Developmental Mappings and Phenotypic Complexity

Per Kristian Lehre and Pauline C. Haddow
Department of Computer and Information Science
Norwegian University of Science and Technology
Sem Sælands vei 7-9, NO-7491 Trondheim, Norway
{lehre,pauline}@idi.ntnu.no

Abstract- The effect of phenotypic complexity on distance correlation plots is investigated for two developmental mappings, a mapping based on L-systems, and a 2D cellular automata mapping. Our treatment of complexity is based on the theory of Kolmogorov complexity. A new genotype sampling algorithm called Crossection Walk is introduced.

1 Introduction

Multicellular organisms develop from single-celled zygotes to grown-up organisms in a developmental process implicitly defined by their DNAs. Biological development thus consists of a small genotype (relative to the phenotype) and a development process (implicitly defined by the genome) which enables the genotype to develop to the phenotype. The scalability inherent in biological development has attracted the interest of a number of researchers in the field of EC and particularly the field of evolvable hardware (EHW).

Several attempts have been made to construct artificial equivalents of a developmental process. We will refer to such approaches as *artificial development*, but they are also named ontogeny, embryogenesis or computational development. This field can be considered as a subfield of evolutionary computation, and is fairly new.

At least two main directions can be found in this subfield. One is to make biologically plausible models of development in order to learn about development in nature. The other is to make indirect genotype-phenotype mappings inspired by biological development.

Early work includes Kitano [Kit90] who used a matrix-rewriting grammar (based on L-systems [Lin68]) as implicit genotype-phenotype mappings when evolving artificial neural network (ANN) structures . Later, Gruau proposed a graph rewriting grammar called cellular encoding [Gru94], also for evolving ANN structures. Boers and Sprinkler also employed graph grammars [BSK01] in the same application domain. Other research includes [SL98, GB02, vRCD$^+$03, TH03].

The field of artificial development is an expanding field. Empirical results, although promising, may be said to be inconclusive with respect to the scalability of artificial development. This work takes a step toward answering the scalability question by considering phenotypic complexity i.e. the complexity of the solutions achievable through the application of artificial development. Our notion of complexity is based on the theory of Kolmogorov complexity [LV97]. Our goal is to see if phenotypic complexity has any effect on distance correlation plots where these plots illustrate the distance preservation between pairs of genotypes and their corresponding phenotypes given developmental mapping. Experiments are based on two existing development algorithms: Kitano's matrix rewriting grammar [Kit90] and 2D cellular automata.

The rest of the paper is organized as follows: In Section 2, Kolmogorov complexity is briefly introduced, the plots are described, and the Crossection Walk sample algorithm is introduced. Details of the experimental setup are described in Section 3, and the following Section 4 contains the distance correlation plots and an interpretation of these result. Finally, in Section 5, the results are set in a larger context, along with some advices on further research.

2 Theory

A *developmental mapping* may be represented by a function $\phi : \mathcal{G} \rightarrow \mathcal{P}$ where the set $\mathcal{G}$ is called the *genospace* and is equipped with a distance metric $d_{\mathcal{G}}$, and correspondingly, the set $\mathcal{P}$ is called the *phenospace* and is equipped with a distance metric $d_{\mathcal{P}}$. Elements of the genospace $\mathcal{G}$ are called genotypes, and elements of the phenospace $\mathcal{P}$ are called phenotypes. Given two genotypes x and y, or two phenotypes $\phi(x)$ and $\phi(y)$, we say that their *genotypic distance* is $d_{\mathcal{G}}(x, y)$ and their *phenotypic distance* is $d_{\mathcal{P}}(\phi(x), \phi(y))$. The overall situation is illustrated in Figure 1.

Our objective is to characterize how the developmental mapping ϕ preserves distances from $\mathcal{G}$ into $\mathcal{P}$ i.e. how well the genotypic distances correlate to corresponding phenotypic distances. And such correlations are studied with *distance correlation plots* where genotypic distances are plotted against phenotypic distances.

If we assume that the correlation is invariant over the whole genospace - as is often done - we could obtain the distance correlation plots simply by plotting genotypic distance against phenotypic distance for randomly sampled pairs of genotypes.

