The Genetic Coding Style of Digital Organisms

Philip Gerlee, Torbjörn Lundh [1]

[1] Department of Mathematical Sciences, Chalmers University of Technology,
SE-412 96 Göteborg, Sweden
http://www.math.chalmers.se/~torbjrn/coding_style

Abstract. Recently, all the human genes were identified. But understanding the functions coded in the genes is a much harder problem. We are used to view DNA as some sort of a computer code, but there are striking differences. For example, by using entropy, it has been shown that the DNA code is much closer to random code than written text, which in turn is less ordered than ordinary computer code. Instead of saying that the DNA is badly written, using common programming standards, we might say that it is written in a different style − an evolutionary style. In this paper the coding style of creatures from the artificial life platform Avida has been studied. Avida creatures that have evolved under different size merit methods and mutation rates have been analysed using the notion of stylistic measures. The analysis has shown that the evolutionary coding style depends on the environment in which the code evolved, and that the choice of size merit method and mutation probabilities affect different stylistic properties of the genome. A better understanding of Avida's coding style, might eventually lead to insights of evolutionary codes in general.

1 Introduction

It was shown, using block entropy, in [1], that the DNA is much closer to random code than human written computer code. Furthermore a lot of examples have been found where genes have been reused for different purposes during development. As an example, take the **runt** gene in *Drosophila*, which has been shown [2] to be used in sex determination, segmentation and central nervous system creation. These findings suggest that the DNA is coded in a rather different fashion compared to human computer code.

C. Adami made in his survey talk in Stony Brook 10/27/98 on artificial life, a brief remark about the quality of the evolved program codes of the digital organisms [3] in the artificial life platform Avida [4]: ``*The codes that are evolved will eventually be almost totally unreadable. Things are never used only once, but two or more times. It is a kind of a 'madman's' code.*''

How can we capture these comments about the style, or quality, of the computer code and the DNA, in a quantified manner? Can we do that in such a general manner that we will be able to use analogous quality measure both for carbon − and silicon based genetic codes?

When a population evolves and adapts to an environment, information about what traits or chemical reactions that are beneficial in that environment are written into the genome. The environment of course influences *what* information is coded into the genome, but what we are interested in is to investigate *how* the information is coded. In this paper we will investigate the different coding styles that the environment enforces on the genomes by examining Avida creatures that have evolved under different size merit methods and mutation rates. This will be done using the notion of a stylistic measure that was introduced in [5].

2 Codes

One common way to view functions is as black boxes. Here we are interested of the internal structure of such black boxes performing equivalent tasks. Let us give a simple example of two codes that interprets the same simple real valued function:

$$f(x) = \frac{1}{x-2} \qquad g(x) = \begin{cases} \dfrac{x+2}{x^2-4} & \text{if } x \neq -2 \\ -\dfrac{1}{4} & \text{if } x = -2. \end{cases} \tag{1}$$

The genome of a creature can be thought of as a code that is written in an alphabet consisting of a finite numbers of letters, which is read or interpreted by the CPU. The genomes found in nature are written with the four letters A, T, G and C, which corresponds to the base pairs in the DNA. In Avida the genomes consists of a combination of 24 different CPU instructions, which alter the state of the virtual CPU in some manner.

We will therefore define a **code** to be a finite string of letters taken from a finite alphabet A, such that when the code is interpreted, the code will represent a well-defined function or process, f.

$$Code = \{\alpha_i\}_{i=1}^{k}, \quad \text{where } \alpha_i \in A. \tag{2}$$

From the above definition it is clear that different codes can have the same function representation. Let us therefore define a class of codes C_f to be the set of all codes that perform the function f when they are interpreted. From a genetic point of view one can interpret the function of the code as the phenotype of a creature, and thus the code classes as classes of phenotypically equivalent creatures.

In Avida the interpretation is performed by running a creature through the virtual CPU for one generation. If a successful divide occurs the function that the code defines is simply the tasks that the creature performs during one generation.

