A comparison between developmental and direct encodings

An update of the GECCO 2006 Paper "The Dead State"

Simon Harding
Department of Computer Science
Memorial University
Newfoundland, Canada
simonh@cs.mun.ca

Julian F. Miller
Department of Electronics
University of York
York, UK
jfm@ohm.york.ac.uk

ABSTRACT

In this paper we compare the evolutionary and run time behaviours of two forms of indirect encoding and a direct encoding. The indirect encodings are comparable to approaches used in the emerging field of developmental systems. We evolve a developmental encoding based on a cellular automaton to grow bit patterns of predefined complexities (in the Kolmogorov sense), and compare its behaviour to the evolution of a direct encoding on the same task. We find that the developmental encodings perform worse at high complexities than the direct encodings, and suggest that this is an artefact from using cell overwriting. Our findings suggest that developmental approaches may scale better for tasks that have complexities below a certain threshold. In this updated version of the paper, we also compare the difference in behaviour of the two approaches for short and long bit strings, and observe that again cell overwriting is detrimental to performance.

Keywords

Developmental Systems, Genetic Algorithms, Cellular Automata

1. INTRODUCTION

It has been suggested that a form of indirect genotype to phenotype mapping, called a developmental mapping, may have advantages over direct encodings of the phenotype. Developmental approaches are a bio-inspired technique that uses a concept of growth to map a genotype (an individual's DNA) to a phenotype (a mature individual). Typically such systems are modelled at a cellular level, where at the beginning of the developmental process there is a single cell. A set of rules, typically an evolved program, then defines the behaviour of the cell. One of the most important behaviours is that of growth. The program can instruct the cell to replicate in some fashion, producing another cell in the environment. The cells in a developmental environment are allowed to interact with each other, which in turn, produces a complex set of interactions between the cells in the environment.

It has been thought that developmental systems may allow us to evolve more complicated, robust and scalable solutions to many tasks, such as neural networks, electronic circuits or computer programs. Direct encodings, it has been argued, do not scale well - the size of the genotype increases as the size of the phenotype, and that if the phenotype has interacting components the system becomes un-evolvable. However, only meagre evidence been produced for the unscalability of direct mappings. Usually, researchers utilize rather nave direct mappings, and also attempt to address relatively simple problems. Comparing developmental systems to direct encodings is difficult. It is important to choose comparable direct and indirect representations, and then use "optimal" parameters to prevent differences in behaviour being caused by poor configuration.

The motivation for this paper is to try to get at what kind of problems are most suitable to be tackled using developmental mappings. Our way of approaching this is to look at problems which have predefined complexities (in the Kolmogorov sense). High complexity patterns are random in nature and almost incompressible. While low complexity patterns are nearly completely ordered and highly compressible. We do not expect developmental mappings to be useful for either of these problems. Instead, we expect developmental mappings to be useful for patterns that are of intermediate complexity, perhaps being characterized by having randomness on a small scale and order on a larger scale. It is not clear that Kolmogorov complexity is the most suitable meeasure with which to classify various patterns of this type. However, we are unaware of another more suitable measure. Notwithstanding these negative remarks about our approach reported in this paper, we feel it is attractive to try to classify in a continuous manner (via complexity) a range of target patterns (bit patterns) and to examine the relative differences between evolving direct mappings and developmental ones. Accordingly, we present, in this paper, initial results for an approach that aims to minimise the differences between the direct and indirect mappings, in order to see the difference in behaviour of the two systems.

1.1 Related Work

A number of researchers have considered developmental mappings and compared them to direct mappings.

Kitano developed a method for evolving the architecture of an artificial neural network using a matrix re-writing system that manipulated adjacency matrices[8]. Although Kitano claimed that his method produced superior results to direct methods (i.e. a fixed architecture, directly encoded and evolved), it was later shown in a more careful study that the two approaches were of equal quality [13].

Gruau devised a graph re-writing method called cellular encoding [4]. Cellular encoding is a language for local graph transformations that controls the division of cells which grow into artificial neural networks. The cells (which we can identify as nodes in the ANN) store connection strengths (weights) and a threshold value. The cells also store a grammar tree that defines the graph re-writing rules and a register that defines the start position in the grammar tree. When cells divide the daughter cells are identical to their parent. The grammar tree was evolved using an evolutionary algorithm. This method was shown to be effective at optimising both the architecture and weights at the same time, and scaled better than a direct encoding (where all the weights had to be independently evolved) [5].

