Graph-Based Evolution of Visual Languages

Penousal Machado[1], Henrique Nunes[1] and Juan Romero[2]

[1] CISUC, Department of Informatics Engineering, University of Coimbra, 3030 Coimbra, Portugal machado@dei.uc.pt
[2] Faculty of Computer Science, University of Coruña, Coruña, Spain

Abstract. We present a novel evolutionary engine for the evolution of context free grammars. The system relies on specially designed graph-based crossover and mutation operators. While in most evolutionary art systems each individual corresponds to a single artwork, in our approach each individual is a context free grammar that specifies a family of shapes following the same production rules. To assess the adequacy and completeness of the system we perform experiments using automated fitness assignment and user-guided evolution. The experimental results show that the system is able to create diverse and interesting families of shapes even when the initial population is composed of minimal grammars.

1 Introduction

The main inspiration of this research is the seminal work of Stiny and Gips [1] who introduced the concept of Shape Grammars and built, among others, shape grammars that capture the architectural "language" of Frank Lloyd Wright's prairie houses. Although the grammars were hand-built, their results show that: (i) it is possible to capture specific *visual languages* using a set of production rules; (ii) it is then possible to use this set of rules to automatically generate new objects that belong to the same visual language.

With the goal of developing a system that generates novel visual languages, we created an evolutionary engine where each individual is a Context Free Design Grammar (CFDG) [2] and built appropriate genetic operators for their manipulation. The use of CFDGs allows the specification of complex families of shapes through a compact set of rules, and has several potential advantages over several other Evolutionary Art (EA) representations. The development of a graph-based crossover operator was motivated by the need to take into account the underlying structure of the individuals while exchanging genetic code.

A thorough survey of EA systems is beyond the scope of this paper and can be found in [3]. To the best of our knowledge, there are two examples of the use of CFDG for EA. Unfortunately, neither of them allows the evolution of visual languages. *CFDG Mutate* [4] only allows the application of mutation operators and does not handle non-deterministic grammars, which means each individual represents a single shape (see Section 2). Saunders and Grace [5] present a parametric system that evolves parameters of specific CFDG hand-built grammars.

Although it allows some degree of exploration, it has the same shortcomings as other parametric evolution approaches: there are strong constraints that limit the search space and define the type of imagery produced by the system.

A previous work [6] showed that our engine allows the evolution of interesting, complex and diverse visual languages when the initial population is composed of hand-built CFDGs. In this paper we focus on the description of the evolutionary engine, namely crossover and mutation operators, and on testing its generation abilities in the absence of hand-built grammars.

2 Context Free

Context Free [7] is a popular open-source application that renders images specified using a simple language entitled CFDG (for a full description of CFDG see [2]). In essence, and although the notation is different from the one used in formal language theory, 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 nonterminal 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

Fig. 1 presents a grammar and the image it generates. Programs are interpreted by starting with S (in this case $S = TREE$) and proceeding by the expansion of the production rules in breath-first fashion. Predefined Σ symbols call drawing primitives (e.g., CIRCLE). CFDG is an *augmented* context free grammar: it takes parameters that produce semantic operations (e.g., *size* produces a scale change). Program interpretation is terminated when there are no V symbols left to expand or the further expansion does not change the image [6].

The grammar of Fig. 1 is deterministic, there is exactly one rule for each V symbol, therefore its interpretation will always result in the same image. To specify languages of shapes we have to resort to non-determinism. In Fig. 2 we present a non-deterministic version of this grammar with two different production rules for the 'TREE' symbol. When several production rules are applicable one of them is selected randomly and the expansion proceeds. One can control the probability of selection by specifying a weight after the V symbol (0.8 and 0.2 in Fig. 2).

3 Evolutionary Context Free Art

In this section we describe our evolutionary engine. For the sake of parsimony we focus on the key components of the system.