3 Styles of a Code

If we look at two codes from the same class we know that they perform the same function, i.e. they have the same phenotype, but their genotype may differ. As the evolutionary process is very sensitive to perturbations we cannot expect to find the same genotype if evolution occurred two times in the same environment. It's therefore interesting to study the style of the code, as this is more likely to be invariant between two instances of the same process. We would therefore like to define the style of a code. This is done by introducing measures (3) on the code class C_f, that maps each code to a point in [0,1].

$$\mu_i : C_f \to [0,1]. \tag{3}$$

Putting together several of these measures we can create a profile measure $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3, \ldots)$, which might serve as a 'fingerprint' of the code.

4 Measures

As we are interested in distinguishing between different coding styles of creatures from Avida we have constructed four different measures that measure different properties of the genome. Three of the measures are calculated from the functional genomic array (FGA) of the genome, a representation which reveals the genetic structure and the localisation of genes [6]. The FGA is a $N \times M$ binary matrix, where N is length of the genome and M is the number of tasks the creature manages including replication. The entry at position (i,j) is 1 if instruction number i is involved in the calculation of task number j and 0 otherwise.

4.2 Gene Correlation

This measure shows how correlated different genes are and is constructed in the following manner. Sum the functional genomic array along the rows, and remove all zero entries in the resulting vector. Sum up all entries that are larger than one and divide by the number of tasks plus one (for replication) and by the number of non-zero entries in the vector.

If we let $\mathbf{T}$ be the row sum of the FGA, N be the number of tasks the creature manages plus one (for replication), L_E, the number of non-zero entries in $\mathbf{T}$ (the number of essential instructions), then the gene correlation is given by (4), where the sum runs over all indices for which $\mathbf{T_i} > 1$.

$$g_c = \frac{\sum_i \mathbf{T}_i}{NL_E}. \tag{4}$$

The gene correlation is a number between 0 and 1, where 0 means that all tasks including replication are coded disjoint in the genome, and 1 means that they all de-

pend on the same instructions. Note that this measure also can be interpreted as the compression of information in the genome.

4.3 Redundancy

This measure gives the fraction of instructions that don't affect the tasks and replication of a creature. As above the FGA is summed along the rows, but this time we count how many zero entries the vector holds, this number is then normalised by the length of the creature.

4.4 Introns

This measure simply gives the fraction of instructions that never are executed when the creature is executed for one life-cycle. Under the default size merit method 4, the fraction of introns is rather low (see table 1), as they are punished. Note that the introns are a subset of the redundant instructions.

4.5 Fragility

This measure gives the number of instructions that are essential for replication, and is constructed from the FGA by simply summing the replication column. This measure isn't normalised with the length of the creature because the number of instructions needed to replicate are independent of the length of the creature, as the creatures use copy loops. But as we want a measure to lie in [0,1] this measure is normalised using the map (5).

$$f(x) = \frac{x}{x+c}, \quad c > 0 \tag{5}$$

From preliminary data we know that the number of instructions essential for replication in most cases lie between 5 and 50. The c in (5) is therefore optimised so that the image of the interval $f([5,50])$ is maximised. The optimisation is straight forward and gives the value $c = \sqrt{(50 \cdot 5)} \approx 15.8$, which gives $f(50) \approx 0.76$ and $f(5) \approx 0.24$. In an environment that doesn't reward computational efforts it is only the fragility that is minimised [7], which results in a minimised genome length.

5 Comparison of Different Styles

5.1 Size Merit Methods

The different size merit methods in Avida generate qualitatively different styles of coding, for example size merit method 1, which gives merit proportional to copied size, tends to produce introns and does not put any pressure on the lengths of the creatures, while size merit method 0 does quite the opposite, because merit is independent of size. The third size merit method, 4, takes the minimum of copied and executed size as merit.

The merit is a very important property as it is used in the calculation of the fitness. This implies that the size merit method has a direct impact on how selection is performed in Avida [4].

The question is if these differences in merit calculation can show in a stylistic analysis of the different methods. To investigate this we created three sets of creatures each containing approximately 60 creatures, from the three size merit methods 0, 1, 4, the full settings for these runs can be found in the appendix.