Bentley and Kumar examined a number of genotype – phenotype mappings on a problem of creating a tessellating tile pattern[3]. They found that the indirect developmental mapping (that they referred to as an implicit embryogeny) could evolve tiling patterns much quicker than a variety of other representations (including direct) and further, that they could be subsequently grown to (iterated) much larger sized patterns. One drawback that they reported was that the implicit embryogeny tended to produce the same types of patterns (i.e. of relatively low complexity). As we will see later our results support this finding.

Hornby and Pollack evolved context free L-systems to define three dimensional objects (table designs)[6]. They found that their generative system could produce designs with higher fitness and faster than direct methods. They point out that generative or developmental systems will scale better than direct methods when there is modularity present. For instance, in the case of furniture design if there is a module that is responsible for producing a table leg, evolution only needs to alter and perfect one module rather than having to independently adjust an arbitrary number of independent table leg producing coding regions. This kind of developmental modularity is evident in nature in the form of Hox genes [14]. It provides a powerful and evolvable mechanism whereby evolution can alter the number of sub-parts in an animal body plan.

Eggenberger investigated the relative merits of a direct, and developmental genetic representation for the difficult problem of optical lens design[7]. He found that the direct method (where the location of optical elements was evolved) scaled very badly when compared with the developmental approach.

Roggen and Federici compared evolving direct and developmental mappings for the task of producing specific two dimensional patterns of various sizes (the Norwegian Flag and a pattern produced by Wolfram 1D CA rule 90)[10]. They showed in both cases that as the pixel dimensions of the patterns increased the developmental methods outperformed the direct. It is noteworthy, that performance disparity was much more marked for the relatively regular Norwegian flag pattern.

2. OUR APPROACH

In our approach we developed a simple, deterministic developmental system based on 1D cellular automata. We compare the behaviour and evolvability of two variants of a phenotype-genotype mapping with a direct encoding of the phenotype. The direct encoding method uses a binary representation, where the fitness of the individual is measured directly from the genotype. The two systems with an indirect mapping use a cellular automaton to construct a simple developmental system. For one of the cellular automata approaches, we implement a form of cell growth and cell death to mimic a more biological developmental process.

Other developmental systems have used 2D cellular automata (e.g. [9, 10]), however we chose 1D to simplify the problem further, and to make exhaustive experiments computationally tractable.

The developmental system consists of a toroidal, 1D binary cellular automata, with a neighbourhood diameter of 3 (i.e. the cells can only observe their immediate neighbours). The genotype-phenotype mapping is performed by iterating the cellular automaton from an initial starting configuration. The transition between the states in the cellular automata comes from a rule set, which we evolve. In this instance, the rules take 3 binary inputs and output a single binary state, thus there are a total of 256 possible rules. The inputs are the states of the current cell, and its neighbouring cells. The output of the rule determines the next state of that cell, and if we are using growth and death, may control the state of its neighbours (see section 2.2).

In these experiments, the cellular automata rules and the initial state are determined by evolution. The task is to find an initial state for the CA and the rules, that produce a specified pattern when the CA is iterated.

2.1 Genotype

The direct encoding and developmental approaches use different genetic representations. However, both of the representations can be considered as binary strings.

For the direct encoding, the genotype is a binary string of the same length as the target pattern, i.e. 50 bits. We use two point crossover and a uniform mutation (see table 2.1 for parameters). The target size of 50 bits was an arbitrary decision. However, it was based on computational practicality. To make statistically meaningful analysis we needed to assess the relative merits of the two types of genotype-phenotype mapping on a large range of target patterns of different complexities.

For the developmental systems, the genotype contains two sections: the initial states and the CA rules. The initial states are represented as a binary string, the states of which are copied into the first row of the cellular automata before evaluation. There are 2^8 possible rules for a 3 input binary system. We encode these rules as an eight bit binary string, with each bit specifying the output state for a particular input pattern. For both sections of the genotype two point crossover and a uniform mutation were used. However, the rules and initial states have different mutation rates (see table 2.1). For the majority of the experiments the length initial state is the same as the length of the target pattern. However, for some experiments the length of the initial state is restricted (so that we could examine the effect of growth

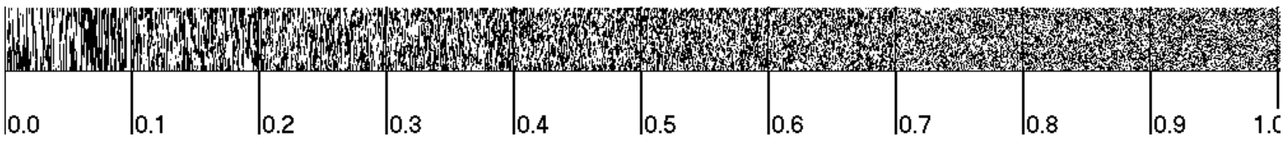

Figure 1: The target patterns. Horizontal axis shows pattern complexity.

in the system).