But, as we will see, the assumption that distance-preservation is invariant does not always hold. A develop-

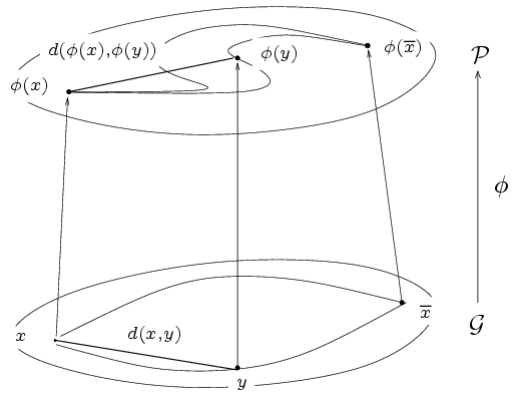

Figure 1: Artificial Development Mapping from Genospace to Phenospace

mental mapping may have regions where distances are very well preserved, and other regions where distances are hardly preserved at all.

So we will characterize distance-preservation by sampling different regions of each developmental mapping. One distance correlation plot is made for each region.

Subsection 2.1 briefly introduces the complexity measure and how it is approximated. Subsections 2.2 and 2.3 contain descriptions of the distance correlation plots along with the sampling algorithm Crossection Walk.

2.1 Kolmogorov Complexity

What are complex bitstrings? Take the examples below.

$$x = 10101010101010101010$$
$$y = 11010101011011101010$$

The bitstring x appears less complex than the bitstring y. The bistring x is simply ten repetitions of 10. Increasing the length of bitstring x by adding more repetitions of 10 does intuitively not make x any more complex. Conversely it is more difficult to see any pattern in bitstring y.[1]

Bitstring x is simple because we can make a short description of the string. Bitstring y appears more complex since it it cannot be easily described in a few words.

Kolmogorov associated the complexity of an object with the length of the shortest description of the object. To have an unambiguous measure of an object's description length, the description length was defined as the length of the shortest program that generates the object on a fixed, universal Turing machine.

The Kolmogorov complexity measure is incomputable, so for practical experiments, we need an approximation. Compression is used as an upper approximation of the Kolmogorov complexity. Here, we use the Lempel-Ziv algorithm to compress the binary phenotype. Bitstrings that are easily compressible have low complexity, while bitstrings that cannot be compressed are complex. So the compression ratio is proportional to the complexity.

This way of measuring complexity has both theoretical and practical advantages. It is based on well-founded theory, and the Lempel-Ziv approximation can easily be computed using the standard compression library `zlib` which is easily called with a few lines of code.

2.2 Genotype-Phenotype Distance Correlation Plots

Intuitively, distance correlation plots estimate how well distances are preserved in the genotype-phenotype mapping. Each plot is based on a single genotype x_0 in $mathcalG$. We will call this genotype x_0 the *origin genotype of the plot*. Given a sampled subset G of the genospace $\mathcal{G}$, we plot the distance between the genotypes $d_{\mathcal{G}}(x_0, x_i)$ against the distance of the corresponding phenotypes $d_{\mathcal{P}}(\phi(x_0), \phi(x_i))$ for each element x_i in G. The result is a scatter plot which indicates the correlation between genotypic distance and phenotypic distance relative to the origin genotype of the plot.

The functions $d_{\mathcal{G}}(\cdot,\cdot)$ and $d_{\mathcal{P}}(\cdot,\cdot)$ are distance metrics over the genospace and phenospace respectively. In this paper, we only work with bitstrings, so we use the Hamming distance as metric, and we will simply denote it by $d(\cdot,\cdot)$.

As an example, the simplest possible correlation plot - a sharp diagonal - would be obtained from a direct encoding map. At the other extreme, we can obtain a very different correlation plot from a mapping where the genotype is the seed to a random number generator, and the phenotype is the pseudo-random number generated from the seed. With a good random number generator, the plot will look like uniformly distributed noise in the correlation plot.