The optimal comparison would be to compare creatures from the same code class (phenotype), but as evolution in Avida proceeds with different pace each run one has no guarantee that a certain phenotype has evolved after a fixed number of updates. One way to accomplish this would be to reward only those functions that we want the phenotype to perform and use a large number of fixed updates, but this approach also has its drawbacks. If the required phenotype appears early in evolution, then the code is optimised during the rest of the run as no new functions are rewarded. If a creature from the above run is compared to a creature from a run where the required phenotype appeared just before the maximal number of updates their coding styles would certainly differ, as one creature has optimised its coding while the other has not. Instead we decided to compare creatures with approximately the same complexity. This was done by extracting a creature from the dominant genotype after 40 000 updates in each run and if it had reached a certain degree of complexity (it managed at least three boolean functions) it was kept for analysis.

Table 1 shows the average and standard deviation for each measure under the three size merit methods. Another way to analyse the data is to perform a principal component analysis on the data, which reduces the dimensionality to 2, and gives a better graphical representation. The result of the PCA can be seen in fig. 2. The vectors that span the plane in fig. 1, are the 1st and 2nd principal components (6).

Table 1. The average and standard deviation of the stylistic measures for creatures from three different size merit method.

Size-Merit Method	Gene correlation	Redundancy	Introns	Fragility

0	0.2655 (0.1221)	0.1666 (0.1102)	0.0249 (0.0496)	0.4860 (0.0402)
1	0.2647 (0.1161)	0.6443 (0.1836)	0.3581 (0.1767)	0.5639 (0.0706)
4	0.2288 (0.0990)	0.5230 (0.1905)	0.0708 (0.0911)	0.5565 (0.0705)

$$P_1 = \begin{pmatrix} -0.0132 & -0.6263 & -0.5723 & 0.1700 & -0.5011 \end{pmatrix} \quad (6)$$
$$P_2 = \begin{pmatrix} 0.7861 & -0.1275 & 0.0156 & -0.5990 & -0.0825 \end{pmatrix}$$

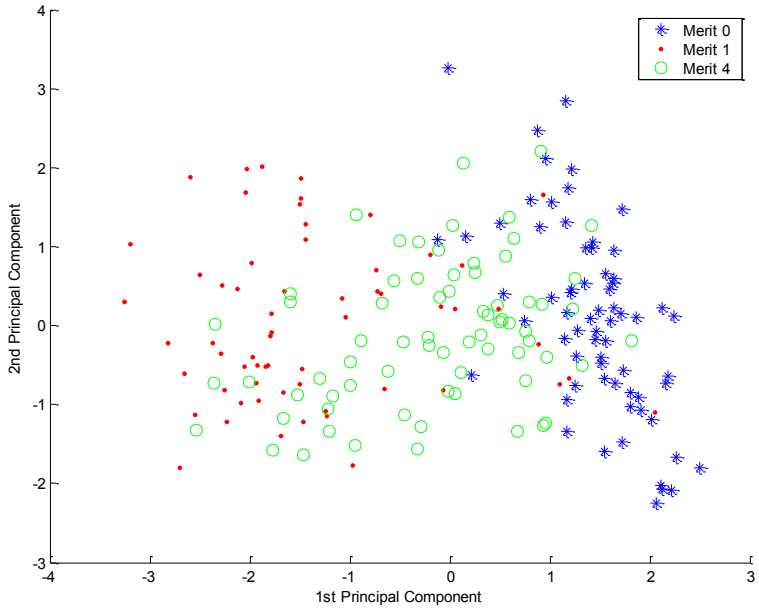

Fig. 1. Principal component analysis of the profile measure of creatures that have evolved under different size merit methods.

5.2 Mutation Rates

The mutation rates in Avida play an important role in the adaptive process. If the mutation rates are low evolution tends to proceed very slow as the fitness landscape is explored at a low pace and if they are high the population will have trouble sustaining information in the genomes [8,9].

To investigate how the mutation rates influence the coding style we created three sets of creatures that had evolved under different copy mutation probabilities each

containing approximately 60 creatures. The point mutation rates were set to zero, the insert/delete mutation probabilities were kept constant at 0.05 per divide and the size merit method was set to 4 (the default value in Avida). The copy mutation probability was set to one low value (p_c = 0.001), one intermediate value (p_c = 0.005) and one high value (p_c = 0.025).

