1

Graph-Based Evolutionary Art

Penousal Machado, João Correia, and Filipe Assunção

CISUC, Department of Informatics Engineering, University of Coimbra,
3030 Coimbra, Portugal
{machado,jncor}@dei.uc.pt, fga@student.dei.uc.pt

Summary. A graph-based approach for the evolution of Context Free Design Grammars is presented. Each genotype is a directed hierarchical graph and, as such, the evolutionary engine employs graph-based crossover and mutation. We introduce six different fitness functions based on evolutionary art literature and conduct a wide set of experiments. We begin by assessing the adequacy of the system and establishing the experimental parameters. Afterwards, we conduct evolutionary runs using each fitness function individually. Finally, experiments where a combination of these functions is used to assign fitness are performed. Overall, the experimental results show the ability of the system to optimize the considered functions, individually and combined, and to evolve images that have the desired visual characteristics.

Key words: Evolutionary Art, Graph-Based Genetic Programming, Fitness Assignment

1.1 Introduction

The development of an evolutionary art system implies two main considerations: (i) the design of a generative system that creates individuals; (ii) the evaluation of the fitness of such individuals [17]. In the scope of this Chapter we address both of these considerations.

Influenced by the seminal work of Karl Sims [25], the vast majority of evolutionary art systems follows an expression-based approach: the genotypes are trees encoding symbolic expressions and the phenotypes – i.e., images – are produced by executing the genotypes over a set of x, y values. While this approach has been proven fruitful, it has several shortcomings, most notably: (i) Although it is theoretically possible to evolve any image [12], in practice, expression-based evolutionary art tends to produce abstract, mathematical images; (ii) Due to the representation, the images lack graphic elements that are typically present in most forms of art, such as lines, strokes, clearly defined shapes and objects; (iii) Creating an appealing image by designing a symbolic

expression by hand, or even understanding an evolved expression, is a hard endeavour.

Extending previous work [14, 13], we describe an approach that overcomes these limitations and introduces new possibilities. Inspired on the work of Stiny and Gips [27], who introduced the concept of *shape grammars*, we explore the evolution of Context Free Design Grammars (CFDGs) [9], which allow the definition of complex families of shapes through a compact set of production rules. As such, in our approach, each genotype is a well-constructed CFDG. Internally, and for the purposes of recombination and mutation, each genotype is represented as a hierarchical directed graph. Therefore, the evolutionary engine deviates from traditional tree-based Genetic Programing (GP) and adopts graph-based crossover and mutation operators. The details of the representation are presented in Section 1.3, while Section 1.4 describes the genetic operators.

In Section 1.5 we introduce several fitness assignment schemes based on evolutionary art literature. Then, in the same Section, we describe how we combine several of these measures in a single fitness function.

We conduct several tests to assess the adequacy of the system and determine reasonable experimental settings. In particular, we focus on the impact of unexpressed code in the evolutionary process, presenting and analyzing different options for handling these portions of code. Furthermore, we study how non-deterministic mapping between genotypes and phenotypes influences the robustness of the evolved individuals. These experiments are reported in Section 1.6. Based on the results of these tests, we conduct experiments using each of the previously defined fitness functions individually. The description and analysis of the experimental results is presented in Section 1.7. The analysis of the results highlights the type of images favored by each fitness function and the relations among them. We then proceed by presenting results obtained when using a combination of functions to guide fitness (Section 1.7.2). The analysis of these results is focused on the ability of the system to create imagery that simultaneously addresses the different components of the fitness functions. We finalize by drawing overall conclusions and identifying future work.

1.2 State of the Art

Although there are noteworthy expression-based evolutionary art systems (e.g. [25, 29, 28, 12, 8]), systems that allow the evolution of images that are composed of a set of distinct graphic elements such as lines, shapes, colors and textures are extremely scarce.

Among the exceptions to the norm, we can cite the work of: Baker [1], who uses a Genetic Algorithm (GA) operating on strings of variable size to evolve line drawings; Heijer and Eiben [6] who evolve Scalable Vector Graphics (SVG), manipulating directly SVG files through a set of specifically designed

```
startshape Edera

rule Edera {
     CIRCLE    {s 5}
     Ciglio    {}
     Edera     {x -5 y -1 s 0.90} }

rule Ciglio {
     SQUARE    {hue 200 sat 0.5}
     Pelo      {r 5 hue 200 sat 0.5}
     Ciglio    {y -1 r 0.5 s 0.998 b 0.005} }

rule Ciglio {
     SQUARE    {hue 200 sat 0.5}
     Pelo      {r 5 hue 200 sat 0.5}
     Ciglio    {y -1 r 0.5 s 0.998 b 0.005 flip 90} }

rule Ciglio .008 {
     SQUARE    {hue 200 sat 0.5}
     Pelo      {r 5 hue 200 sat 0.5}
     Ricciolo  {y -1 s 0.998 b 0.005} }

rule Ricciolo {
     SQUARE    {hue 200 sat 0.5}
     Pelo      {r 5 hue 200 sat 0.5}
     Ricciolo  {y -1 r 3 s 0.998 b 0.005} }

rule Ricciolo .005 {
     SQUARE    {hue 200 sat 0.5}
     Pelo      {r 5 hue 200 sat 0.5}
     Ricciolo  {y -1 r 3 s 0.998 b 0.005 flip 90} }

rule Pelo {
     CIRCLE    {s 5 0.1} }
```

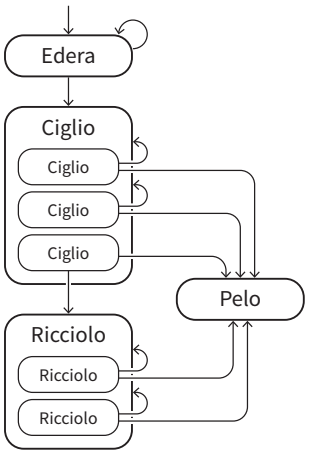

Fig. 1.1: On the left, a CFDG adapted from `www.contextfreeart.org/gallery/view.php?id=165`; On the right, the same CFDG represented as a graph (the labels of the edges were omitted for the sake of clarity).

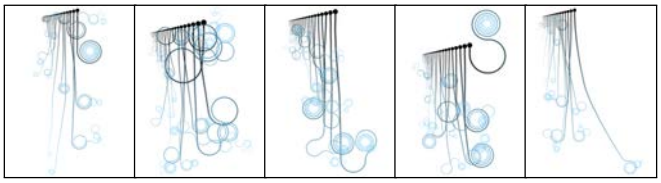

Fig. 1.2: Examples of images produced by the CFDG depicted in Figure 1.1.

mutation and recombination operators. Unlike GP approaches, where the representation is procedural, the representations adopted in these works are, essentially, descriptive – in the sense that the genotypes describe the elements of the images in a relatively directed way instead of describing a procedure, i.e. program, that once executed or interpreted produces the image as output.

In addition to our early work on this topic [14, 13], there are two examples of the use of CFDG for evolutionary art purposes. Saunders and Grace [24] use a GA to evolve parameters of specific CFDG hand-built grammars. As the name indicates, *CFDG Mutate* [4] allows the application of mutation operators to CFDGs. Unfortunately the system only handles deterministic grammars (see Section 1.3) and does not provide recombination operators.

O'Neil et al. [21] explore the evolution of shape grammars [27] using Grammatical Evolution [20] for design purposes, generating 2D shapes [21] and 3D structures [19]. Although they do not use CFDGs, their work is, arguably, the one that is most similar in spirit to the described in this Chapter, due to the adoption of a procedural representation based on grammars and a GP approach.

1.3 Representation

Context Free [9] is a popular open-source application that renders images which are specified using a simple language entitled CFDG (for a full description of CFDG see [5]). Although the notation is different from the one used in formal language theory, in essence, a CFDG program is an augmented context free grammar, i.e., a 4-tuple: (V, Σ, R, S) where:

1. V is a set of non-terminal symbols;
2. Σ is a set of terminal symbols;
3. R is a set of production rules that map from V to $(V \cup \Sigma)^*$;
4. S is the initial symbol.

Figure 1.1 depicts the CFDG used to illustrate our description. Programs are interpreted by starting with the S symbol (in this case $S = Edera$) and proceeding by the expansion of the production rules in breath-first fashion. Predefined Σ symbols call drawing primitives (e.g., SQUARE). CFDG is an *augmented* context free grammar: it takes parameters that produce semantic operations (e.g., s produces a scale change). Program interpretation is terminated when there are no V symbols left to expand, when a predetermined number of steps is reached, or when the rendering engine detects that further expansion does not induce changes to the image [14].

Like most CFDGs, the grammar depicted in Figure 1.1 is non-deterministic: several production rules can be applied to expand the symbols *Ciglio* and *Ricciolo*. When several production rules are applicable one of them is selected randomly and the expansion proceeds. Furthermore, the probability of selecting a given production may be specified by indicating a weight (e.g., 0.08). If no weight is specified a default value of 1 is assumed. The non-deterministic nature of CFDGs has profound implications: each CFDG implicitly defines a language of images produced using the same set of rules (see Figure 1.2). Frequently, these images share structural and aesthetic properties. One can specify the seed used by the random number generator of the grammar interpreter, which enables the replicability of the results.

In the context of our evolutionary approach each genotype is a well-constructed CFDG grammar. Phenotypes are rendered using Context Free. To deal with non-terminating programs a maximum number of expansion steps is set. The genotypes are represented by directed graphs created as follows:

1. Create a node for each non-terminal symbol. The node may represent a single production rule (e.g., symbol *Edera* of Figure 1.1) or encapsulate the set of all production rules associated with the non-terminal symbol (e.g., symbols *Ciglio* and *Ricciolo* of Figure 1.1);
2. Create edges between each node and the nodes corresponding to the non-terminals appearing in its production rules (see Figure 1.1);
3. Annotate each edge with the corresponding parameters (e.g., in Figure 1.1 the edges to *Pelo* possess the label '{r 5 hue 200 sat 0.5}').