2.3 Genotype sampling with Crossection walks

Our proposed sampling method is called *Crossection Walk* and is summarized in Algorithm 1.[2]

In the following, if the symbol x denotes a bitstring, then the symbol $\overline{x}$ denotes the complementary bitstring, and $x(k)$ is the kth bit of bitstring x. So, if the bitstring $x = 1100$, then the complementary bitstring $\overline{x} = 0011$ and the bit $x(2) = 1$. Again, we will let $d(\cdot,\cdot)$ denote the Hamming distance on bitstrings. So, to continue our example, we have $d(x,\overline{x}) = 4$ when $x = 1100$.

[1] For the curious reader, this bitstring is related to the words phenotypic complexity.

[2] Note to referees: Due to a technical problem with latex, the algorithm was not allowed inside a figure environment. This will be fixed in the camera-ready version.

The purpose of the algorithm, is to sample random elements from $\mathcal{G}$ where the genotype distances to x_0 are uniformly distributed along the entire interval $[0, d(x_0, \overline{x_0})]$. Intuitively, the method is a "random walk" from x to the complementary binary string $\overline{x}$. In each step, we move one unit distance from the origin x, and one unit step closer to the complement $\overline{x}$. The price to pay for uniform distance relative to x_0, is a limited distance between elements of consecutive walks.

Algorithm 1: Crossection Walk

Data : A bitstring x_0 of length ℓ

Result : A set of $\ell + 1$ bitstrings of length ℓ

begin

$y_0 \leftarrow x_0$

$J \leftarrow \{1, 2, ..., \ell\}$

$\sigma \leftarrow$ a random permutation of J

for $i \leftarrow 1$ *to* ℓ **do**

$$y_i(k) \leftarrow \begin{cases} \overline{y_{i-1}}(k) & \text{when} \quad k = \sigma(i) \\ y_{i-1}(k) & \text{elsewhere} \end{cases}$$

return $\{y_0, y_1, ..., y_\ell\}$

end

Example: Two possible results of calling the algorithm with input 1111 is the sequences of bitstrings

$$\begin{aligned} A &= (1111, 1011, 1010, 0010, 0000), \text{ and} \\ B &= (1111, 0111, 0101, 0001, 0000). \end{aligned}$$

3 Experiments

The following experimental procedure was carried out on both developmental mappings. Let $\phi : \mathcal{G} \rightarrow \mathcal{P}$ denote any of the two mappings.

First, a subset of the genospace $G \subseteq \mathcal{G}$ was randomly sampled. Then, for each element x in G, the complexity of the corresponding phenotype $\phi(x)$ was approximated with Lempel-Ziv compression. The elements in G were then ordered according to complexity.

Then, for each genotype $x \in G$, the Crossection Walk Algorithm was called 100 times on input x. Denote the resulting set of genotypes G_x. Then, a distance correlation plot was computed based on this set G_x. Hence, the number of spots in each plot is 100 times the distance $d(x, \overline{x})$.

The result is a set of distance correlation plots with different origin. We say that the plots were sampled from different regions of the landscape.

The next subsection contains details about the artificial development mappings used in the experiments.

3.1 The two Developmental Mappings

We use two different artificial development mappings in the experiment: a slightly modified version of Kitano's original development mapping, and 2D cellular automata.

The cellular automata used have two states, and are run on a finite 2D grid of size 128×128 with a Moore- neighborhood (i.e. 9 neighbors). A genotype represents one of the 2^{9^2} rule sets, i.e. the genotype size is 512 bits. The automata were iterated for 100 steps from an initial position where all cells were in state 0 except for a single cell in the center with state 1.

We used a slightly modified version of Kitano's development system [Kit90]. In his original scheme, after the desired number of iterations, the matrix consisted of symbols and/or 1s and 0s. All symbols were replaced by 0s to obtain a final bit matrix. This choice put a bias toward 0s, which was perfectly sensible in Kitano's application where the bits were adjacency matrix descriptions of neural network structures.

In our experiments, we want to balance the number of 1s and 0s. We therefor introduce a new set of rewrite rules, one for each upper-case symbol. Each of these rules is on the form $Symbol \rightarrow \{0, 1\}$. In the modified scheme, possible symbols in the final matrix are replaced using the above mentioned rules. Lower-case symbols are replaced by 0 as in the original algorithm.