It should be noted that the length of the genotype for the developmental encoding can be longer than that for the direct encoding. For direct encoding the genotype length is 50 bits. Where the initial state in the CA is the same length as the target pattern, the total genotype length is 58 bits. Where we grow from a smaller initial state, the initial state is of length 6, hence the total length of the chromosome is 14 bits. The length of the smaller initial state was arbitrarily chosen as 6, and in future we wish to investigate how the length of this affects the behaviour of the system.

We used a basic genetic algorithm with a single population, elitism and tournament selection. To allow for comparison between different techniques, we determined all the parameters for the algorithm using the parameter optimisation technique described in section 2.5.

Throughout this paper, the following abbreviations will be used to distinguish between the different encoding types: DIR = Direct Encoding, CA = Cellular Automata, DEV = CA With cell death/growth.

	DIR	CA	DEV
Parameter			
Population	10	36	81
Elitism	2	8	1
Tournament	2	3	7
Mutation rates			
Phenotype	0.035	N/A	N/A
Initial State	N/A	0.0891	0.188
Rules	N/A	0.0115	0.0816

Table 1: "Optimal" parameter values for different experiment types. DIR = Direct Encoding, CA = Cellular Automata, DEV = CA With cell death/growth

2.2 Growth and Death

The ability to grow into neighbouring cells, and for cells to die, is a distinction between the developmental model and a normal cellular automata. In the developmental system where dead states are allowed, a cellular automata update rule is only run for cells that are alive (denoted by state 1). Cells that are type 0 are considered as dead. If a rule produces a live output state and there are dead cells within its neighbourhood in the next iteration, the cell grows into these dead cells. A rule that produces a 0 output can be viewed as killing that cell.

For example, if the current state of the cells is 00100, and the CA rule is such that an input pattern 010 outputs 1. After one iteration the pattern would become 01110. No update rule was executed for the cells in state 0, however the middle cell was alive (state 1) and produced an alive output. This alive state then "grew" into the neighbouring dead cells. In this example, there are 5 cells, however as 4 were dead we consider that the rules were only executed once. In the next stage, the update rule will be executed 3 times.

2.3 Initial States

We compare the behaviour of two types of initial state configurations. In the first instance, the entire length of the cellular automata is specified by the genotype. However, this does not have the sense of "growth" required for a developmental system. Therefore, we also investigate the behaviour of a system where only a section of the initial row can be determined by evolution, with the remaining cells in a dead state.

2.4 The Task

The task is to produce a particular bit pattern. For the developmental systems, the output pattern is the current state of the cellular automaton. With direct encoding, the output pattern is the binary string genotype. The fitness is determined as the proportion of states that are not the same in both the target and the current output of the individual i.e. a hamming distance defined as the percentage of incorrect bits.

For the developmental systems, we did not restrict when the pattern had to occur, however we did record when the nearest match to the target pattern was observed so we can determine if evolution preferred direct encoding to an indirect encoding. The fitness was measured at every iteration of the CA, and the best (minimum) fitness used as the the fitness score. Measuring the fitness at every iteration also means that we do not have to decide when the solution is "mature" (i.e. should be compared with the target).

We generated 1002 different target bit patterns, made up of 11 different complexity classes. Figure 1 shows these targets, with the lower complexity patterns towards the left, and random patterns (high complexities) at the right of the image.

Target patterns were generated with complexities that ranged from 0 (the two patterns which were all either 1 or 0) to 1 (a random and hence incompressible) pattern.

2.5 Parameter Choice

To make the experiments more comparable, for the indirect mappings we used a technique to find the optimal parameters for the evolutionary algorithm. Parameters are derived from a sequential parameter optimization (SPO) of earlier experiments. SPO [1, 2] is a technique, that combines classical and modern statistical approaches, aiming at optimizing parameters of non-deterministic algorithms. SPO has its seeds in the fields of design and analysis of (deterministic) computer experiments (DACE) [11] and global optimization[12]. It extends theses approaches by classical statistics to cope with non–deterministic algorithms.