Algorithm 1 Random initialization of an individual.

procedure RANDOMINITIALIZATION
  $terminal \leftarrow$ set of terminal symbols
  $min_v, max_v \leftarrow$ minimum, maximum number of non-terminal symbols
  $min_p, max_p \leftarrow$ minimum, maximum number of production rules per non-terminal
  $min_c, max_c \leftarrow$ minimum, maximum number of calls per production
  $nonterminal \leftarrow RandomlyCreateNonTerminalSet(min_v, max_v)$
  for all $V \in nonterminal$ **do**
    $numberofproductions \leftarrow random(min_p, max_p)$
    for $i \leftarrow 1, numberofproductions$ **do**
      $productionrule \leftarrow NewProductionRule(V)$
      $numberofcalls \leftarrow random(min_c, max_c)$
      for $j \leftarrow 1, numberofcalls$ **do**
        if $random(0,1) < prob_t$ **then**
          $productionrule.InsertCallTo(RandomlySelect(terminal))$
        else
          $productionrule.InsertCallTo(RandomlySelect(nonterminal))$
        end if
        $productionrule.RandomlyInsertProductionRuleParameters()$
      end for
    end for
  end for
  $individual.setProductionRules(productionrules)$
  $individual.RandomlySelectStartShape(nonterminal)$
  $individual.RandomlyCreateBackgroundColor()$
end procedure

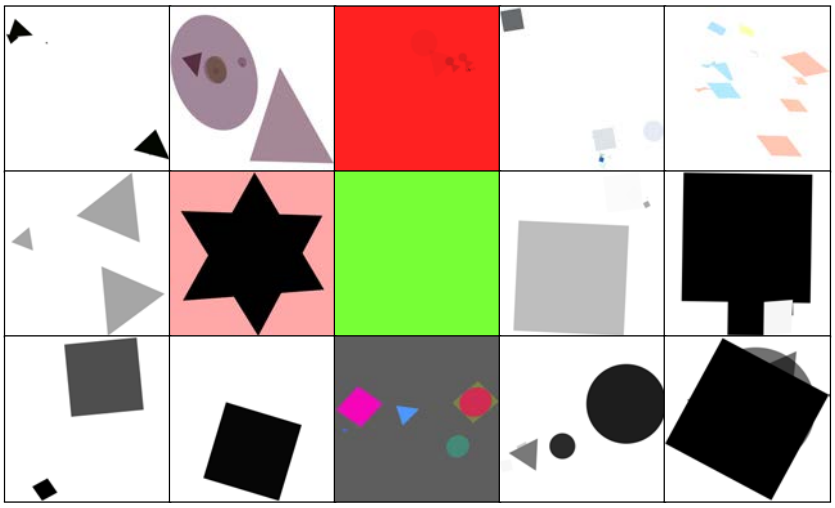

Fig. 1.3: Examples of phenotypes from a randomly created initial population.

1.4 Genetic Operators

In this Section we describe the genetic operators designed to manipulate the graph-based representation of CFDGs, namely: initialization, mutation and crossover.

1.4.1 Random Initialization

The creation of the initial population for the current evolutionary engine is of huge importance, being responsible for generating the first genetic material that will be evolved through time. In our previous works on the evolution of CFDGs the initial population was supplied to the evolutionary engine: the first population was either composed of human-created grammars [13] or of a single minimal grammar [14]. Although both those options have merit, the lack of an initialization procedure for the creation of a random population of CFDGs was a limitation of the approach.

In simple terms, the procedure for creating a random CFDG can be described as follows: we begin by randomly determining the number of non-terminal symbols and the number of production rules for each of the symbols (i.e. the number of different options for its expansion). Since this defines the nodes of the graph, the next step is the random creation of connections among nodes and calls to non-terminal symbols. The parameters associated with the calls to terminal and non-terminal symbols are also established randomly. Finally, once all productions have been created, we randomly select a starting node and background color. Algorithm 1 details this process, which is repeated until the desired number of individuals is reached. Figure 1.3 depicts a sample of a random initial population created using this method.

1.4.2 Crossover Operator

The crossover operator used for the experiments described in this Chapter is similar to the one used in our previous work on the same topic [14, 13]. The rational was to develop a crossover operator that would promote the meaningful exchange of genetic material between individuals. Given the nature of the representation, this implied the development of a graph-based crossover operator that is aware of the structure of the graphs being manipulated. The proposed operator can be seen as an extension of the one presented by Pereira et al. [22]. In simple terms, this operator allows the exchange of subgraphs between individuals.

The crossover of the genetic code of two individuals, a and b, implies: (i) Selecting one subgraph from each parent; (ii) Swapping the nodes and internal edges of the subgraphs, i.e., edges that connect two subgraph nodes; (iii) Establishing a correspondence between nodes; (iv) Restoring the outgoing and incoming edges, i.e., respectively, edges from nodes of the subgraph to non-subgraph nodes and edges from non-subgraph nodes to nodes of the subgraph.

Algorithm 2 Transversing the minimum spanning trees of two subgraphs.

```
procedure TRANSVERSE(a, b)
    set_correspondence(a, b)
    mark(a)
    mark(b)
    repeat
        if unmarked(a.descendants) ≠ NULL then
            next_a ← RandomlySelect(unmarked(a.descendants))
        else if a.descendants ≠ NULL then
            next_a ← RandomlySelect(a.descendants)
        else
            next_a ← a
        end if
        **** do the same for next_b ****
        transverse(next_a, next_b)
    until unmarked(a.descendants) = unmarked(b.descendants) = NULL
end procedure
```

Subgraph selection – Randomly selects for each parent, a and b, one crossover node, v_a and v_b, and a subgraph radius, r_a and r_b. Subgraph s_{ra} is composed of all the nodes, and edges among them, that can be reached in a maximum of r_a steps starting from node v_a. Subgraph s_{rb} is defined analogously. Two methods were tested for choosing v_a and v_b, one assuring that both v_a and v_b are in the connected part of the graph and one without restrictions. The radius r_a and r_b were randomly chose being the maximum allowed value the maximum depth of the graph.

Swapping the subgraphs – Swapping s_{ra} and s_{rb} consists in replacing s_{ra} by s_{rb} (and vice-versa). After this operation the outgoing and the incoming edges are destroyed. Establishing a correspondence between nodes repairs these connections.

Correspondence of Nodes – Let s_{ra+1} and s_{rb+1} be the subgraphs that would be obtained by considering a subgraph radius of $r_a + 1$ and $r_b + 1$ while performing the subgraph selection. Let mst_a and mst_b be the minimum spanning trees (MSTs) with root nodes v_a and v_b connecting all s_{ra+1} and s_{rb+1} nodes, respectively. For determining the MSTs all edges are considered to have unitary cost. When several MSTs exist, the first one found is the one considered. The correspondence between the nodes of s_{ra+1} and s_{rb+1} is established by transversing mst_a and mst_b, starting from their roots, as described in Algorithm 2.

Restoring outgoing and incoming edges – The edges from a $\notin s_{ra}$ to s_{ra} are replaced by edges from a $\notin s_{rb}$ to s_{rb} using the correspondence between the nodes established in the previous step (e.g. the incoming edges to v_a are redirected to v_b, and so on). Considering a radius of $r_a + 1$ and $r_b + 1$ instead of r_a and r_b in the previous step allows the restoration of the outgoing edges. By definition, all outgoing edges from s_a and s_b link

to nodes that are at a minimum distance of $r_a + 1$ and $r_b + 1$, respectively. This allows us to redirect the edges from s_b to b $\notin s_b$ to a $\notin s_a$ using the correspondence list.

1.4.3 Mutation Operators

The mutation operators were designed to attend two basic goals: allowing the introduction of new genetic material in the population and ensuring that the search space is fully connected, i.e., that all of its points are reachable from any starting point through the successive application of mutation operators. This resulted in the use of a total of ten operators, which are succinctly described on the following paragraphs.

Startshape mutate – randomly selects a non-terminal as starting symbol.

Replace, Remove or Add symbol – when applied to a given production rule, these operators: replace one of the present symbols with a randomly selected one; remove a symbol and associated parameters from the production rule; add a randomly selected symbol in a valid random position. Notice that these operators are applied to terminal and non-terminal symbols.

Duplicate, Remove or Copy & Rename rule – these operators: duplicate a production rule; remove a production rule, updating the remaining rules when necessary; copy a production rule, assigning a new randomly created name to the rule and thus introducing a new non-terminal.

Change, Remove or Add parameter – as the name indicates, these operators add, remove or change parameters and parameter values. The change of parameter values is accomplished using a Gaussian perturbation.

1.5 Fitness Assignment

Fitness assignment implies interpreting and rendering the CFDG. This is accomplished by calling the Context Free [9] application. Grammars with infinite recursive loops are quite common. As such, it was necessary to establish an upper bound to the number of steps that a CFDG is allowed to make before its expansion is considered complete. The original version of Context Free only allows the definition of an upper bound for the number of drawn shapes. This is insufficient for our goals, because it allows endless loops, provided that no shapes are drawn. As such, it was necessary to introduce several changes to the source code of Context Free (which is open source) to accommodate our needs. When calling Context Free we give as input (i) the CFDG to be interpreted and rendered, (ii) the rendering size, (iii) the maximum number of steps (iv) the rendering seed. We receive as output an image file. The maximum number of steps was set to 100000 for all the experiments described in this Chapter. The "rendering seed" defines the seed of the random number

generator used by Context Free during the expansion of the CFDGs. The rendering of the same CFDG using different rendering seeds can, and often does, result in different images (see Section 1.3). We performed tests using fixed and randomly generated rendering seeds. The results of those tests will be described in Section 1.6.

We use six different hardwired fitness functions based on evolutionary art literature and conduct tests using each of these functions to guide evolution. In a second stage, we perform runs using a combination of these measures to assign fitness. In the reminder of this Section we describe each of the functions and the procedure used to combine them.

JPEG Size

The image returned by Context Free is encoded in JPEG format using the maximum quality settings. The size of the JPEG file becomes the fitness of the individual. The rationale is that complex images, with abrupt transitions of color are harder to compress and hence result in larger file sizes, whereas simple images will result in small file sizes [12, 16]. Although this assignment scheme is rather simplistic, it has the virtue of being straightforward to implement and yield results that are easily interpretable. As such, it was used to assess the ability of the evolutionary engine to complexify and to establish adequate experimental settings.

Number of Contrasting Colors

As the name indicates, the fitness of an individual is equal to the number of contrasting colors present in the image returned by Context Free. To calculate the number of contrasting colors we: (i) reduce the number of colors using a quantization algorithm; (ii) sort all colors present in the image by descending order of occurrence; (iii) for all the colors, starting from the most frequent ones, compute the Euclidean distance between the color and the next one in the ordered list, if it is lower than a certain threshold remove it from the group; (iv) return as fitness the number of colors present on the list when the procedure is over. In these experiments, the Red, Green, Blue (RGB) color space was adopted. We quantize the image to 256 colors using the quantization algorithm from the Graphics Interchange Format (GIF) format [10]. The threshold was set to 1% of the maximum Euclidean distance between colors (255^3 for the RGB color space).

Fractal Dimension, Lacunarity

The use of fractal dimension estimates in the context of computational aesthetic has a significant tradition [26, 18]. Although not as common, lacunarity measures have also been used [2, 3]. For the experiments described in this

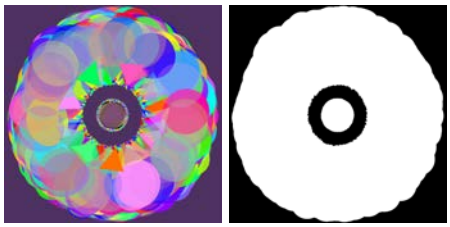

Fig. 1.4: Example of the transformation from the input color image (left image) to the background/foreground image (right image) used for the *Fractal Dimension* and *Lacunarity* estimates.

