of the entropic measures. The mutation operator produces the smoothest landscapes for the population of 8 individuals. For smaller populations, crossover operators are becoming more valuable, probably because they are able to introduce more diversity to the search process. Recall that standard version of CGP usually operates with the population of 5 individuals. Our results confirm that the use of four or less individuals is not advantageous. We can also observe different behaviors of crossover operators we

have investigated. For the search process it is beneficial when the single point crossover provides only one mutated offspring to the new population; on the other hand, the both offspring should be used in case of the multi-point crossover.

Examples of resulting Lena images (see Fig. 6) were obtained by the mutation operator and the multipoint crossover (1) operator, with an 8-member population and after 100000 generations. Note that the best filters reported in [13] are able to remove this type of noise perfectly.

Approx. 22 hours of computation of a 1.3 GHz CPU are needed to finish a single run of CGP (100000 populations were produced on the topology of 8×5-node graphs, for 128×128-pixel images and with 8-member population).

7 Conclusions

A fitness landscape analysis was performed for the image filter design problem approached using functional-level CGP. Smoothness and ruggedness of fitness landscapes were measured for five genetic operators. It was shown that the mutation operator and the single-point crossover operator generate the smoothest landscapes and thus they are useful for practical applications in this area. In contrast to the gate-level evolution, a destructive behavior of a simple crossover operator was not confirmed. As the mutation operator is easy to implement, it remains the most useful operator for image filter evolution using CGP.

Acknowledgements

This work was supported by the Grant Agency of the Czech Republic under No. 102/07/0850 *Design and hardware implementation of a patent-invention machine* and the Research intention No. MSM 0021630528 – Security-Oriented Research in Information Technology.

References

[1] Alander J. T.: Population size, building block, fitness landscape and genetic algorithm search efficiency in combinational optimisation: An empirical study. Practical Handbook of Genetic Algorithms, Vol. 3, CRC Press, 1999, p. 459–485

[2] Collins, M.: Finding needles in haystacks is harder with neutrality. Genetic Programming and Evolvable Machines. Vol. 7, No. 2, 2006, p. 131–144

[3] Daida J. M. et al.. What makes a problem GP-hard? Genetic Programming and Evolvable Machines. Vol. 2, No. 2, 2001, p. 165–191

[4] Jones, T.: Evolutionary Algorithms, Fitness Landscapes and Search. PhD thesis, University of New Mexico, 1995, p. 249

[5] Martinek, T., Sekanina, L.: An evolvable image filter: Experimental evaluation of a complete hardware implementation in FPGA. In: Proc. of the 6th International Conference Evolvable Systems: From Biology to Hardware, ICES 2005, Sitges, Barcelona, LNCS 3637, Springer-Verlag, 2005, p. 76–85

[6] Miller, J., Thomson, P.: Cartesian genetic programming. In: Proc. of the 3rd European Conference on Genetic Programming, LNCS 1802, Springer Verlag, Berlin, 2000, p. 121–132
[7] Miller, J., Job, D., Vassilev, V.: Principles in the evolutionary design of digital circuits – Part I. Genetic Programming and Evolvable Machines. Vol. 1., No. 1, 2000, p. 8–35
[8] Miller, J., Job, D., Vassilev, V.: Principles in the evolutionary design of digital circuits – Part II. Genetic Programming and Evolvable Machines. Vol. 1, No. 2, 2000, p. 259–288
[9] Miller, J.: What bloat? Cartesian Genetic Programming on Boolean Problems. In: GECCO'01 - Late Breaking Papers. Proc. of the 3rd Genetic and Evolutionary Computation Conference, San Francisco, CA, Morgan Kaufmann Publishers, 2001, p. 295–302
[10] Miller, J., Smith, S.: Redundancy and Computational Efficiency in Cartesian Genetic Programming. IEEE Transactions on Evolutionary Computation. Vol. 10, No. 2, 2006, p. 167–174
[11] Reeves, C.: Fitness Landscapes. In: Burke, E.K. and Kendall, G., Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, Springer 2005, pp. 587-610.
[12] Sekanina, L.: Image Filter Design with Evolvable Hardware. In: Applications of Evolutionary Computing – Proc. of the 4th Workshop on Evolutionary Computation in Image Analysis and Signal Processing EvoIASP'02, LNCS 2279, Springer-Verlag, Berlin, 2002, p. 255–266
[13] Sekanina, L.: Evolvable components: From Theory to Hardware Implementations, Springer-Verlag, Natural Computing Series, 2004
[14] Vassilev, V., Miller, J., Fogarty, T.: On the Nature of Two-Bit Multiplier Landscapes. In: Proc. of the 1st NASA/DoD Workshop on Evolvable Hardware, Pasadena, CA, IEEE Computer Society, Los Alamitos, 1999, p. 36–45
[15] Vassilev, V., Job, D., Miller, J.: Towards the Automatic Design of More Efficient Digital Circuits. In: Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware, CA, USA IEEE Computer Society, Los Alamitos, 2000, p. 151–160
[16] Vassilev, V., Miller, J.: Scalability Problems of Digital Circuit Evolution. In: Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware, CA, USA, IEEE Computer Society, Los Alamitos, 2000, p. 55–64
[17] Walker, J., Miller, J.: Investigating the performance of module acquisition in cartesian genetic programming. Proc. of GECCO 2005, ACM, 2005, p. 1649–1656
[18] Wright, S.: The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In: Proceedings of the 6th Int. Congress on Genetics, 1932, p. 355–366
[19] Yu, T., Miller, J.: Neutrality and the Evolvability of Boolean Function Landscape. Proc. of the 4th European Conference on Genetic Programming, LNCS 2038, Springer Verlag, Berlin, 2001, p. 204–217