3. RESULTS

3.1 Direct Encoding

As a baseline comparison, we also evolved the target pattern directly. The performance of the direct encoding is not dependent upon the target pattern (as the bits are independent of each other). Using a population of size 10, with a tournament size of 2, and 2 individuals transmitted to the next population through elitism, we found that on average direct encoding required 705 evaluations (standard deviation 205). At minimum direct encoding with these parameters required 370 evaluations.

The success rate was consistently 100 percent.

3.2 Success Rates

Figures 2and 3 show how the success rate varies as the complexity is increased. Where the initial state was smaller than the target pattern there were few instances where the fitness was 0 (i.e. a perfect solution), therefore, for this experiment success is determined when the fitness is below 0.1. The minimum and average fitness for these results is shown in figure 4.

It is clear that the success rate decreases as the complexity increases, and that in instances where we allow the solution to grow from a smaller initial state the problem difficulty is increased.

Figure 2 shows that the success rate increases slightly when the complexity is > 0.6, the reason for this is unclear and requires further investigation.

It appears that when cell death/growth is allowed that the problem becomes particularly difficult for higher complexities. We suggest that this is because of cell over-writing, which becomes destructive as when a cell grows it can alter the states of three cells. This would seem to be an inherent problem in systems that permit overwriting of cells states (as there is loss of information), and do not grow in a more biological manner by physically adding cells that displace other cells.

From figure 5 we can see that as the complexity of the target pattern increases, the time taken to evolve a set of rules and the initial state increases. Consistently, the use of cell growth/death makes the problem harder for evolution to solve.

3.3 Use of time

We recorded the cellular automata iteration where the highest fitness value was recorded. From this, we are able to see if evolution "prefers" to use the temporal property of developmental systems, or if the easiest action is to evolve the initial state to be the target pattern and then evolve a simple function, such as an identity function.

Figure 6 shows the percentage of solutions that exploited time and that achieved a perfect fitness score. It is clear that in the majority of cases evolution uses the iterative nature of the cellular automata, and that as the pattern complexity increases evolution begins to favour a direct encoding. Where cell death/growth is used, once the complexity exceeds 0.6 there is a massive change, and direct encoding is preferred.

3.4 Computational requirements

For direct encoding, we can consider that the mapping from genotype to phenotype requires no effort, as there is no translation. For the developmental encodings, each time a rule is executed some computation must be performed. Although the rules used in these experiments are simple, other developmental systems use complicated evolved programs. If these programs have to be run many times to generate an output, then it would appear that development could become computationally expensive. On a sequential computer, the larger the number of rules executed, the longer in time it will take for the output to be reached. By their nature, cellular developmental systems operate in parallel. It should be noted that in parallel the total effort is the same, however the total time may be reduced - depending on the implementation. If each cell is physically implemented as it's own processor, it would be expensive, and therefore, practically, we would expect that many cells will be run on a sequential processor.

In these experiments the number of times a rule was executed was counted. For the encoding without growth/death, the number of times the rule is run remains consistent. However, where there are dead cells the rule is not run.

Figure 7 shows that for simpler patterns, the minimum number of times the rule is executed increases with complexity. Both figures 7 and 8 show the average number of times the rules are executed. When starting with an initial state the same size as the target pattern, on average, the rate is the same. However, when growing from a smaller

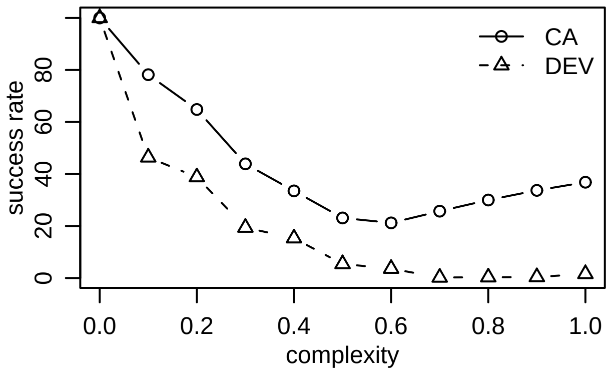

Figure 2: Success rate for different complexity classes, where the initial state is the same size as the target pattern.