Each genotype is a well-constructed CFDG grammar. Phenotypes are rendered using Context Free. To deal with nonterminating programs a maximum rendering time is set. The genotype is represented by a directed graph created as follows:

```
startshape TREE
rule TREE  { CIRCLE {}
             TREEA {size 0.95 y 1.6}}
rule TREEA { CIRCLE {}
             TREEB {size 0.95 y 1.6}}
rule TREEB { CIRCLE {}
             TREEC {size 0.95 y 1.6}}
rule TREEC { CIRCLE {}
             TREED {size 0.95 y 1.6}}
rule TREED { CIRCLE {}
             TREE {size 0.95 y 1.6 rotate 45}
             TREE {size 0.95 y 1.6 rotate -45}}
```

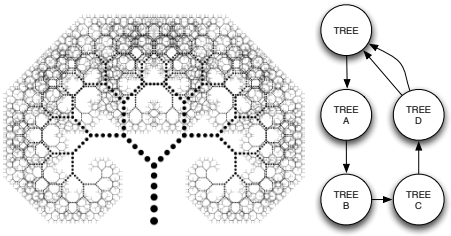

Fig. 1. A deterministic grammar, the tree-like shape it generates, and its graph representation. The labels of the edges have been omitted.

```
startshape TREE
rule TREE 0.80 {
     CIRCLE {}
     TREE {size 0.95 y 1.6}
}
rule TREE 0.20 {
     CIRCLE {}
     TREE {size 0.95 y 1.6
            rotate 45}
     TREE {size 0.95 y 1.6
            rotate -45}}
```

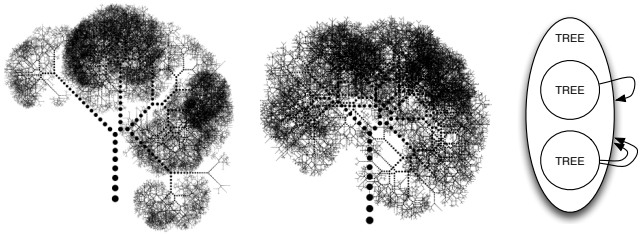

Fig. 2. A non-deterministic version of the grammar presented in Fig. 1, two instances of the family of tree-like shapes it defines and its graph representation. The labels of the edges have been omitted.

1. Create a node for each nonterminal symbol. In deterministic grammars each node represents a single production rule (see Fig. 1). In non-deterministic grammars each node encapsulates the set of all production rules associated with the nonterminal symbol, e.g., the grammar presented on Fig. 2 results in a directed graph composed of a single node that encapsulates the two rules associated with the nonterminal 'TREE'.
2. Create edges between each node and the nodes corresponding to the nonterminals appearing in its production rules (see Figs. 1 and 2).
3. Annotate each edge with the corresponding parameters (e.g. In Fig. 1 the edge connecting TREE with TREEA possesses the label 'size 0.95 y 1.6')

3.1 Crossover Operator

The design of genetic operators that are well-suited to the representation is vital for the success of an evolutionary algorithm. In our case the biggest challenge was to design a crossover operator that allows the meaningful exchange of genetic material between individuals. Given the nature of the representation, we developed a graph-based crossover operator based on the one presented by Pereira et al. [8]. 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.

Subgraph selection Randomly selects for each parent, a and b, one crossover node, v_a and v_b, and a subgraph radius, ra and rb. Subgraph s_{ra} is composed of all the nodes, and edges among them, that can be reached in a maximum of ra steps starting from node v_a. Subgraph s_{rb} is defined analogously.

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 $ra + 1$ and $rb + 1$ 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_{ra+1} is established by transversing mst_a and mst_b, starting from their roots, as described in Algorithm 1.

Restoring outgoing and incoming edges The edges from $a \notin s_{ra}$ to s_{ra} are replaced by edges from $a \notin s_{ra}$ 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 $ra + 1$ and $rb + 1$ instead of ra and rb 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 $ra + 1$ and $rb + 1$, 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.

Figs. 3 and 4 present examples of the crossover operator at the genotype and phenotype level.