Chapter the fractal dimension is estimated using the box-counting method and the λ lacunarity value estimated by the Sliding Box method [11]. By definition, the estimation of the fractal dimension and lacunarity requires identifying the "object" that will be measured. Thus, the estimation methods take as input a binary image (i.e. black and white), where the white pixels define the shape that will be measured, while the black pixels represent the background. In our case, the conversion to black and white is based on the CFDG background primitive. All the pixels of the same color as the one specified by the CFDG background primitive are considered black, and hence part of the background, the ones that are of a different color are considered part of the foreground (see Figure 1.4). Once the estimates are computed we assign fitness according to the proximity of the measure to a desired value, as follows:

$$fitness = \frac{1}{1 + |target_{value} - observed_{value}|} \tag{1.1}$$

We use the target values of 1.3 and 0.90 for fractal dimension and lacunarity, respectively. These values were established empirically by calculating the fractal dimension and lacunarity of images that we find to have desirable aesthetic qualities.

Complexity

This fitness function, based on the work of Machado et al. [12, 16, 15], assesses several characteristics of the image related with complexity. In simple terms, the rationale is valuing images that constitute a complex visual stimulus but that are, nevertheless, easy to process. A thorough discussion of the virtues and limitations of this approach is beyond the scope of this Chapter, as such, we focus on practical issues pertaining its implementation. The approach relies on the notion of compression complexity, which is defined as calculated using the following formula:

$$C(i, scheme) = RMSE(i, scheme(i)) \times \frac{s(scheme(i))}{s(i)} \tag{1.2}$$

where i is the image being analysed, *scheme* is a lossy image compression scheme, $RMSE$ stands for the root mean square error, and s is the file size function.

To estimate the complexity of the visual stimulus ($IC(i)$) they calculate the complexity of the JPEG encoding of the image (i.e. $IC(i) = C(i, JPEG)$). The processing complexity ($PC(i)$) is estimated using a fractal (quadratic tree based) encoding of the image [7]. Considering that as time passes the level of detail in the perception of the image increases, the processing complexity is estimated for different moments in time ($PC(t0, i)$, $PC(t1, i)$) by using fractal image compression with different levels of detail. In addition to valuing images with high visual complexity and low processing complexity, the approach also values images where PC is stable for different levels of detail. In other words, according to this approach, an increase in description length should be accompanied by an increase in image fidelity. Taking all of these factors into consideration, Machado et al. [12, 16, 15] propose the following formula for fitness assignment:

$$\frac{IC(i)^a}{(PC(t0,i) \times PC(t1,i))^b \times (\frac{PC(t1,i)-PC(t0,i)}{PC(t1,i)})^c} \tag{1.3}$$

where a, b and c are parameters to adjust the importance of each component.

Based on previous work [15], the ability of the evolutionary engine to exploit the limitations of the complexity estimates was minimized by introducing limits to the different components of this formula, as follows:

$$\begin{cases} IC(i) & \rightarrow max(0, \alpha - |IC(i) - \alpha|) \\ PC(t0,i) \times PC(t1,i) & \rightarrow \gamma + |(PC(t0,i) \times PC(t1,i)) - \gamma| \\ PC(t1,i) - PC(t0,i) & \rightarrow \delta + |(PC(t1,i) - PC(t0,i)) - \delta| \end{cases} \tag{1.4}$$

where α, γ and δ operate as target values for $IC(i)$, $(PC(t0, i) \times PC(t1, i)$ and $PC(t1, i) - PC(t0, i)$, which were set to 6, 24 and 1.1, respectively. These values were determined empirically through the analysis of images that we find to be desirable. Due to the limitations of the adopted fractal image compression scheme this approach only deals with greyscale images. Therefore, all images are converted to greyscale before being processed.

Bell

This fitness function is based on the work of Ross et al. [23] and relies on the observation that many fine-art works exhibit a normal distribution of color gradients. Following Ross et al. [23] the gradients of each color channel are calculated, one by one, in the following manner:

$$|\nabla r_{i,j}|^2 = \frac{(r_{i,j} - r_{i+1,j+1})^2 + (r_{i+1,j} - r_{i,j+1})^2}{d^2} \tag{1.5}$$

where $r_{i,j}$ is the image pixel intensity values for position (i, j) and d is a scaling factor that allows to compare images of different size; this value was set to 0.1% of half the diagonal of the input image (based on [23]). Then the overall gradient $S_{i,j}$ is computed as follows:

$$S_{i,j} = \sqrt{|\nabla r_{i,j}|^2 + |\nabla g_{i,j}|^2 + |\nabla b_{i,j}|^2} \tag{1.6}$$

Next, the response to each stimulus $R_{i,j}$ is calculated:

$$R_{i,j} = \log \frac{S_{i,j}}{S_0} \tag{1.7}$$

Where S_0 is a detection threshold (set to 2 as indicated in [23]). Then the weighted mean (μ) and standard deviation (σ^2) of the stimuli are calculated as follows:

$$\mu = \frac{\sum_{i,j} {R_{i,j}}^2}{\sum_{i,j} R_{i,j}} \tag{1.8}$$

$$\sigma^2 = \frac{\sum_{i,j} R_{i,j}(R_{i,j} - \mu)^2}{\sum_{i,j} R_{i,j}} \tag{1.9}$$

At this step we introduce a subtle but important change to Ross et al. [23] work: we consider a lower bound for the σ^2, which was empirically set to 0.7. This prevents the evolutionary engine to converge to monochromatic images that, due to the use of a small number of colors, trivially match a normal distribution. This change has a profound impact in the experimental results, promoting the evolution of colorful images that match a normal distribution of gradients.

Using μ, σ^2 and the values of $R_{i,j}$ a frequency histogram with a bin size of $\sigma/100$ is created, which allows calculating the deviation from normality (DFN). The DFN is computed using q_i, which is the observed probability and p_i, the expected probability considering a normal distribution. Ross et al. [23] uses:

$$DFN = 1000 \cdot \sum p_i \log \frac{p_i}{q_i} \tag{1.10}$$

However, based on the results of preliminary runs using this formulation, we found that we consistently obtained better results using:

$$DFN_s = 1000 \cdot \sum (p_i - q_i)^2 \tag{1.11}$$

Which measures the squares of the differences between expected and observed probabilities. Therefore, in the experiments described in this Chapter Bell fitness is assigned according to the following formula: $1/(1 + DFN_s)$.

Table 1.1: Parameters used for the experiments described in this Chapter.

Initialization (see algorithm 1)	Values
min, max number of symbols	(1,3)
min, max number of rules	(1,3)
min, max calls per production rule	(1,2)
Evolutionary Engine	**Values**
Number of runs	30
Number of generations	100
Population size	100
Crossover probability	0.6
Mutation probability	0.1
Tournament size	10
Elite size	Top 2% of the population
CFDG Parameters	**Values**
Maximum number expansion steps	100000
Limits of the geometric transformations	rotate ∈ [0,359], size ∈ [-5,5] x ∈ [-5,5], y ∈ [-5,5], z ∈ [-5,5] flip ∈ [-5,5], skew ∈ [-5,5]
Limits of the color transformations	hue ∈ [0,359], saturation ∈ [-1,1] brightness ∈ [-1,1], alpha ∈ [-1,1]
Terminal symbols	SQUARE, CIRCLE, TRIANGLE

1.5.1 Combining Different Functions

In addition to the tests where the fitness functions described above were used to guide evolution, we conducted several experiments where the goal was to simultaneously maximize several of these functions. This implied producing a fitness score from multiple functions, which was accomplished using the following formula:

$$combined_{fitness}(i) = \prod_{j} \log\left(1 + f_j(i)\right) \tag{1.12}$$

where i is the image being assessed and f_j refers to the functions being considered. Thus, to assign fitness based on the *Complexity* and *Bell* functions we compute: $log(1+Complexity(i)) \times log(1+Bell(i))$. By adopting logarithmic scaling and a multiplicative fitness function we wish to promote the discovery of images that maximize all the measures being considered in the experiment.

1.6 Configuring the Evolutionary Engine

The evolutionary engine has several novel characteristics that differentiate it from conventional GP approaches. Therefore, it was necessary to conduct a series of tests to assess the adequacy of the engine for the evolution of CFDGs and to determine a reasonable set of configuration parameters. These

tests were conducted using *JPEG Size* as fitness function and allowed us to establish the experimental parameters summarized in Table 1.1, which are used throughout all the experiments described herein. In general, the results show that the engine is not overly sensitive to the configuration parameters, depicting an adequate behavior for a wide set of parameter configurations. Although the optimal parameters settings are likely to depend on the fitness function, a detailed parametric study is beyond the scope of this Chapter. Therefore, we did not attempt to find an optimal combination of parameters.

The use of a graph-based representation and genetic operators is one of the novel aspects of our approach. The use of such operators may introduce changes to the graph that may make some of the nodes (i.e. some production variables) unreachable from the starting node. For instance, a mutation of the node *Edera* of Figure 1.1 may remove the call to node *Ciglio* making most of the graph unreachable. Although, unreachable nodes have no impact on the phenotype, their existence may influence the evolutionary process. On one hand they may provide space for neutral variations and promote evolvability (unreachable nodes may become reattached by subsequent genetic operators), on the other they may induce bloat since they allow protection from destructive crossover. To study the impact of unreachable nodes in the evolutionary process we considered three variations of the algorithm:

Unrestricted – The crossover points are chosen randomly;
Restricted – The crossover points are chosen randomly from the list of reachable nodes of each parent;
Restricted with Cleaning – In addition to enforcing the crossover to occur in a reachable region of the graph, after applying crossover and mutation all unreachable nodes are deleted.

Figure 1.5 summarizes the results of these tests depicting the best and average fitness for each population. As it can be observed, although the behaviors of the three different approaches are similar, the restricted versions consistently outperform the unrestricted implementation by a small, yet statistically significant, margin. The differences between the restricted approaches are not statistically significant.

The differences among the three approaches become more visible when we consider the evolution of the number of reachable and unreachable nodes through time. As it can be observed in Figure 1.6, without cleaning, the number of unreachable nodes grows significantly, clearly outnumbering the number of reachable nodes. The number of reachable nodes of the restricted versions is similar, and smaller than the one resulting from the unrestricted version. Although cleaning does not significantly improve fitness in comparison with the restricted version, the reduction of the number of rules implies a reduction of the computational cost of interpreting the CFDGs and applying the crossover operators. As such, taking these experimental findings into consideration, we adopt the *Restricted with Cleaning* variant in all further tests.

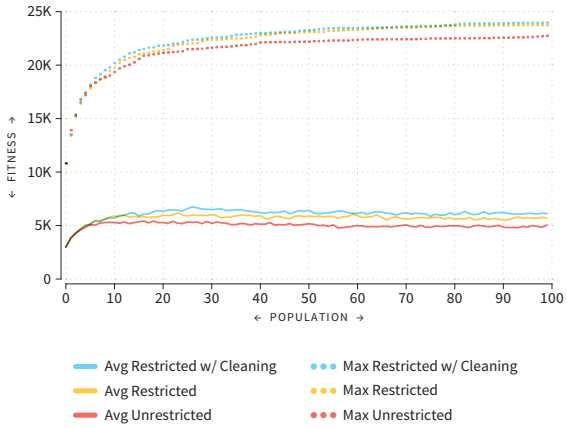

Fig. 1.5: Best and average fitness values for different implementations of the genetic operators using *JPEG Size* as fitness function. The results are averages of 30 independent runs.