The grammar is encoded using a constant-size binary string. The encoding is divided into 17 blocks, one for each of the upper-case symbols $S, A, B, ..., P$. Each block is divided into five pieces. The first piece is of one bit and represent the final rewriting of a symbol in the last iteration. The other four pieces, each of five bits, designate the rewriting symbols. Each of the 32 symbols $A, B, ..., P, a, b, .., p$ is given a 5 bit unique identifier. A is 00000, B is 00001, a is 10000 and so on. An encoding of a grammar thus uses $17 \cdot (1 + 4 \cdot 5) = 357$ bits. An example is shown below:

$$\overbrace{1\ \underbrace{00000}_{A}\ \underbrace{10000}_{a}\ \underbrace{10001}_{b}\ \underbrace{00001}_{B}}^{S}, \qquad S \rightarrow \begin{matrix} A & a \\ b & B \end{matrix}, \quad S \rightarrow 1$$

$$\overbrace{0\ \underbrace{01111}_{P}\ \underbrace{11111}_{p}\ \underbrace{10001}_{b}\ \underbrace{00000}_{A}}^{A}, \qquad A \rightarrow \begin{matrix} P & p \\ b & A \end{matrix}, \quad A \rightarrow 0$$

$$\vdots \qquad\qquad \vdots$$

$$\overbrace{1\ \underbrace{10001}_{b}\ \underbrace{00001}_{B}\ \underbrace{00001}_{B}\ \underbrace{01111}_{P}}^{P}, \qquad P \rightarrow \begin{matrix} b & B \\ B & P \end{matrix}, \quad P \rightarrow 1$$

4 Results and Discussion

Each of the plots in Figures 2[3] and 3 is a distance correlation plot where the x-axis represents genotypic distances and the

[3]Note to referees: The y-axis in the plots are incorrectly labeled "Fitness difference from optimal genome". This will be changed to "Phenotype

y-axis represents phenotypic distances to the origin genome of the plot. The plots give a picture of the correlation between the two distances.

The plots are ordered with increasing phenotypic complexity. In the upper plot, the origin genotype has a phenotype with low complexity. In the bottom plot, the origin genotype has a phenotype with high complexity.

For convenience, we will denote plots where the origin genotype has a phenotype with low complexity as *low complexity plots*, and we will denote plots where the origin genotype has a phenotype with high complexity as *high complexity plots*.

In the upper plot in Figure 2, we have a low complexity plot. As described in Section 2.2, the first plot resembles nearly a direct encoding mapping from genotype to phenotype. The variance of the plot is relatively high.

As we continue downward through the plots in Figure 2, a curve starts to appear. This curve introduces a certain threshold which is indicated with an arrow labeled 1 in the bottom plot. Above this threshold, the plot is almost horizontal. Below the threshold, the plot is almost linear.

As we move downward toward the high-complexity plots, the variance reduces.

And in Figure 3, the distance correlation plots for the CA mapping are shown. Beginning at the top, the plot has a relatively high variance. But the plot is not linear as was the case for the low-complexity plot in Figure 2. There seems to be a threshold as was the case in the high-complexity plot in Figure 2.

When moving downwards to the high complexity plots, the variances in the plots reduce significiantly. The bottom plot appears as a thin, horizontal stripe.

For both mappings, high complexity plots appear smoother than the low complexity plots.

Similar effects of phenotypic complexity were observed in the plots not shown here. The effect was most easily observed on the matrix rewriting grammar mapping.

5 Conclusion

We have shown that phenotypic complexity when measured with Kolmogorov complexity, has a clear, and strong impact on distance correlation plots for a matrix-rewriting grammar mapping and a 2D CA mapping.

High phenotypic complexity appears to have a negative effect on the distance correlation plots, while very low phenotypic complexity gives plots which resembles that of direct encoding.