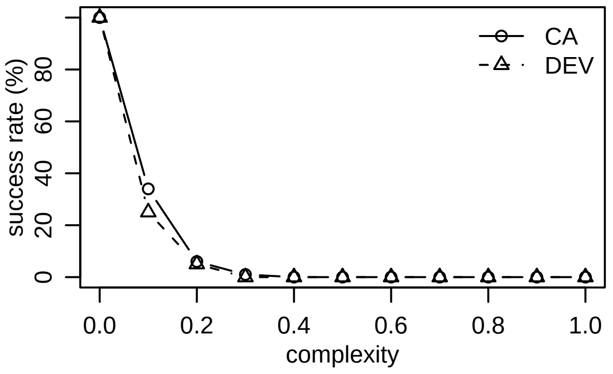

Figure 3: Success rate for different complexity classes, where the initial state is smaller than the target pattern.

initial state, the number of times increases as complexity increases.

This shows a potential advantage for developmental systems that allow for growth. It appears that if the problem is of the right complexity, then you may be able to find a solution and run the genotype phenotype mapping efficiently.

In the context of compression, we certainly see an advantage. For direct encoding the phenotype is 50 bits long, but for the non-direct encoding the genotype is only 14 bits. Unfortunately, for this problem, the target patterns are too short to accurately compare with other forms of compression such as Zip. In principle, the patterns of complexity 1 are totally random, and hence incompressible. Starting from the small initial condition, the best fitness obtained using a normal CA was 0.18 (average fitness 0.3, standard deviation 0.04) and where growth/death are used (DEV) the minimum fitness is 0.24 (average fitness 0.35, standard deviation 0.04). For some applications, such as images or audio, this lossy compression may be acceptable.

4. INITIAL STRING LENGTH

When the initial string length was reduced to 6, and the cellular automata allowed to run from this size, we found that the success rate dropped considerably for all complexities (figure **??**. Figure 14 shows that despite the success rate dropping, the overall behaviour is similar to where the initial state is the same length as the target.

5. EVOLVABILITY FOR LONG BIT STRINGS

Using the same approach, we investigated the behaviour for strings of length 500. Again, we compared the performance across a range of complexity target patterns. One of the criticisms of the original version of this paper was that we should expect development to work better only on larger phenotypes. However, we did not have the time to run the computationally expensive experiments to determine the behaviour in this regard.
We find that in this case both systems performed worse than for the shorter target string.
The first observation is that neither the cellular automata or developmental system managed to find the target pattern for any complexity of target pattern. From the graphs in figures 9 and 10, we can again see that as complexity increases, the quality of the results decreases. Comparing these with the results for the string of length 50, we can see that it is harder to evolve to longer strings, and evolution is unable to reduce the error to the same limits.
In order to see the effect of running the evolution for longer, we increased the number of evaluations per run from 2000 to 5000. The graphs is figures 11 and 12 show that this helped improve the results slightly, but there were still no successful runs nor was there a change in behaviour relating to complexity.

6. CONCLUSIONS

This paper shows that for this model of developmental systems, that direct encoding can sometimes perform worse (in terms of evolvability) than the indirect encodings. However, it appears that this advantage is only true for the lower complexity patterns. For problems of complexity 0 it should be no surprise that there are a large number of rules that can rapidly produce a string of all 1s or all 0s. In this updated paper, we have also shown that this method does not scale in terms of target pattern length, or target pattern complexity and that performing evolution for a longer number of evaluations has limited impact.

We suggest that the cell overwriting issue may be the cause of the poorly performing indirect mappings. To test this, a more sophisticated model will need to be developed, which of course reduces our ability to accurately compare between methods.

The direct encoding is a much more reliable process. In the experiments presented here it always finds a solution. The success rate of the indirect encodings, rapidly diminishes as the complexity is increased.

The ability to use developmental systems as a lossy compression is something that requires further investigation.

It is hinted here that the developmental model here may be useful for problems of low complexity. In these experiments we were able to define complexity ourselves, however, for real world applications, the complexity of the problem is unlikely to be known. This will make choosing the appropriate technique less easy. If you want to guarantee finding a solution, direct encoding seems appropriate. However, if you want to minimise the time spent evolving or the perform some form of compression, than it may be worth attempting a developmental approach.