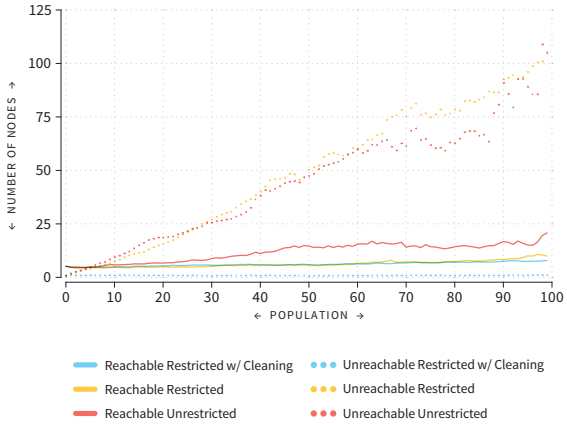

Fig. 1.6: Evolution of the average number of reachable and unreachable nodes across populations for different implementations of the genetic operators using *JPEG Size* as fitness function. The results are averages of 30 independent runs.

The non-deterministic nature of the CFDGs implies that each genotype may be mapped into a multitude of phenotypes (see Section 1.3). The genotype to phenotype mapping of a non-deterministic grammar depends on a rendering seed, which is passed to Context Free. We considered two scenarios: using a fixed rendering seed for all individuals; randomly generating the rendering seed whenever genotype to phenotype occurs. The second option implies that the fitness of a genotype may, and often does, vary from one evolution to the other, since the phenotype may change.

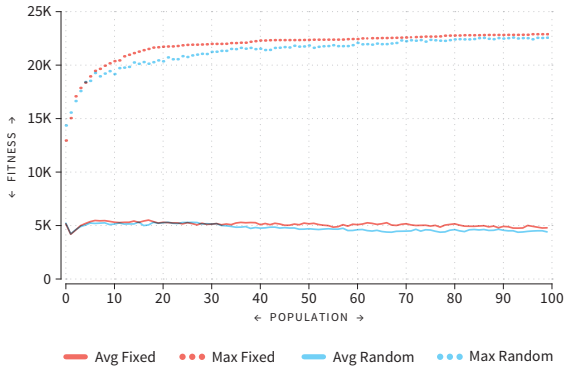

Fig. 1.7: Evolution of the best and average fitness across populations when using fixed and random rendering seeds using *JPEG Size* as the fitness function. The results are averages of 30 independent runs.

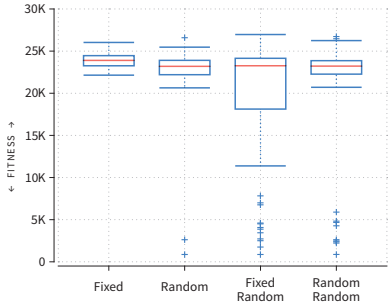

Fig. 1.8: Box plots of fitness values of the fittest individuals of each of the 30 evolutionary runs using different rendering seed setups.

Figure 1.7 summarizes the results of these tests in terms of the evolution of fitness through time. As expected, using a fixed rendering seed yields better fitness, but the differences between the approaches are surprisingly small and decrease as the number of generations increases. To better understand this result we focused on the analysis of the characteristics of the CFGDs being evolved. Figure 1.8 depicts box plots of fitness values of the fittest individuals of each of the 30 evolutionary runs using different setups:

Fixed – individuals evolved and evaluated using fixed rendering seeds; Random – individuals evolved using random rendering seeds and evaluated using the same seeds as the ones picked randomly during evolution;

Fixed Random – individuals evolved using fixed rendering seeds and evaluated with 30 random seeds each;

Random Random – individuals evolved using random rendering seeds and evaluated with 30 random seeds each.

In other words, we take the genotypes evolved in a controlled static environment (fixed random seed) and place them in different environments, proceeding in the same way for the ones evolved in a changing environment. The analysis of the box plots shows that, in the considered experimental settings, the fitness of the individuals evolved in a fixed environment may change dramatically when the environmental conditions are different. Conversely, using a dynamic environment promotes the discovery of robust individuals that perform well under different conditions. Although this result is not unexpected, it was surprising to notice how fast the evolutionary algorithm was able to adapt to the changing conditions and find robust individuals. In future tests we wish to explore, and exploit, this ability. Nevertheless, for the purposes of this Chapter, and considering that the use of a fixed rendering seed makes the analysis and reproduction of the experimental results easier, we adopt a fixed rendering seed in all further tests presented in this Chapter.

1.7 Evolving Context Free Art

After establishing the experimental conditions for the evolutionary runs we conducted a series of tests using each of the fitness functions described in Section 1.5 to guide evolution. In a second step, based on the results obtained, we combined several of these measures performing further tests. The results of using each of the measures individually are presented in Section 1.7.1 while those resulting from the combination of several are presented in Section 1.7.2.

1.7.1 Individual Fitness Functions

Figure 1.9 summarizes the results of these experiments in terms of evolution of fitness. Each chart depicts the evolution of the fitness of the best individual when using the corresponding fitness function to guide evolution. The values yield by the other 5 fitness functions are also depicted for reference to illustrate potential inter-dependencies among fitness functions. The values presented in each chart are averages of 30 independent runs (180 runs in total). To improve readability we have normalized all the values by dividing each raw fitness value by the maximum value for that fitness component found throughout all the runs.

The most striking observation pertains the *Fractal Dimension* and *Lacunarity* fitness functions. As it can be observed, the target values of 1.3 and 0.9 are easily approximated even when these measures are not used to guide fitness. Although this is a disappointing result, it is an expected one. Estimating the fractal dimension (or lacunarity) of an object that is not a fractal and that can be described using Euclidean geometry yields meaningless results. That is, although you obtain a value, this value is meaningless in the sense that there is no fractal dimension to be measured. As such, these measures may fail to capture any relevant characteristic of the images. In the considered

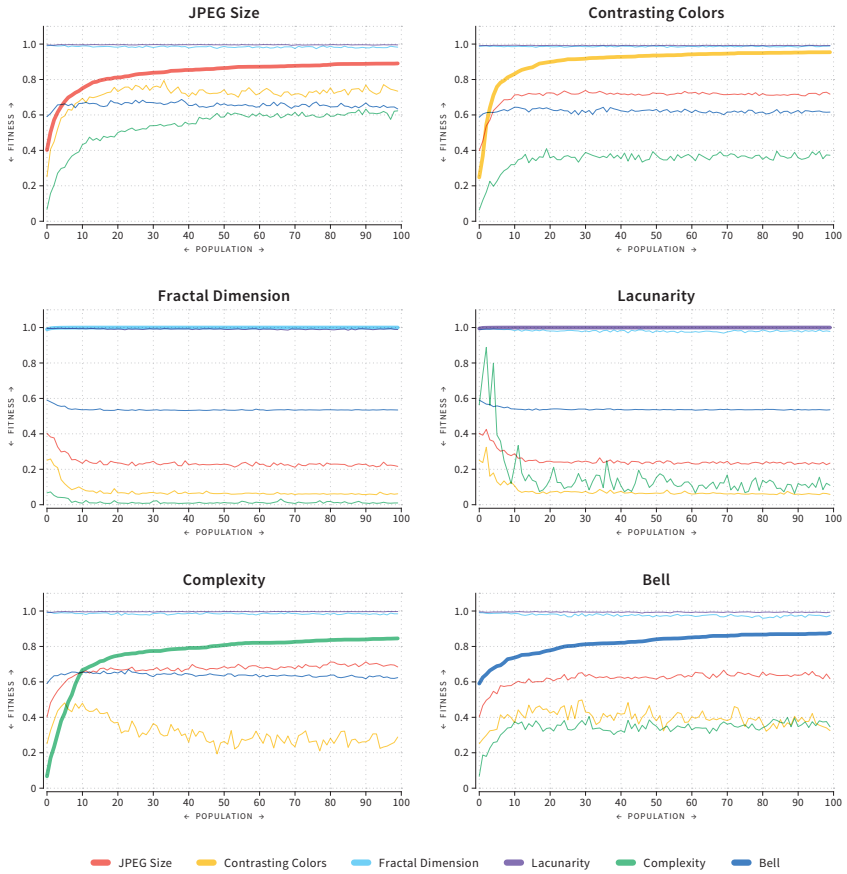

Fig. 1.9: Evolution of the fitness of the best individual across populations. The fitness function used to guide evolution is depicted in the title of each chart. The other values are presented for reference. The results are averages of 30 independent runs for each chart.

experimental conditions, the evolutionary algorithm was always able to find, with little effort, non-fractal images that yield values close to the target ones. Most often than not, these images are rather simplistic. We conducted several tests using different target values, obtaining similar results.

An analysis of the results depicted in Figure 1.9 reveals that maximizing *JPEG Size* promotes *Contrasting Colors* and *Complexity*, but does not promote a distributing of gradients approaching a normal distribution (*Bell*). Likewise, maximizing *Contrasting Colors* originates an improvement in *JPEG Size* and *Complexity* during the early stages of the evolutionary process; *Bell* is mostly unaffected. Using *Complexity* to guide evolution results in an increase of *JPEG Size* and *Contrasting Colors* during the early stages of the

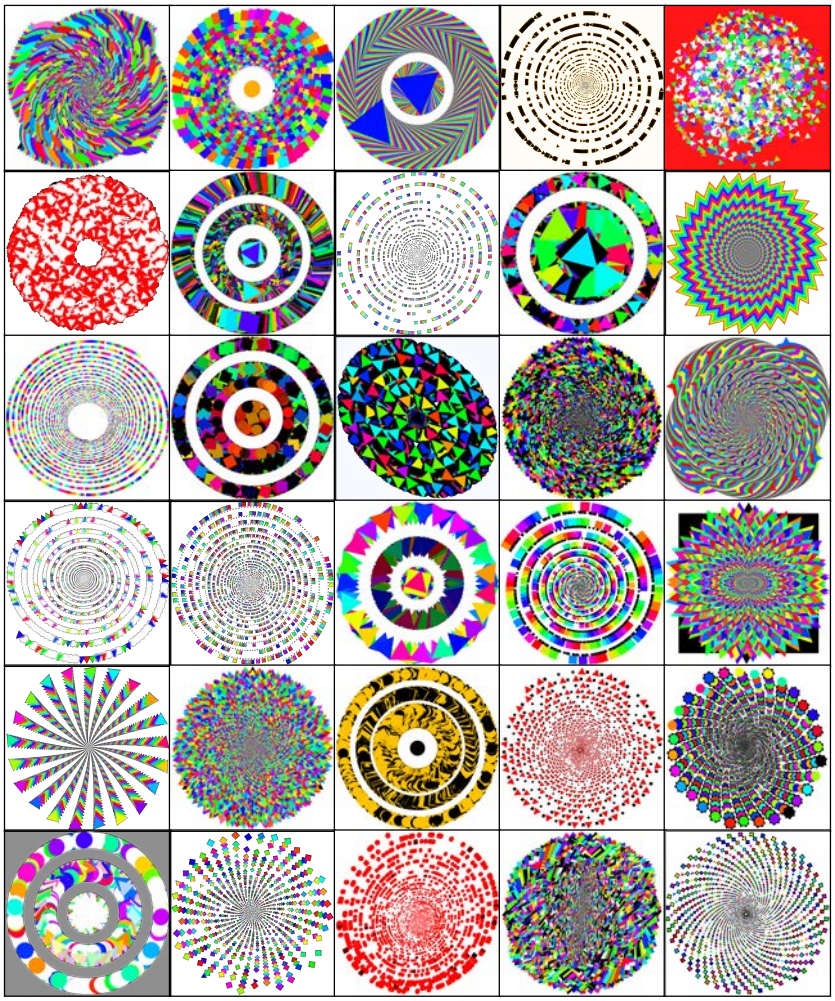

Fig. 1.10: Best individual of each of the 30 runs using *JPEG Size* as fitness function.