The fact that we observed a very similar effect with two relatively different developmental mappings strengthens our result. However, we cannot claim that all developmental

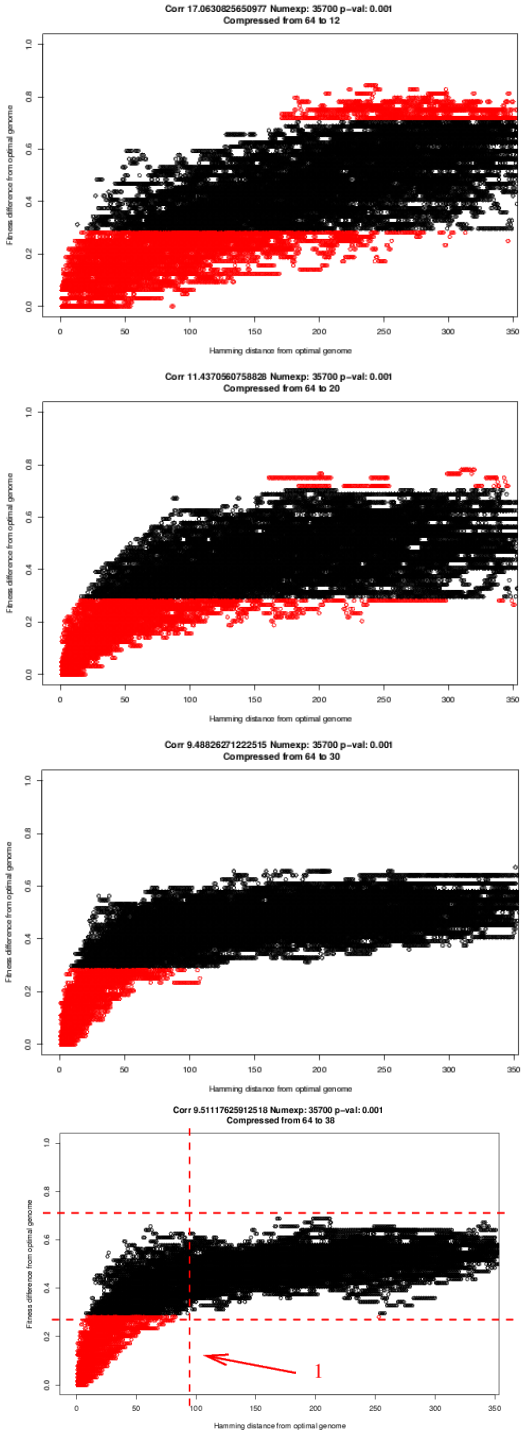

Figure 2: Four distance correlation plots from the matrix rewriting grammar mapping.

distance to origin genotype of the plot".