As in the experiment above the populations evolved for 40 000 updates after which the dominant genotype was extracted, but it was only kept for analysis if it managed three or more boolean functions. A detailed description of the settings can be found in the appendix. The three sets of creatures where then analysed using the stylistic measures and the results can be found in table 2 and a PCA-plot in fig. 2. The 1st and 2nd principal components are given by (7).

Table 2. The average and standard deviation of the stylistic measures for three sets of creatures that have evolved under different copy mutation rates

p_c	Gene correlation	Redundancy	Introns	Fragility
0.001	0.2263 (0.1093)	0.6076 (0.2006)	0.0487 (0.0883)	0.5877 (0.0811)
0.005	0.2288 (0.0990)	0.5230 (0.1905)	0.0708 (0.0911)	0.5565 (0.0705)
0.025	0.3037 (0.1372)	0.2847 (0.1558)	0.0372 (0.0985)	0.4872 (0.0465)

$$P_1 = \begin{pmatrix} -0.2087 & 0.6374 & 0.4462 & 0.2359 & 0.5435 \end{pmatrix}$$
$$P_2 = \begin{pmatrix} -0.7657 & -0.0693 & -0.5643 & 0.2695 & 0.1337 \end{pmatrix} \quad (7)$$

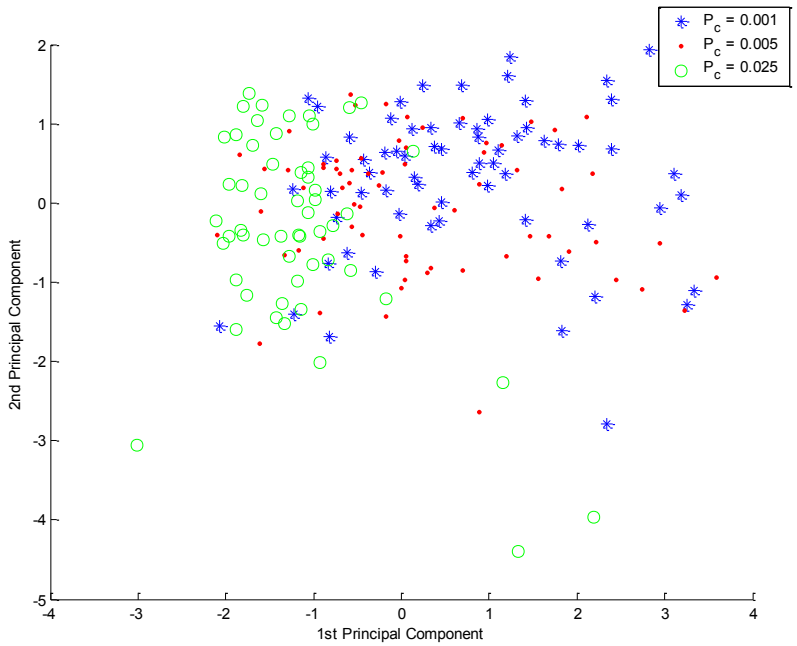

Fig. 2. Principal component analysis of the profile measure of creatures that have evolved under different copy mutation probabilities.

6 Discussion

6.1 Size Merit Methods

In fig. 2 one can see a separation between the different size merit methods, which clearly indicates that there are differences in their coding style enforced by the size merit method. One way of analysing these differences is to look at the composition of the 1st principle component (6). The measures with the larger weights show where the styles differs the most. This shows that the coding styles differ most with respect to the redundancy, intron and fragility measures. What can also be seen in the PCA plot is that the coding style within each size merit methods varies quite much, a thing that also can be seen in the standard deviations in table 1. The reason for this is that the evolutionary process in Avida takes different paths each run due to the randomness in the process, and thus giving rise to a unique coding each run. The large standard deviations makes it impossible to draw any conclusions about how the gene correlation is affected by the size merit method, but the other measures show separation between the different size merit methods.

The results of this experiment are intuitive and can be explained directly from how the merit is calculated. Size merit method 1 for example has a high fraction of introns as the method gives merit proportional to copied size and therefore does not punish introns and method 0, which gives merit independent of size, gives a efficient coding with a low fraction of both introns and redundant instructions.