runs, but the number of *Contrasting Colors* tends to decrease as the number of generations progresses. The *Complexity* fitness function operates on a greyscale version of the images, as such it is not sensitive to changes of color. Furthermore, abrupt changes from black to white create artifacts that are hard to encode using JPEG compression, resulting in high IC estimates. Fractal image compression, which is used to estimate PC, is less sensitive to these abrupt changes. Therefore, since the approach values images with high IC and low PC, and since it does not take color information into consideration, the convergence to images using a reduced palette of contrasting colors is expected. Like for the other measures, *Complexity* and *Bell* appear to be

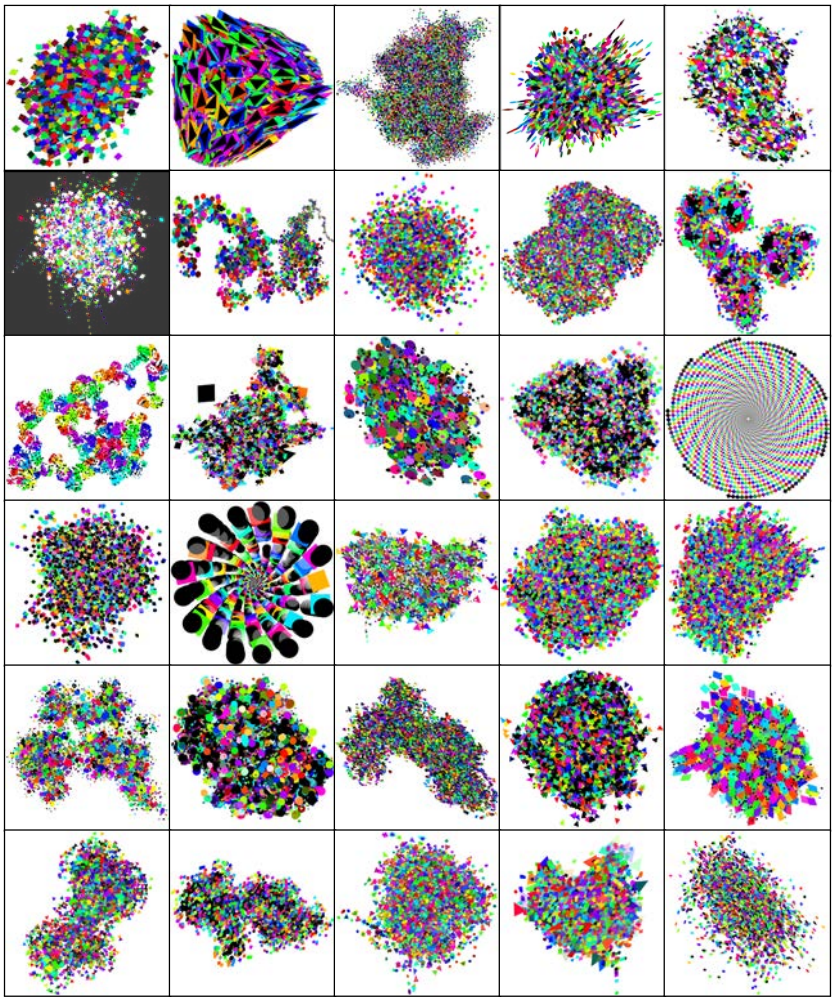

Fig. 1.11: Best individual of each of the 30 runs using *Contrasting Colors* as fitness function.

unrelated. Finally, maximizing *Bell* promotes an increase of *JPEG Size*, *Contrasting Colors* and *Complexity* during the first generations. It is important to notice that this behavior was only observed after enforcing a lower bound for σ^2 (see Section 1.5). Without this limit, maximizing *Bell* results in the early convergence to simplistic monochromatic images (typically a single black square on a white background). The adoption of a quadratic deviation from normality estimate (DFN_s) also contributed to the improvement of the visual results.

Figures 1.10 to 1.15 depict the best individual of each evolutionary run using the different fitness functions individually. A degree of subjectivity in the

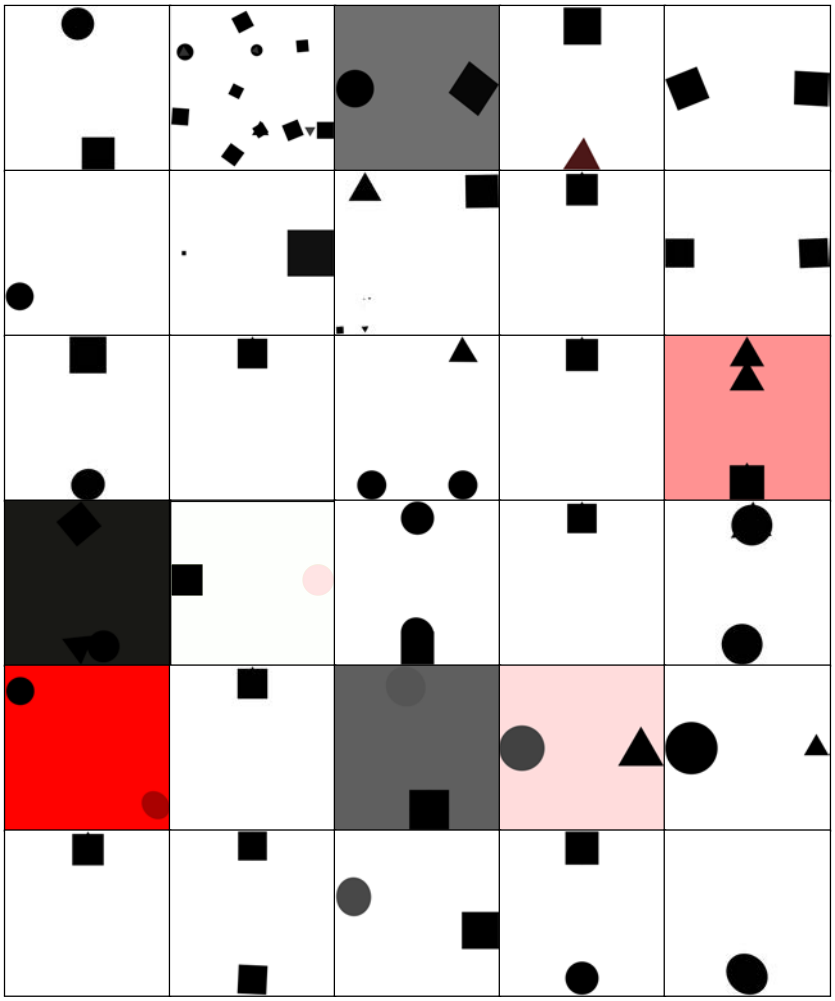

Fig. 1.12: Best individual of each of the 30 runs using *Fractal Dimension* as fitness function.

analysis of the visual results is unavoidable. Nevertheless, we believe that most of the findings tend to be consensual. When using *JPEG Size* to guide evolution, the evolutionary engine tended to converge to colorful circular patterns, with high contrasts of color (see Figure 1.10). The tendency to converge to circular patterns, which is observed in several runs, is related with the recursive nature of the CFDGs and the particularities of the Context Free rendering engine. For instance, repeatedly drawing and rotating a square while changing its color will generate images that are hard to encode. Furthermore, the rendering engine automatically "zooms in" the shapes drawn cropping the empty regions of the canvas. As such, rotating about a fixed point in space tends to

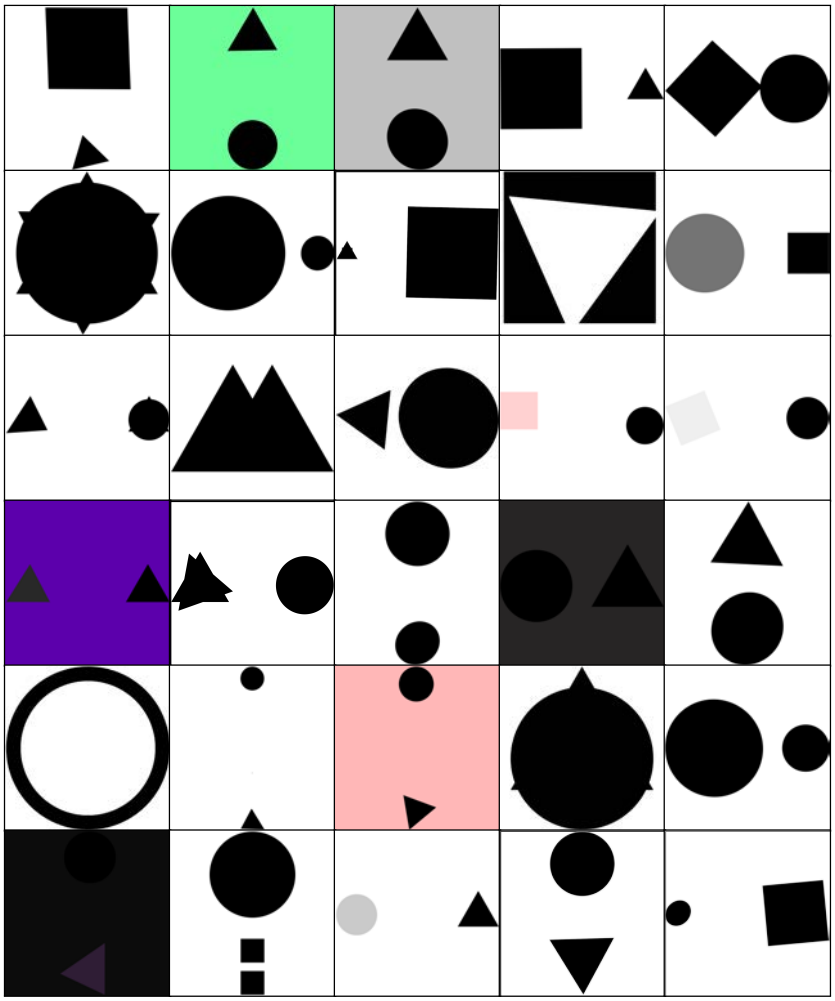

Fig. 1.13: Best individual of each of the 30 runs using *Lacunarity* as fitness function.

result in images that fill the entire canvas, maximizing the opportunities for introducing abrupt changes and, therefore, maximizing file size. Additionally, these CFDGs tend to be relatively stable and robust, which further promotes the convergence to this type of image.