3.2 Mutation Operators

The development of the mutation operators was guided by the need to introduce new genetic code and to ensure that the search space is fully connected, i.e., that all of its points are reachable from any starting point by the successive application of genetic operators. This resulted in the use of ten operators, for which, due to space limitations, we only present a cursory description:

Startshape mutate – randomly selects a nonterminal as starting symbol;

Algorithm 1 transverse(a, b)

$set\_correspondence(a, b)$
$mark(a)$
$mark(b)$
repeat
  if $unmarked(a.descendants) \neq NULL$ **then**
    $next_a \leftarrow RandomlySelect(unmarked(a.descendants))$
  else if $a.descendants \neq NULL$ **then**
    $next_a \leftarrow RandomlySelect((a.descendants))$
  else
    $next_a \leftarrow a$
  end if
  {**** do the same for $next_b$ ****}
  $transverse(next_a, next_b)$
until $unmarked(a.descendants) = unmarked(b.descendants) = NULL$

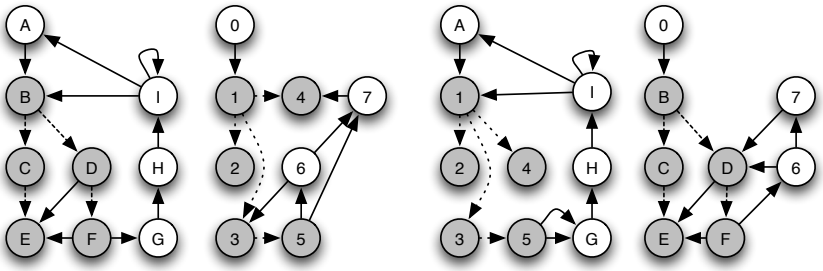

Fig. 3. On the left, the graphs of two parents. Considering $v_a = B$, $v_b = 1$ and $ra = rb = 2$, yields the subgraphs s_{ra} and s_{rb} whose nodes are depicted in grey. To establish the correspondence between nodes, the MSTs, here represented by dotted edges, are determined and Algorithm 1 applied. Considering that the algorithm returns the correspondence list {B-1, C-2, E-2, D-3, F-5, G-6, G-7, D-4}, the crossover operation results in the two descendants presented on the right.

Replace, Remove or Add symbol – when applied to a given production rule these operators: *replace* one of the present symbols by 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 nonterminal symbols.

Duplicate, Remove or Copy&Rename – 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 nonterminal; the copy&rename mutation will only affect the phenotype once the `Add symbol` mutation introduces a call to the new nonterminal in a production rule.

Change, Remove or Add parameter – as the name indicates these operators *add*, *remove* or *change* parameters and parameter values;

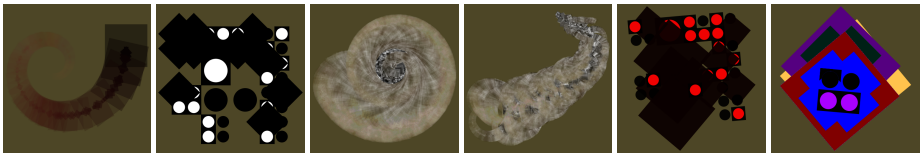

Fig. 4. The two leftmost images are the parents, the others are results of their crossover.

4 Experimentation

In a previous study [6] we showed the adequacy of the engine to evolve families of shapes when the initial population included hand-built grammars. In the current work we are interested in determining if the system is self-sufficient and if it can generate interesting and novel images and shape families without resorting to hand-built grammars. For this purpose we conducted experiments using automated-fitness assignment and user-guided evolution. Using an initial population of randomly created grammars could hide possible shortcoming of the genetic operators. As such, we chose to use as starting population a single individual whose genotype consists of a minimal grammar: a startshape directive and a single production rule composed of a call to the SQUARE terminal (the list of CFDG predefined terminals can be found in [2]).

We use a generational non-elitist approach and roulette wheel selection. In all experiments presented here: population size = 50, crossover probability = 0.7, maximum crossover radius = 3, mutation probability = 0.01 per individual (for each of the ten mutation operators). In the automated fitness runs the max number of generations = 100, while in the user-guided ones the stopping criteria were determined by the users.

4.1 Fitness Functions

In user-guided runs the fitness is supplied by the user who may assign to each individual a value in the $[0, 9]$ interval. For automated fitness runs we experimented three different fitness formulas:

RMSE – The fitness of an individual, a, is calculated by determining the root mean square error between its phenotype, $i(a)$, and a target image, t, as follows: $fitness(a) = 1/(1 + rmse(i(a), t))$

Fractal Dimension – Following Taylor et al. [9], among others, fitness is assigned by estimating the fractal dimension (FD) of $i(a)$ and comparing it with a target value, v, as follows: $fitness(a) = 1/(1 + (FD(i(a)) - v)^2)$. FD is estimated by the box counting method using the *chaos* library (`http://www.math.uic.edu/~culler/chaos/`). In the experiments presented here two arbitrarily chosen target v values, 1.3 and 0.8, are considered.