In nature development appears to produce a large number of repeated units at various levels. At the lowest level there is massive duplication of cells, then at higher levels, there is duplication of body segments. One of the “tricks” used in natural evolution is that cells differentiate from one another largely because of environmental imbalances (e.g. distribution of proteins within cells). These imbalances provide a symmetry breaking mechanism and are not encoded in the genotype. Thus, it suggests that artificial developmental mechanisms should try to make use of this to achieve complexity. In our experiments reported here, all the complexity is imposed at the outset, and there is no mechanism for it to be generated during development. Using a complexity measure based on Kolmogorov may be inappropriate as clearly natural organisms have a complexity that should be measured in some kind of hierarchical way. For example, millipedes are simple when viewed from a high level in that they have many repeating units with the same function. However at a lower level there will be quite a bit of random variation.

7. ACKNOWLEDGEMENTS

The authors would like to acknowledge Christian Lasarczyk (Department of Computer Science,University of Dortmund) for his assistance with the parameter optimisation and visualisation of the results.

8. REFERENCES

[1] T. Bartz-Beielstein. *Experimental Research in Evolutionary Computation – The New Experimentalism*. Natural Computing Series. Springer, Berlin, Berlin, 2006.

[2] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuß. Sequential parameter optimization. In B. McKay et al., editors, *Proc. 2005 Congress on Evolutionary Computation (CEC’05), Edinburgh, Scotland*, volume 1, pages 773–780, Piscataway NJ, 2005. IEEE Press.

[3] P. Bentley and S. Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 35–43, Orlando, Florida, USA, 13-17 1999. Morgan Kaufmann.

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

[5] F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 81–89, Stanford University, CA, USA, 28–31 1996. MIT Press.

[6] G. S. Hornby and J. B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 600–607, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 2001. IEEE Press.

[7] P. E. Hotz. Comparing direct and developmental encoding schemes in artificial evolution: A case study in evolving lens shapes. In *Congress on Evolutionary Computation, CEC 2004*, 2004.

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

[9] J. F. Miller. Evolving a self-repairing, self-regulating, french flag organism. In *GECCO (1)*, pages 129–139, 2004.

[10] D. Roggen and D. Federici. Multi-cellular development: is there scalability and robustness to gain? In X. Yao, E. Burke, J. Lozano, and al., editors, *proceedings of Parallel Problem Solving from Nature 8, PPSN 2004*, pages 391–400, 2004.

[11] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.

[12] M. Schonlau, W. Welch, and R. Jones. Global versus local search in constrained optimization of computer models. In N. Flournoy, W. Rosenberger, and W. Wong, editors, *New developments and applications in experimental design*, volume 34, pages 11–25. Institute of Mathematical Statistics, Beachwood OH, 1998.

[13] A. Siddiqi and S. Lucas. A comparison of matrix rewriting versus direct encoding for evolving neural networks, 1998.

[14] L. Wolpert. *The Principles of Development.* Oxford, UK: Oxford University Press, 1998.

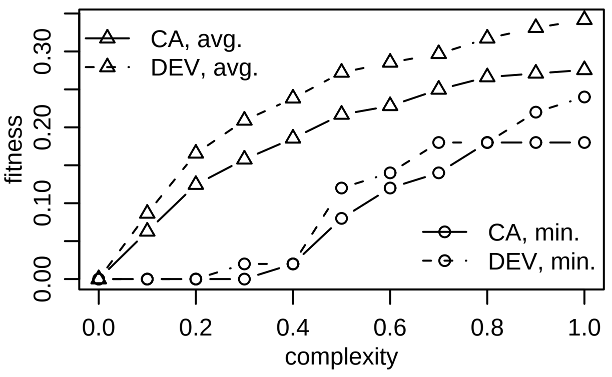

Figure 4: The minimum and average fitnesses achieved where the initial state is smaller than the target pattern, and the target pattern is 50 bits long. Lower is better, with 0 being a perfect score.

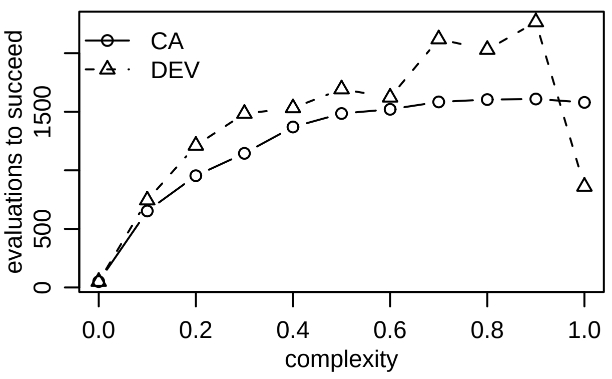

Figure 5: Number of evaluations taken to evolve perfect solutions where the intial state is the same length as the target pattern.