Unsurprisingly, the results obtained when using *Contrasting Colors* are characterized by the convergence to images that are extremely colorful. Although some exceptions exist, most runs converged to amorphous unstructured shapes, which contrasts with circular patterns found when using *JPEG Size*. In our opinion this jeopardizes the aesthetic appeal of the images, that tend to have a random appearance, both in terms of shape and color.

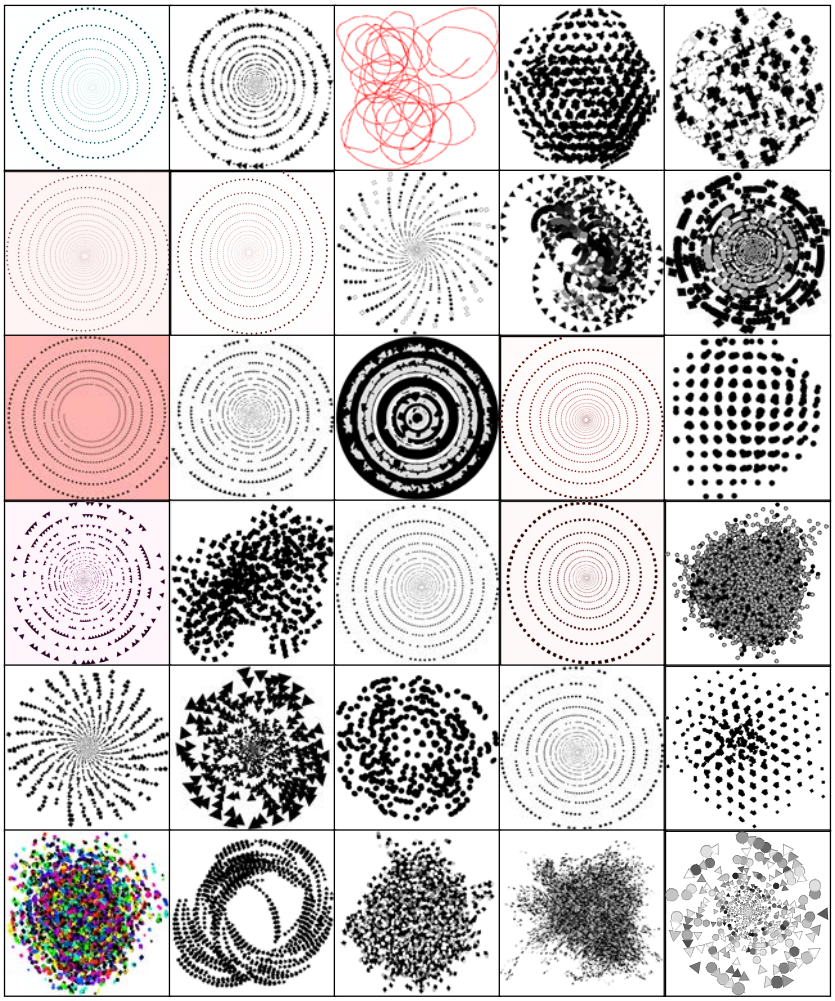

Fig. 1.14: Best individual of each of the 30 runs using *Complexity* as fitness function.

As anticipated by the data pertaining the evolution of fitness, the visual results obtained using *Fractal Dimension* and *Lacunarity* (Figs. 1.12 and 1.13 are disappointing. None of the runs converged to images of fractal nature. These results reinforce earlier findings using expression based evolutionary art systems, indicating that these measures are not suitable for aesthetically driven evolution [16].

As Figure 1.14 illustrates, using *Complexity* tends to promote convergence to monochromatic and highly structured images. As previously, the tendency to converge to circular and spiral patterns is also observed in this case, and is explained by the same factors. Furthermore, since fractal image compression

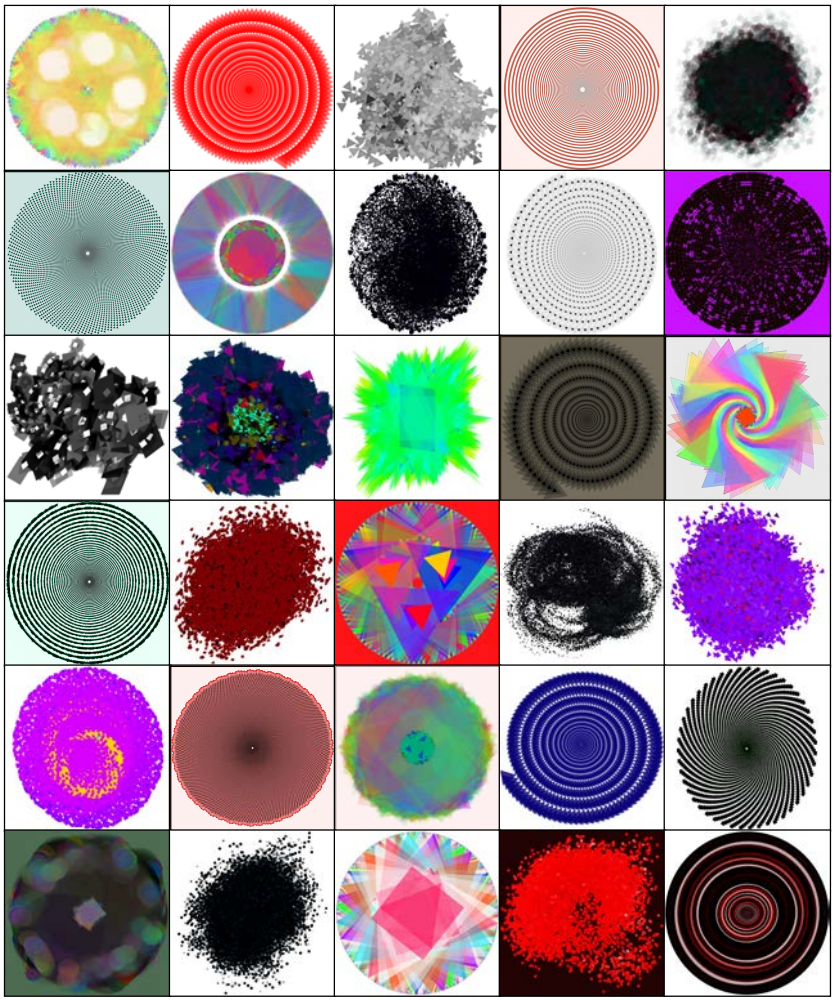

Fig. 1.15: Best individual of each of the 30 runs using *Bell* as fitness function.

takes advantage of the self-similarities present in the image at multiple scales, the convergence to structured and self-similar structures that characterizes these runs was expected. As mentioned when analysing results pertaining the evolution of fitness, the convergence to monochromatic images with high contrast is due to the different sensitivity of JPEG and fractal compression to the presence of abrupt transitions.

The most predominant feature of the images evolved using *Bell*, Figure 1.15, is the structured variation of color, promoted by the need to match a natural distribution of color gradients. The shapes evolved result from an emergent property of the system. In other words, as previously explained, when using CFDG a circular pattern is easily attainable and provides the

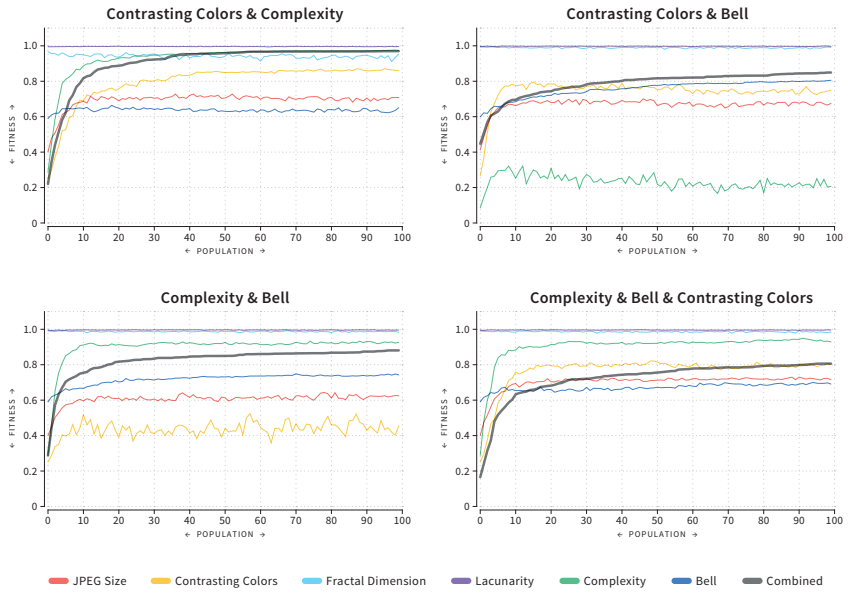

Fig. 1.16: Evolution of the fitness of the best individual across populations using a combination of measures. The combination used to guide evolution is depicted in the title of each chart. The other values are presented for reference, but have no influence in the evolutionary process. The results are averages of 30 independent runs for each chart and have have been normalized to improve readability.

conditions for reaching a natural distribution of color gradients. Although this is not visible in Figure 1.15 the individuals reaching the highest fitness values tend to use a large color palette.