JPEG – We use JPEG compression to estimate image complexity [10]. Fitness is proportional to the file size of the JPEG encoding of $i(a)$: $fitness(a) = max(0, (size(jpeg(i(a)) - const)/1500)$. To increase the evolutionary pressure, a subtraction by a constant is performed ($const = 1000$ in all runs, a

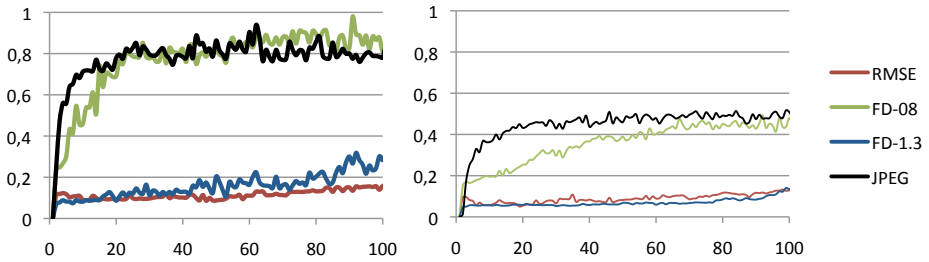

Fig. 5. Evolution of the best (left) and average (right) fitness throughout the automated fitness assignment runs. Results are averages of 10 runs.

value that was set taking into account the file size, 1100, of the image of the first population).

When the genotype encodes a non-deterministic grammar the phenotype may, and does, change from one interpretation to the other. As such, the fitness of an individual may vary from generation to generation. In other words, although an individual is a visual grammar, we assign fitness based on a single sample of this grammar. This design option poses an additional difficulty for the EC algorithm. Nevertheless, in theory, it should eventually converge to grammars predominantly composed of highly fit images.

4.2 Experimental Results

The analysis of the experimental results attained by EA systems typically implies a high degree of subjectivity. The main goals of the automated fitness runs were: (i) determine if the evolution of fitness values corresponds to the expectable behaviour of an EC engine; (ii) determine if the system is able to evolve complex grammars starting from a minimal one. In other words, the goals are validating the engine, the adequacy of the genetic operators, and test their ability to introduce novelty and promote complexification. The possible discovery of interesting shapes is not a goal.

The charts of Fig. 5 show the evolution of fitness throughout the automated runs. As can be observed, these charts display the typical behaviour of EC approaches. Fig. 6 presents some of the individuals evolved in these runs. Generally speaking, and although it is subjective to say it, the runs using RMSE fitness produced the least interesting results. This was expected and consistent with the results attained by researchers using RMSE fitness in expression-based EA (see, e.g., [11]). The individuals evolved using JPEG fitness created the most complex shapes, a result that was also expected. Interestingly, in several of these runs the system tended to evolve star-like shapes. This can be explained by two factors: (i) it is relatively easy to find a compact grammar that generates star-like shapes; (ii) the resulting images often result in a relatively large JPEG files.

The goal of the user-guided runs was to show that it was possible to generate diverse, interesting and novel shapes starting from "scratch". Typically the user guided runs had 20 to 40 generations, and the first half of the run was spent

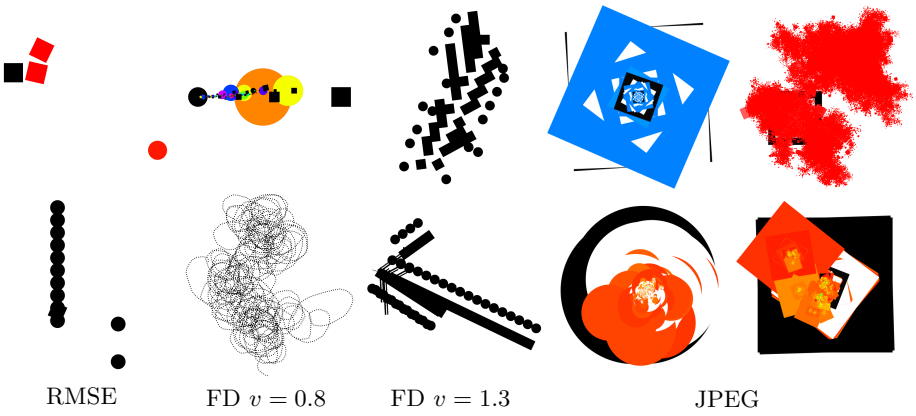

Fig. 6. Examples of images created in automated fitness runs.