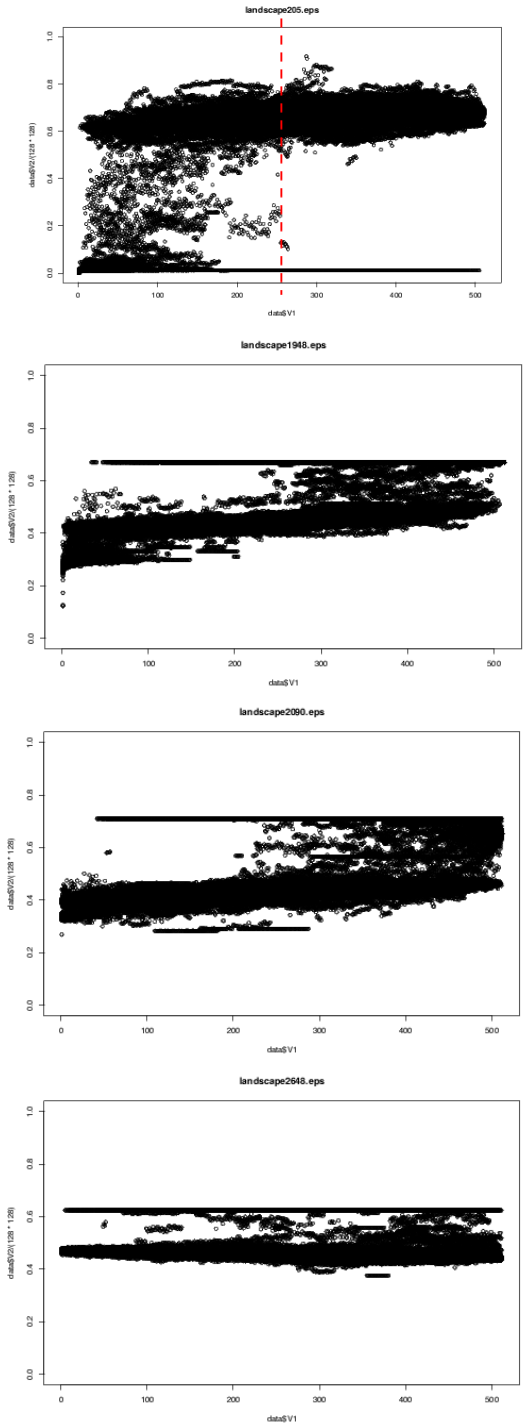

Figure 3: Four distance correlation plots from the CA mapping.

mappings will behave similarly. Indeed, we hope that there may be some mappings that can cope with phenotypic complexity.

A new sampling algorithm called Crossection Walk was introduced to sample genotypes where the distance to a specific genotype is uniformly distributed.

The method used to produce the distance correlation plots is application-independent and relatively simple. Researchers proposing new developmental mapping can easily carry out the same procedure as long as the corresponding genospace and phenospace allow computable metrics.

5.1 Future Work

In order to make our results more general, the method should be carried out on other developmental mappings.

Furthermore, any attempt at a theoretical explanation of the experimental results would be interesting.

The effect of phenotypic complexity on the performance of an evolutionary algorithm using artificial development was not discussed in this paper. This could possibly be investigated on a set of test problems.

Developmental mappings could be investigated with a similar procedure on other phenotypic properties than complexity.

Bibliography

[BSK01] E.J.W. Boers and I.G. Sprinkhuizen-Kuyper. Combined biological metaphors. In M.J. Patel, V. Honavar, and K. Balakrishnan, editors, *Advances in the Evolutionary Synthesis of Intelligent Agents*, A Bradford Book, chapter 6, pages 153–183. MIT Press, Cambridge, Massachusetts, 2001.

[GB02] T.G.W. Gordon and P.J. Bentley. Towards development in evolvable hardware. In Adrian Stoica, Jason Lohn, Rich Katz, Didier Keymeulen, and Ricardo Salem Zebulum, editors, *The 2002 NASA/DoD Conference on Evolvable Hardware*, Alexandria, Virginia, 15-18 July 2002. Jet Propulsion Laboratory, California Institute of Technology, IEEE Computer Society.

[Gru94] F. Gruau. *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm.* PhD thesis, France, 1994.

[Kit90] H. Kitano. Designing neural networks using genetic algorithm with graph generation system. *Complex Systems*, 4:461–476, 1990.

[Lin68] Aristid Lindenmayer. Mathematical models for cellular interactions in development, I & II. *Journal of Theoretical Biology*, 18:280–315, 1968.

[LV97] Ming Li and Paul M. B. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, New York, 2nd edition, 1997.

[SL98] A. Siddiqi and S. Lucas. A comparison of matrix rewriting versus direct encoding for evolving neural networks. In *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pages 392–397. IEEE Press, 1998.

[TH03] Gunnar Tufte and Pauline C. Haddow. Building knowledge into developmental rules for circuit design. In Andy M. Tyrrell, P. C. Haddow, and J. Tørresen, editors, *Evolvable Systems: From Biology to Hardware Proceedings of the 5th International Conference on Evolvable Systems, ICES'2003, Trondheim, Norway March 2003*, volume 2606 of *Lecture Notes in Computer Science*, pages 69–80. Springer-Verlag, Berlin, Heidelberg, 2003.

[vRCD+03] Piet van Remortel, Johan Ceuppens, Anne Defaweux, Tom Lenaerts, and Bernad Manderick. Developmental effects on tunable fitness landscapes. In Andy M. Tyrrell, P. C. Haddow, and J. Tørresen, editors, *Evolvable Systems: From Biology to Hardware Proceedings of the 5th International Conference on Evolvable Systems, ICES'2003, Trondheim, Norway March 2003*, volume 2606 of *Lecture Notes in Computer Science*, pages 117–128. Springer-Verlag, Berlin, Heidelberg, 2003.