1.7.2 Combining Several Measures

We performed several experiments where a combination of measures was used to assign fitness (see Section 1.5.1). We conducted tests combining *Fractal Dimension* and *Lacunarity* with other measures, these results confirm that these measures are ill-suited for aesthetic evolution in the considered experimental setting. Tests using *JPEG Size* in combination with other measures were also performed. The analysis of the results indicates that they are subsumed and surpassed by those obtained when using *Complexity* in conjunction with other metrics. This results from two factors: on one hand *Complexity* already takes into account the size of the JPEG encoding; on the other the limitations of *Complexity* regarding color are compensated by the use of measures that are specifically designed to handle color information. As such, taking into account

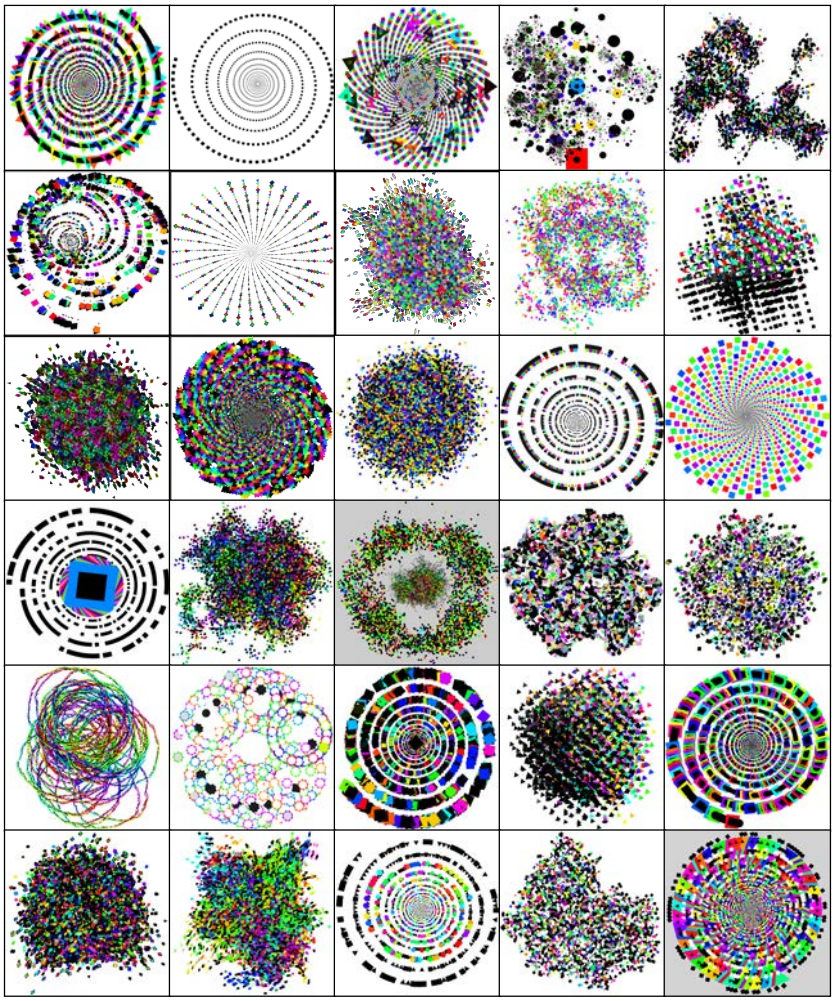

Fig. 1.17: Best individual of each of the 30 runs using the combination of *Contrasting Colors* with *Complexity* as fitness function.

the results described in the previous Section, as well as space constraints, we focus on the analysis of the results obtained when combining: *Contrasting Colors*, *Complexity* and *Bell*.

Figure 1.16 summarizes the results of these experiments in terms of evolution of fitness. Each chart depicts the evolution of the fitness of the best individual when using the corresponding combination of measures as fitness function. The values yield by the remaining measures are depicted but do not influence evolution. The values presented in each chart are averages of 30 independent runs (120 runs in total). As previously, the values have been

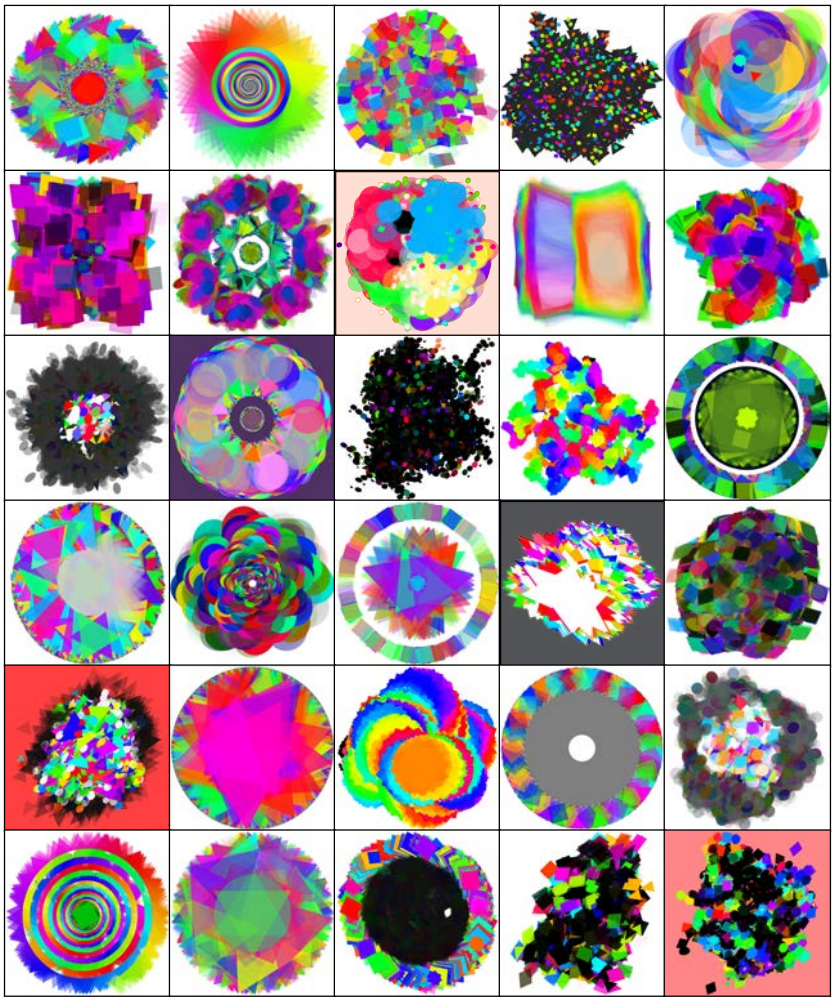

Fig. 1.18: Best individual of each of the 30 runs using the combination of *Contrasting Colors* with *Bell* as fitness function.

normalized by dividing each raw fitness value by the maximum value for that fitness component found throughout all the runs.

As it can be observed, combining *Contrasting Colors* and *Complexity* leads to a fast increase of both measures during the early stages of the runs, followed by a steady increase of both components throughout the rest of the runs. This shows that, although the runs using *Complexity* alone converged to monochromatic imagery, it is possible to evolve colorful images that also satisfy the *Complexity* measure.

Combining *Contrasting Colors* and *Bell* results in a rapid increase of the number of contrasting colors during the first generations. Afterwards,

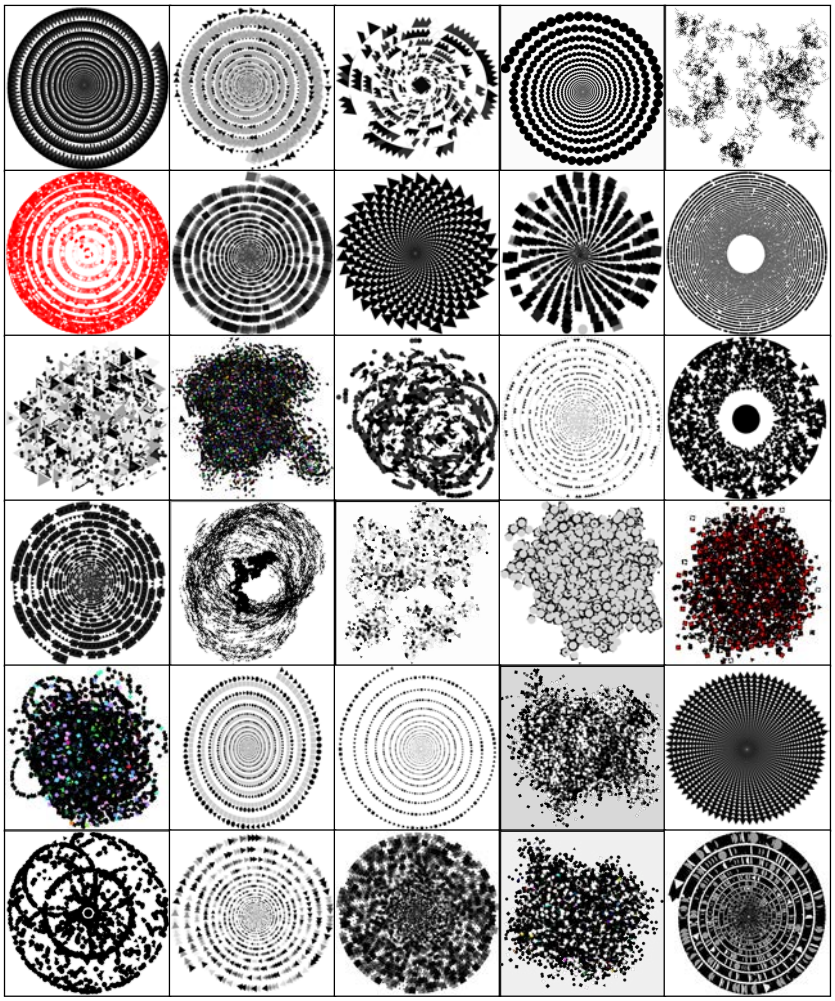

Fig. 1.19: Best individual of each of the 30 runs using the combination of *Complexity* with *Bell* as fitness function.

increases in fitness are mainly accomplished through the improvements of the *Bell* component of the fitness function. This indicates that it is easier to maximize the number of contrasting colors than to attain a normal distribution of gradients. This observation is further attested by the analysis of the charts pertaining the evolution of fitness when using *Contrasting Colors*, *Complexity* and *Bell* individually, which indicate that *Bell* may be the hardest measure to address. The combination of *Complexity* and *Bell* is characterized by a rapid increase of complexity during the first populations, followed by a slow, but steady, increase of both measures throughout the runs. The combination of the three measures further establishes *Bell* as the measure that is most diffi-

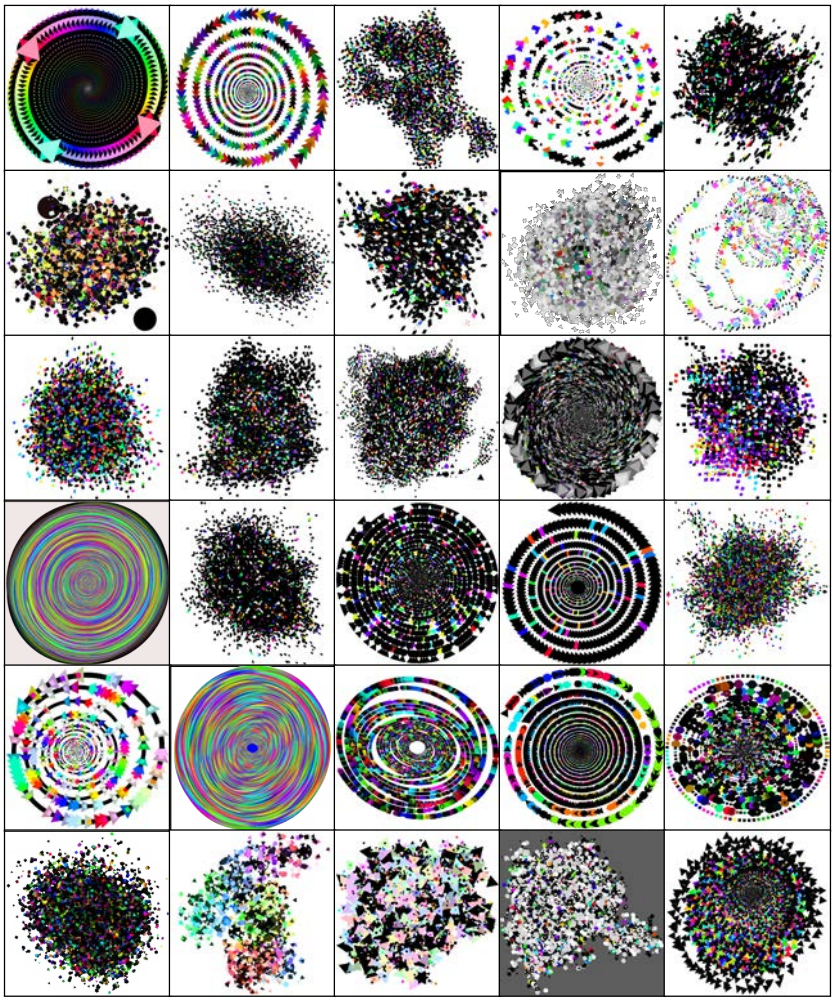

Fig. 1.20: Best individual of each of the 30 runs using the combination of *Contrasting Colors*, *Complexity* and *Bell* as fitness function.

cult to address, since the improvements of fitness are mostly due to increases in the other two measures. Significantly longer runs would be necessary to obtain noteworthy improvements in *Bell*.