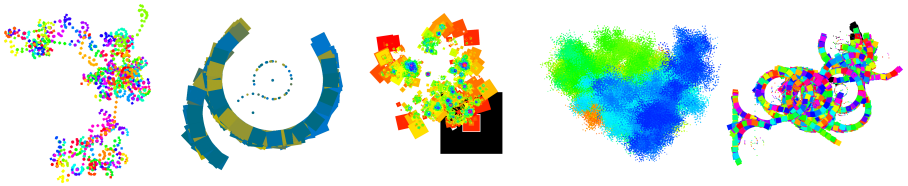

Fig. 7. Examples of images created in user-guided evolution runs.

on finding a grammar that draws more than a single shape. Thus, the use of a single and minimal grammar as first population makes the first generations of these runs quite boring for the user and, in normal circumstances, we would not recommend this initialization approach. Nevertheless, for the purposes of testing the system, we found it adequate. Fig. 7 presents examples of the images evolved in different user guided runs, showing what can typically be expected in these circumstances.

Although it would probably be advisable to increase the mutation probability during the first generations of these runs, the mutation operators proved valid, producing outcomes that are conceptually similar to the effects of mutation in expression-based EA. That is, the effects of mutation range from minor visual alterations to dramatic changes in appearance induced by small changes of the genetic code, with the later occurring less often [12].

In subsequent experiments, we used some of the individuals evolved in these runs as initial populations of other user-guided runs. Fig. 8 presents examples of images evolved in this fashion. The higher population diversity allowed us to get a better grasp of the behaviour of the crossover operator. In general terms, the outcome of the crossover operator appears to depend on the structural similarity of the genotypes and on their size. Additionally, when the parents are unrelated the visual appearance of each descendent tends to be mostly determined by one of the parents (see Fig. 4). An effect that is more visible with small genotypes.

As stated previously, non-deterministic grammars allow the definition of a family of shapes. The potential for evolving families of shapes instead of single

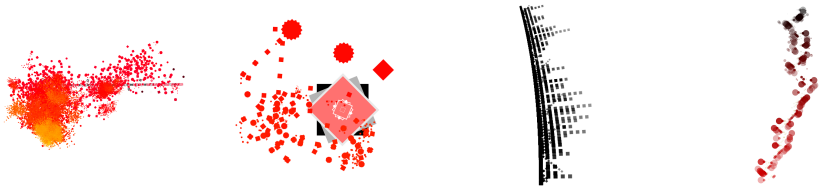

Fig. 8. Examples of images created in user-guided runs initiated with images from previous runs.

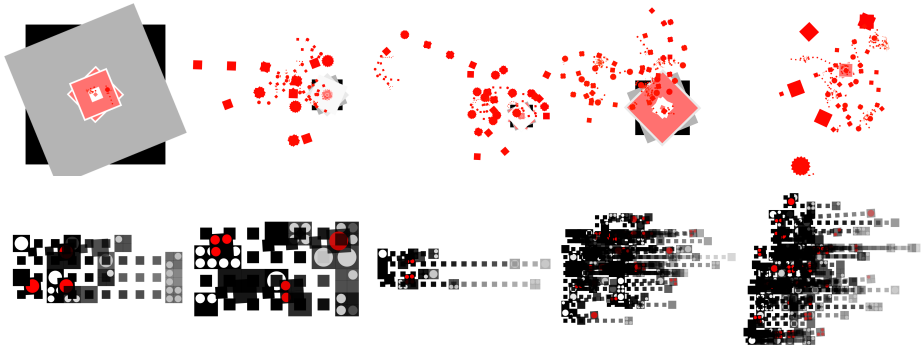

Fig. 9. Instances of the language of shapes defined by two individuals.

