An Evolutionary Methodology for the Automated Design of Cellular Automaton-based Complex Systems

Germán Terrazas Peter Siepmann Graham Kendall Natalio Krasnogor

School of Computer Science and IT
University of Nottingham, Jubilee Campus, Nottingham, NG8 1BB, United Kingdom
[gzt, pas, gxk, nxk]@cs.nott.ac.uk

ABSTRACT

Cellular automata (CA) are an important modelling paradigm in the natural sciences and an extremely useful approach in the study of complex systems. Homogeneity, massive parallelism, local cellular interactions and both synchronous and asynchronous models of rule execution are some of their most prominent features, allowing scientists to model and understand a variety of phenomena in, to name but a few, the physical, chemical, biological, social and information sciences. An ubiquitous problem related with the study of complex systems by means of CA is that of parameter identification. In some cases, analytical methods are available but in many others, due to the bottom-up complexity of the underlying processes, the best route for CA identification is through design optimization by means of a metaheuristic, such as an evolutionary algorithm. In this work we report on a systematic methodology we have developed to control the spatio-temporal behavior of a CA in order to obtain a 'designoid' target pattern. Four independent CA-based complex systems were used to assess our proposal which combines clustering, fitness distance correlation and evolutionary algorithms.

1. INTRODUCTION

Understanding how nature produces and relies upon natural phenomena, such as self-organization, evolution by natural selection, etc., to construct the magnificent engineering solutions routinely find in nature (e.g. eyes, lungs, brains, wings, etc.) is of enormous scientific and technical relevance. Self-organisation, both in the temporal and spatial domain, is a common feature of many complex systems. Such systems have often been studied by means of cellular automata [1][2] and the design of CAs has often been accomplished by means of evolutionary algorithms.

Examples in literature can be found in [3] where a GA was used two evolve a non-uniform CA, a development of the CA paradigm where each cell in the lattice does not use the same rule set. This makes the systems considerably more complex, and thus presents the GA with a more difficult problem, although the papers report that good CAs can still be evolved. In this paper the obtained results were rule sets which allow the CA to generate sophisticated emergent computational strategies subsequently analyzed with regard to the interactions between the system particles. Related work is described in [23] where a coevolutionary approach was used to evolve non-uniform CAs via cellular programming [21] considering three models of asynchrony among blocks of cells. In [4] a research approach using an evolutionary algorithm was employed to discover a new dynamic universal automaton called R capable of building logic circuits and simulating the Game of Life. In [22] and [29], the author reports on the success of a GA in evolving the CA rules themselves (rather than parameters to these rules, as in our case). Particularly interesting is the work based on the fault tolerance of the evolved systems, showing them to have 'graceful degradation' properties.

In this paper we are interested in the automated (evolved) design of the parameter values a CA-based model of a complex system requires in order to attain a specific target spatio-temporal behaviour. This goal is related to the more general aim of designing and controlling complex systems [5]:

"...perhaps the greatest concern is how do we build artificial systems (or manage natural ones) so that the properties that emerge are the ones we want?"

To give a more accurate definition of our objective, we are interested in producing target designoids of the spatio-temporal patterns that emerge from the execution of a set of cellular automaton models. The term 'designoid' was introduced in [6] to refer to objects that seem to have been designed but that were, in fact, evolved.

A CA is defined as an infinite, regular grid of cells, each of which can be in one of a finite number of states. At a given time step, *t*, the state of a cell is a function of that cell's neighborhood at *t-1*. There are a number of possible definitions of a neighbourhood in CA systems as depicted in Figure 1. For instance, the *Moore* neighbourhood uses the eight surrounding cells of the cell in question for the update process, using these eight states as input to the update function. The *von Neumann* neighbourhood only uses the four cells – defined as north, south, east and west – that are strictly adjacent to the central cell. The *Margolus* neighbourhood divides the grid into groups of four cells, to which the update function is applied completely locally (i.e. using only the information in this group of four cells). So as to

allow propagation through the grid, the actual grouping of cells (in the 2x2 arrangement) changes on each update. There are also some extended models as the *Extended Moore* where the distance of the neighbourhood is extended beyond a radius of one.