Figure 1.17 depicts the best individual of each evolutionary run using as fitness a combination of the *Contrasting Colors* and *Complexity* measures. As it can be observed, in most cases, the neat structures that characterize the runs using *Complexity* (see Figure 1.14) continue to emerge. However, due to the influence of the *Contrasting Colors* measure, they tend to be colorful instead of monochromatic. Thus, the visual results appear to depict a good combination of both measures. The same can be stated for the images resulting from using

Contrasting Colors and *Bell*. As can be observed in Figure 1.18, they are more colorful than those evolved using *Bell* (see Figure 1.15) but retain a natural distribution of color gradients, deviating from the "random" coloring schemes that characterize the images evolved using *Contrasting Colors* (see Figure 1.11).

The images obtained when using *Complexity* and *Bell* simultaneously (Figure 1.19) are less colorful than expected. Visually, the impact of *Complexity* appears to overshadow the impact of *Bell*. Nevertheless, a comparison between these images and those obtained using *Complexity* alone (Figure 1.14) reveals the influence of *Bell* in the course of the runs: the monochromatic images are replaced by ones with a wider number of color gradients, and these color changes tend to be subtler.

Finally, as expected, the images obtained in the runs using the three measures (Figure 1.20) often depict, simultaneously, the features associated with each of them. As previously, the influence of the *Bell* measure is less obvious than the others, but a comparison with the results depicted in Figure 1.17 highlights the influence of this measure. Likewise, the structures that emerge from runs using *Complexity* and the colorful images that characterize runs using *Contrasting Colors* are also less often. Thus, although the influence of each measure is observable, we consider that significantly longer runs would be necessary to enhance their visibility.

1.8 Conclusions

We have presented a graph-based approach for the evolution of Context Free Design Grammars. This approach contrasts with the mainstream evolutionary art practices by abandoning expression-based evolution of images and embracing the evolution of images created through the combination of basic shapes. Nevertheless, the procedural nature of the representation, which characterizes Genetic Programming approaches, is retained. We describe the evolutionary engine, giving particular attention to its most discriminating features, namely: representation, graph-based crossover, mutation and initialization.

We introduce six different fitness functions based on evolutionary art literature and conduct a wide set of experiments. In a first step we assess the adequacy of the system and establish satisfactory experimental parameters. In this context, we study the influence of unexpressed genetic code in the evolutionary process and the influence of the environment in the robustness of the individuals. In the considered experimental settings, we find that restricting crossover to the portions of the genome that are expressed and cleaning unexpressed code is advantageous, and that dynamic environmental conditions promote the evolution of robust individuals.

In a second step, we conducted runs using each of the six fitness functions individually. The results show that *Fractal Dimension* and *Lacunarity* are

ill-suited for aesthetic evolution. The results obtained with the remaining fitness functions are satisfactory and correspond to our expectations. Finally, we conducted runs using a combination of the previously described measures to assign fitness. Globally, the experimental results illustrate the ability of the system to simultaneously address the different components taken into consideration for fitness assignment. They also show that some components are harder to optimize than others, and that runs using several fitness components tend to require a higher number of generations to reach good results.

One of the most prominent features of the representation adopted herein is its non-deterministic nature. Namely, the fact that a genotype may be mapped into a multitude of phenotypes, i.e. images, produced from different expansions of the same set of rules. As such, each genotype represents a family of shapes that, by virtue of being generated using the same set of rules, tend to be aesthetically and stylistically similar. The ability of the system to generate multiple phenotypes from one genotype was not explored in this Chapter, and will be addressed in future work. Currently we are conducting experiments where the fitness of a genotype depends on a set of phenotypes generated from it. The approach values genotypes which are able to consistently produce fit and diverse individuals, promoting the discovery of image families that are simultaneously coherent and diverse.

Acknowledgments

This research is partially funded by: Fundação para a Ciência e Tecnologia (FCT), Portugal, under the grant SFRH/BD/90968/2012; project ConCreTe. The project ConCreTe acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET grant number 611733. We acknowledge and thank the contribution of Manuel Levi who implemented the *Contrasting Colors* fitness function.

References

1. E. Baker and M. Seltzer. Evolving line drawings. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 91–100. Morgan Kaufmann Publishers, 1994.
2. J. Bird, P. Husbands, M. Perris, B. Bigge, and P. Brown. Implicit fitness functions for evolving a drawing robot. In *Applications of Evolutionary Computing*, pages 473–478. Springer, 2008.
3. J. Bird and D. Stokes. Minimal creativity, evaluation and fractal pattern discrimination. *Programme Committee and Reviewers*, page 121, 2007.
4. A. Borrell. CFDG Mutate. http://www.wickedbean.co.uk/cfdg/index.html, last accessed in November 2014.

5. C. Coyne. Context Free Design Grammar. http://www.chriscoyne.com/cfdg/, last accessed in November 2014.
6. E. den Heijer and A. E. Eiben. Evolving art with scalable vector graphics. In N. Krasnogor and P. L. Lanzi, editors, *GECCO*, pages 427–434. ACM, 2011.
7. Y. Fisher, editor. *Fractal Image Compression: Theory and Application*. Springer, London, 1995.
8. D. A. Hart. Toward greater artistic control for interactive evolution of images and animation. In *Proceedings of the 2007 EvoWorkshops 2007 on EvoCoMnet, EvoFIN, EvoIASP, EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog*, pages 527–536, Berlin, Heidelberg, 2007. Springer-Verlag.
9. J. Horigan and M. Lentczner. Context Free. http://www.contextfreeart.org/, last accessed in September 2009.
10. C. Incorporated. GIF Graphics Interchange Format: A standard defining a mechanism for the storage and transmission of bitmap-based graphics information. Columbus, OH, USA, 1987.
11. A. Karperien. Fraclac for imagej. In *http://rsb.info.nih.gov/ij/plugins/fraclac/FLHelp/ Introduction.htm*, 1999-2013.
12. P. Machado and A. Cardoso. All the truth about NEvAr. *Applied Intelligence, Special Issue on Creative Systems*, 16(2):101–119, 2002.
13. P. Machado and H. Nunes. A step towards the evolution of visual languages. In *First International Conference on Computational Creativity*, Lisbon, Portugal, 2010.
14. P. Machado, H. Nunes, and J. Romero. Graph-based evolution of visual languages. In C. D. Chio, A. Brabazon, G. A. D. Caro, M. Ebner, M. Farooq, A. Fink, J. Grahl, G. Greenfield, P. Machado, M. ONeill, E. Tarantino, and N. Urquhart, editors, *Applications of Evolutionary Computation, EvoApplications 2010: EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoMUSART, and EvoTRANSLOG, Istanbul, Turkey, April 7-9, 2010, Proceedings, Part II*, volume 6025 of *Lecture Notes in Computer Science*, pages 271–280. Springer, 2010.
15. P. Machado, J. Romero, A. Cardoso, and A. Santos. Partially interactive evolutionary artists. *New Generation Computing – Special Issue on Interactive Evolutionary Computation*, 23(42):143–155, 2005.
16. P. Machado, J. Romero, and B. Manaris. Experiments in computational aesthetics: an iterative approach to stylistic change in evolutionary art. In J. Romero and P. Machado, editors, *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*, pages 381–415. Springer Berlin Heidelberg, 2007.
17. J. McCormack. Facing the future: Evolutionary possibilities for human-machine creativity. In J. Romero and P. Machado, editors, *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*, pages 417–451. Springer Berlin Heidelberg, 2007.
18. T. Mori, Y. Endou, and A. Nakayama. Fractal analysis and aesthetic evaluation of geometrically overlapping patterns. *Textile research journal*, 66(9):581–586, 1996.
19. M. O'Neill, J. McDermott, J. M. Swafford, J. Byrne, E. Hemberg, A. Brabazon, E. Shotton, C. McNally, and M. Hemberg. Evolutionary design using grammatical evolution and shape grammars: Designing a shelter. *International Journal of Design Engineering*, 3(1):4–24, 2010.
20. M. O'Neill and C. Ryan. *Grammatical evolution: evolutionary automatic programming in an arbitrary language*, volume 4. Springer, 2003.

21. M. O'Neill, J. M. Swafford, J. McDermott, J. Byrne, A. Brabazon, E. Shotton, C. McNally, and M. Hemberg. Shape grammars and grammatical evolution for evolutionary design. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 1035–1042, New York, NY, USA, 2009. ACM.
22. F. B. Pereira, P. Machado, E. Costa, and A. Cardoso. Graph based crossover – a case study with the busy beaver problem. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, 1999.
23. B. J. Ross, W. Ralph, and Z. Hai. Evolutionary image synthesis using a model of aesthetics. In G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B.-T. Zhang, C. A. C. Coello, and T. P. Runarsson, editors, *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 1087–1094, Vancouver, BC, Canada, 16–21 July 2006. IEEE Press.
24. R. Saunders and K. Grace. Teaching evolutionary design systems by extending "Context Free". In *EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, pages 591–596. Springer-Verlag, 2009.
25. K. Sims. Artificial evolution for computer graphics. *ACM Computer Graphics*, 25:319–328, 1991.
26. B. Spehar, C. W. G. Clifford, N. Newell, and R. P. Taylor. Universal aesthetic of fractals. *Computers and Graphics*, 27(5):813–820, Oct. 2003.
27. G. Stiny and J. Gips. Shape grammars and the generative specification of paintings and sculpture. In C. V. Freiman, editor, *Information Processing 71*, pages 1460–1465, Amsterdam, 1971. North Holland Publishing Co.
28. T. Unemi. SBART2.4: Breeding 2D CG images and movies, and creating a type of collage. In *The Third International Conference on Knowledge-based Intelligent Information Engineering Systems*, pages 288–291, Adelaide, Australia, 1999.
29. L. World. Aesthetic selection: The evolutionary art of Steven Rooke. *IEEE Computer Graphics and Applications*, 16(1), 1996.