images is one of the main motivations for the use of CFDGs. However, in the runs presented here, while assessing the individuals the user only has access to one instance of the images an individual may generate. This means that the quality, diversity and consistency of the language of shapes encoded by the individual is not directly assessed. Nevertheless, and somewhat surprisingly, non-trivial and interesting families of shapes were still evolved. This is arguably explained by the following factors: (i) The experimental settings, namely starting conditions and genetic operators, naturally lead to non-deterministic grammars, which is confirmed by the analysis of the individuals generated in automated fitness runs; (ii) User Fatigue – the user eventually grows tired of individuals that systematically generate the same image, therefore the evolutionary process indirectly favors non-deterministic grammars; (iii) individuals that fail to reliably generate images valued by the user will eventually be discarded by evolution, in other words consistency is also favored. Fig. 9 presents instances of the visual languages defined by two of the evolved individuals.

5 Conclusions and Future Work

We presented a novel evolutionary engine that allows the evolution of CFDGs, is able to cope with non-deterministic grammars, and allows their recombination through a graph-based crossover operator. Due to these abilities, it successfully overcomes the limitations of previous EC approaches where CFDGs are used. When compared with typical expression-based and parametric evolution models, our approach presents several advantages, including the abilities: to evolve visual

languages instead of individual images; to use hand-coded grammars; and to allow the direct editing of the genotypes by the user.

Although the interpretation of the results is subjective, they provide evidence of the adequacy of the developed crossover and mutation operators. They also indicate that further experimentation is required to fully explore the potential of the approach for the creation of visual languages. Nevertheless, we consider this to be an important step in that direction.

In terms of future work, redesigning the user interface, exploring automatic image fitness assignment schemes, and developing approaches to automatically assess a language of shapes in terms of consistency, diversity and aesthetic qualities of the generated images are our top priorities.

Acknowledgments We would like to thank the anonymous reviewers for their thorough and insightful comments. This research is partially funded by the Spanish Ministry for Science and Technology, research project $TIN2008-06562/TIN$.

References

1. Stiny, G., Gips, J.: Shape grammars and the generative specification of paintings and sculpture. In Freiman, C.V., ed.: Information Processing 71, Amsterdam, North Holland Publishing Co. (1971) 1460–1465
2. Coyne, C.: Context free design grammar. http://www.chriscoyne.com/cfdg/ (last accessed in September 2009)
3. Lewis, M.: Evolutionary visual art and design. In Romero, J., Machado, P., eds.: The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music. Springer Berlin Heidelberg (2007) 3–37
4. Borrell, A.: Cfdg mutate. http://www.wickedbean.co.uk/cfdg/index.html (last accessed in September 2009)
5. Saunders, R., Grace, K.: Teaching evolutionary design systems by extending 'Context Free". In: EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing, Springer-Verlag (2009) 591–596
6. Machado, P., Nunes, H.: A step towards the evolution of visual languages. In: First International Conference on Computational Creativity, Lisbon, Portugal (2010)
7. Horigan, J., Lentczner, M.: Context Free. http://www.contextfreeart.org/ (last accessed in September 2009)
8. Pereira, F.B., Machado, P., Costa, E., Cardoso, A.: Graph based crossover — A case study with the busy beaver problem. In: Proceedings of the Genetic and Evolutionary Computation Conference. Volume 2., Orlando, Florida, USA, Morgan Kaufmann (13-17 July 1999) 1149–1155
9. Taylor, R.P., Micolich, A.P., Jonas, D.: Fractal analysis of Pollock's drip paintings. Nature **399** (June 1999) 422
10. Machado, P., Cardoso, A.: Computing aesthetics. In Oliveira, F., ed.: Proceedings of the XIVth Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence. Volume 1515 of LNCS., Porto Alegre, Brazil, Springer (1998) 219–229
11. Colton, S., Torres, P.: Evolving approximate image filters. In: EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing, Springer-Verlag (2009) 467–477
12. Machado, P., Cardoso, A.: All the truth about NEvAr. Applied Intelligence, Special Issue on Creative Systems **16**(2) (2002) 101–119