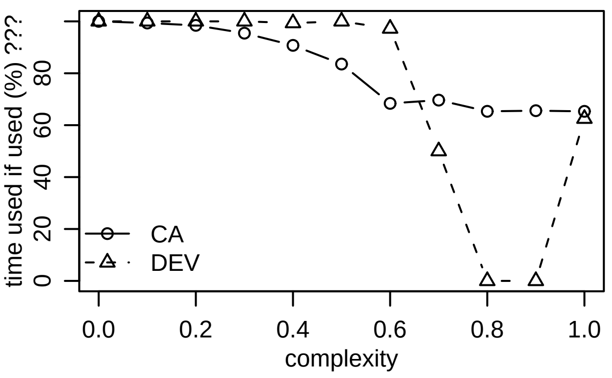

Figure 6: The percentage of solutions that exploited time and that achieved a perfect fitness score

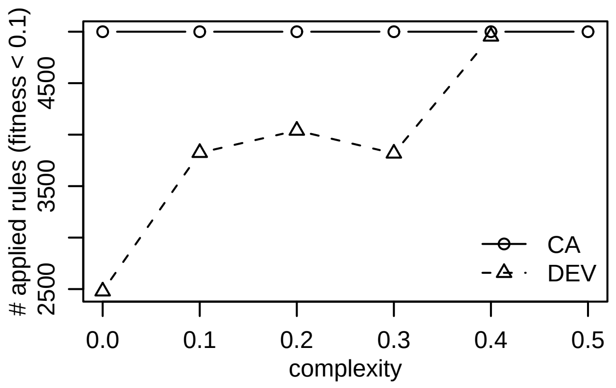

Figure 7: How the number of times a rule is executed for different complexity classes, where the initial state is the same size as the target pattern.

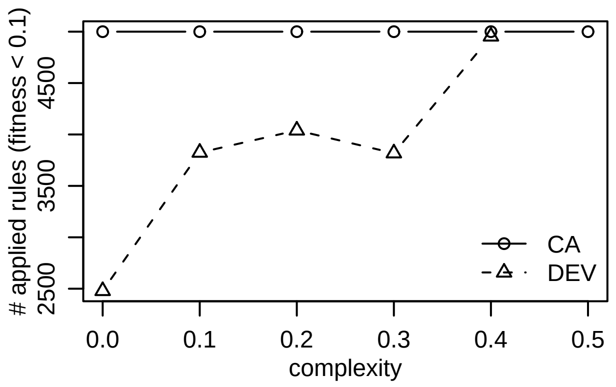

Figure 8: How the number of times a rule is executed for different complexity classes, where the initial state is smaller than the target pattern. It should be noted that as the success rate for these results was low, the results here are for solutions which achieved a fitness of less than 0.1.

APPENDIX

Results for target bit strings of length 500

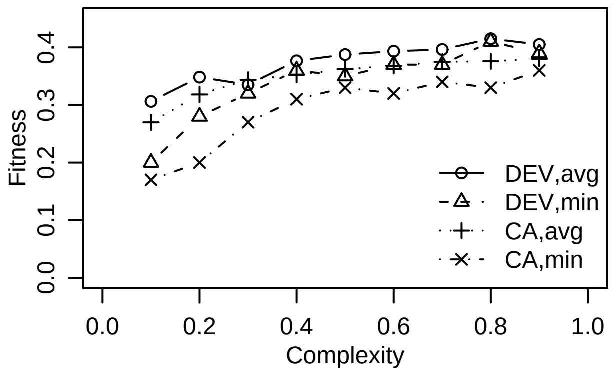

Figure 9: The minimum and average fitnesses achieved where the initial state is smaller than the target pattern, and the target pattern is 500 bits long. Lower is better, with 0 being a perfect score. These results are from running 2000 evaluations.