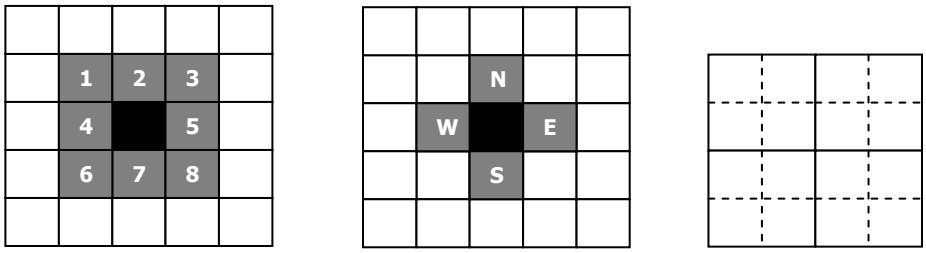
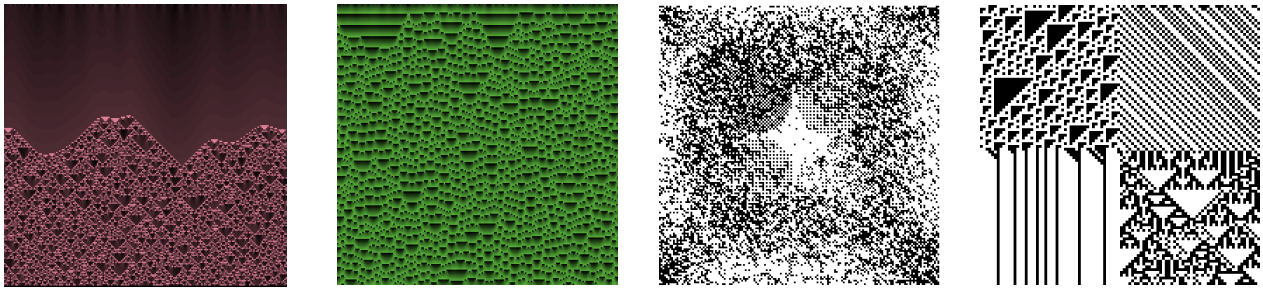


Figure 1: Illustrations of Moore, von Newman and Margolus neighbourhoods.

All the complex system models used in this work have been implemented in the *NetLogo* [7] program. The first model is a cellular automaton known as the coupled map lattice, referred to by *NetLogo* and from hereonin as *CA continuous*. Like a standard CA, it consists of a time-space representation, but the states are encoded as continuous rather than discrete values [8][9]. One of the applications this system has is the modelling of the behaviour of a boiling liquid [10]. At each step, the value of a cell is a function of its neighbours, in essence representing the process of heat diffusion. The same mechanism is also widely used for the study of complex dynamics in nonlinear chemical and biological problems. The second CA model, called *Turbulence*, is also based on a coupled map lattice. This model is used for investigating the relationship between turbulence, laminarity and viscosity of a fluid flowing through a pipe and how the roughness of the pipe surface can affect the fluid's behaviour. The third complex system is the *Gas Lattice* model (also known as the Hardy, de Pazzis and Pomeau model [11]). This program can model the propagation of circular waves. The underlying space is composed of Margolus neighbourhoods, each containing a number of particles, each of which belong to two spacio-temporally separate sublattices, one modelling propagation, the other collision. The last model used in this work was developed by the authors and it is called the *Meta-automaton*. This system is a one dimensional cellular automaton of radius 1. The purpose of this system is to show how the change of dynamics along space and time affects the information flow, to understand how rules behave in a given configuration and how different combinations of rules could affect the complexity of the system. Example patterns generated from all four of these models are shown in Figure 2.



(a) (b) (c) (d)

Figure 2: Sample snapshots of spacio-temporal patterns from the *CA Continuous* (a), *Turbulence* (b), *Gas Lattice* (c), and *Meta-automaton* models (d).