6.2 Mutation Rates

The principal component analysis plot (fig. 2) in this case does not show the same separation between the coding styles as in the case with size merit method. The coding style of the creatures from the low and intermediate mutation probabilities seem clustered together, but the coding style of the high mutation probability show at least some separation from the other two. The 1st principal component (7) shows that the largest differences between the coding styles lie in the redundancy, fragility and gene correlation measures.

The averages of the stylistic measures (table 2) shows the same tendency as the PCA plot, the low and intermediate mutation probabilities give approximately the same values while the high probability differs in the gene correlation, redundancy and fragility measures.

The fragility decreases when the mutation probability increases. The reason for this is that a high mutation probability requires a more efficient coding of the self-replication. A creature that uses too many instructions for self-replication would be less likely to produce a viable offspring when the copy mutation probability is high.

The gene correlation on the other hand increases when the mutation probability increases. The high mutation probability forces the creatures to compress the information in the genome, in order to make it less likely to be struck by a deleterious mutation. This compression corresponds to that the genes share instructions which gives a higher gene correlation.

The reason why the redundancy decreases when the mutation probability increases is because a redundant instruction that is copied incorrectly may alter the execution in the offspring which may lead to the loss of a gene or even the capability to self-replicate. It is therefore disadvantageous to have a high redundancy when the mutation probability is high.

7 Conclusion

The results of the experiments clearly show that settings in Avida produce different coding styles. Most of the differences that appear are intuitive and can be explained directly from for example how the merit is calculated. While some results, like the change in gene correlation in the experiments with mutation probabilities, requires an understanding of the evolutionary process in Avida.

What these experiments show is that different environmental settings affect different stylistic properties of the genome. The gene correlation does not seem to depend much on the size merit method but rather on the mutation probabilities and the frac-

tion of introns seems independent of the mutation probabilities but depends strongly on the size merit method. The fragility measure on the other hand seems to depend on both mutation probabilities and size merit method. But the main result is that we can distinguish between different coding styles from different environments using a stylistic profile measure. It would be very interesting if one could study other evolutionary driven systems using this stylistic approach in order to look for universal features of evolutionary driven code in general and DNA in particular.

References

1. Schmitt, A.O., Herzel H.: Estimating the entropy of DNA sequences, Journal of Theoretical Biology, 188: 3, (1997), 369-377
2. Duffy, J., Gergen, P.: Sex, Segments, and the Central Nervous System: Common genetic mechanisms of cell fate determination, Adv. in Genetics, Vol. 31, (1994) 1-28
3. Wilke, C.O., Adami, C.: The biology of digital organisms. Trends. Ecol. Evol., 17, (2002), 528-532
4. Ofria C., Wilke, C.O.: Avida: A Software Platform for Research in Computational Evolutionary Biology. Artificial Life, 10, (2004), 191-229
5. Lundh, T.: In search of an evolutionary coding style, SUNY Stony Brook IMS preprint 3, (2000)
6. Lenski, R.E., Ofria C., Penncock, R.T., Adami, C.: The evolutionary origin of complex features, Nature 423, (2003) , 139 - 144
7. Lenski, R.E., Ofria C., Collier, T.C., Adami, C.: Genome complexity, robustness and gene interactions in digital organisms, Nature 400, (1999), 661 – 664
8. Adami, C., Edlund, J.A.: Evolution of robustness in digital organisms. Artificial Life 10, (2004), 167-179
9. Wilke, C.O, Wang, J.L., Ofria, C., Lenski, R.E., Adami, C.: Evolution of digital organisms at high mutation rate leads to survival of the flattest. Nature, 412, (2001), 331-333

Appendix

All experiments for this paper were performed with Avida 1.3.0 for Windows. The settings used for the experiments were the default settings in Avida, except for the changes in size merit method and copy mutation rates. The task bonuses were set to the default value except for the fact that the rewards for 3-input boolean functions were removed. In the runs with size merit 0 the task bonuses were raised slightly in order to prevent the evolution from going in to a size minimising state. Each run was started with a time based random seed and the ancestor used in all experiments was `creature.base`, which is supplied with Avida. The ancestor, `genesis` and `task_set` files for all experiments can be downloaded from http://www.math.chalmers.se/~torbjrn/coding_style.