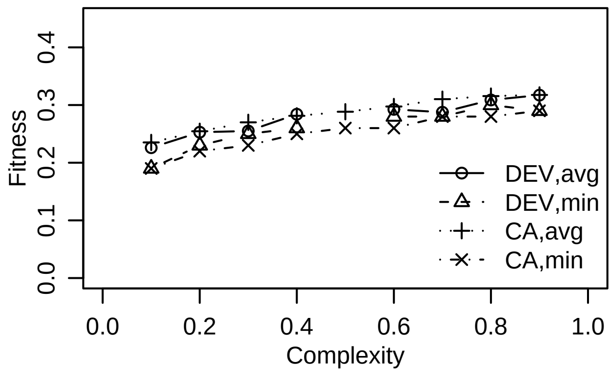

Figure 10: The minimum and average fitnesses achieved where the initial state is the same length as the target pattern, and the target pattern is 500 bits long. Lower is better, with 0 being a perfect score. These results are from running 2000 evaluations.

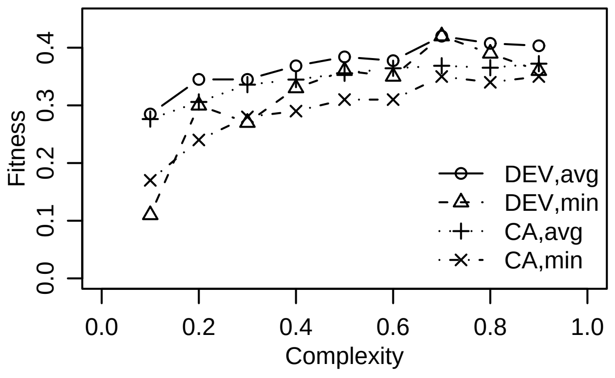

Figure 11: The minimum and average fitnesses achieved where the initial state is smaller than the target pattern, and the target pattern is 500 bits long. Lower is better, with 0 being a perfect score. These results are from running 5000 evaluations.

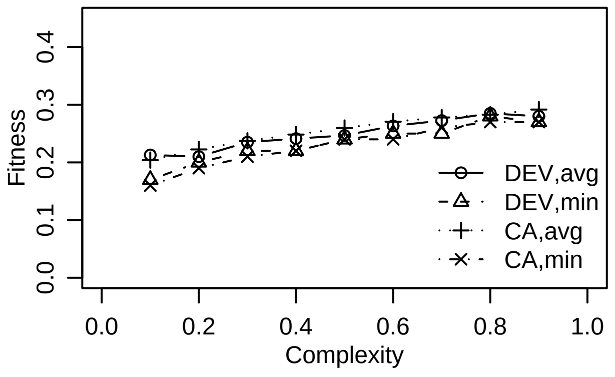

Figure 12: The minimum and average fitnesses achieved where the initial state is the same length as the target pattern, and the target pattern is 500 bits long. Lower is better, with 0 being a perfect score. These results are from running 5000 evaluations.

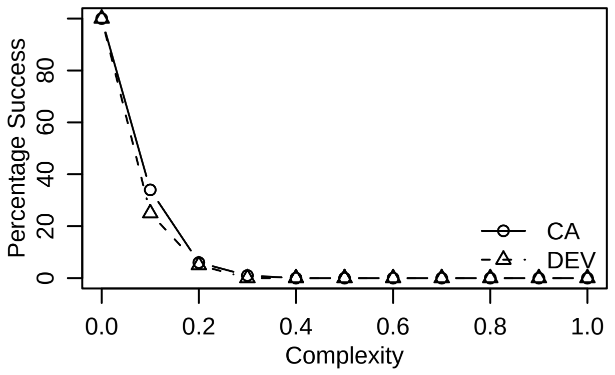

Figure 13: Success rate where the initial state is the shorter than the target pattern, and the target pattern is 50 bits long.

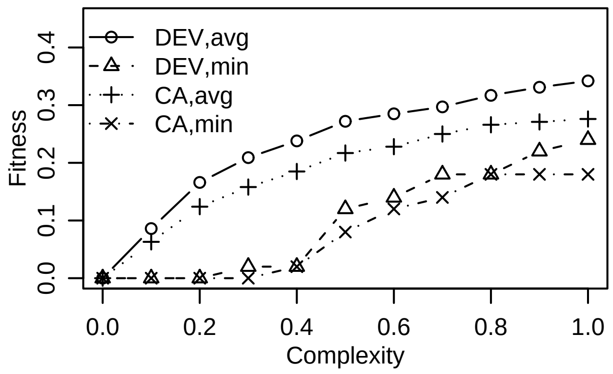

Figure 14: The minimum and average fitnesses achieved where the initial state is the shorter than the target pattern, and the target pattern is 50 bits long. Lower is better, with 0 being a perfect score.