In the following section, a detailed overview of our methodology will be presented, i.e. the evolutionary algorithm, the fitness correlation distance method and clustering. After that, in section 3, a set of experiments, results and analyses of our proposal will be presented. Finally, in section 4, discussions and future work will be shown.

2. METHODOLOGY

In this section, we describe the methodology used to evolve designoid images that capture the target spacio-temporal behaviour of the complex system in question. The principal aim of our proposal is to present a protocol not only for evolving target CA behaviours, but also, most crucially, for verifying the robustness of this evolutionary process. The process, which combines clustering, fitness distance correlation and evolutionary algorithms, seems to be robust across the four different cellular automaton-based complex systems we have investigated, and could have applications to many other complex domains.

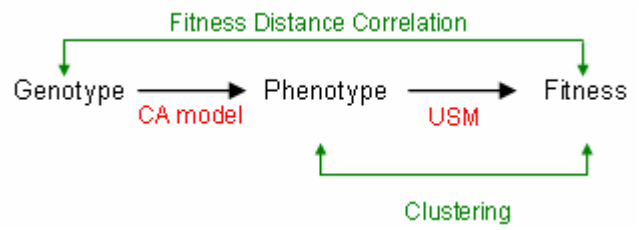


Figure 3: Diagram of mappings from genotype onto phenotype and from phenotype onto numerical fitness value, and relationship to the analysis methods.

Motivation

In a complex system such as the four CA-based models described above, the mapping from genotype to phenotype and then from phenotype to fitness is a highly complex, non-linear relationship. Figure 3 shows the three stage mapping process from genotype (the real numbered parameters) onto a phenotype though the execution of the complex system (the CA model) itself and then from this phenotype (a spacio-temporal pattern) onto a numerical fitness value via the objective function.

Fitness Distance Correlation (FDC) is a measure of how effectively the fitness of an individual correlates to its *genotypic* distance to a known optimum. In other words, given two different genotypes, FDC measures the correlation of the (numerical) Euclidian distance between these genotypes against the value assigned by the objective function. If there is only a small relationship between these two values, a parameter optimisation GA, or for that matter any metaheuristic based on the same representation, will have very little effect. Hence, FDC analyses the genotype – fitness relationship. We must also analyse the phenotype – fitness relationship, in other words, we must verify that the objective function can effectively differentiate between dissimilar phenotypes and effectively classify similar phenotypes as such for the purpose of effective selection. If the fitness function cannot achieve this, a parameter optimisation GA will have difficulty evolving towards better solutions as the selection process will not have sufficient accurate information to bias the search. For verification of the phenotype-fitness relationship, we use clustering. Although FDC has been used in the past to assess the quality of the representation vis-à-vis the fitness function, e.g. [30], it has never been combined with a clustering process to obtain better insight of the complex mappings represented in figure 3.

Such is the complexity of the genotype – phenotype mapping, that FDC can not be guaranteed to give a completely accurate picture, indeed the objective function itself is also only an *approximation* of two individuals' phenotypic similarity. For these reasons, relying on only one of FDC or clustering to validate an objective function would not be adequate. Hence, we use both methods to show that a given function is suitable for use in evolving designoid spacio-temporal behaviour patterns. Details of the fitness function used, as well as details of the implementation of both the FDC analysis and clustering methods are presented in the next section.

2.1 Genetic algorithm

We have used a genetic algorithm whose aim is to generate a spatio-temporal behaviour 'closest' to some specified target image.

Let T be a pre-defined set of real numbers that act as input parameters to the CA model. These parameters give rise to a particular spacio-temporal behaviour, captured in the pattern F_T. We initialise the GA with a population of parameter sets, P_0, P_1, … P_n (which each map to some spacio-temproal behaviour, F_0, F_1, …, F_n) and evolve this population in the hope that a parameter set D will emerge that produces a pattern, F_D as similar as possible to F_T as shown by Figure 4 below.

Each candidate pattern (F_i) is compared to the target pattern (F_T) for similarity using an information distance-based metric described in more detail below. This metric returns a numerical representation of similarity that is considered as the *fitness* of each individual.

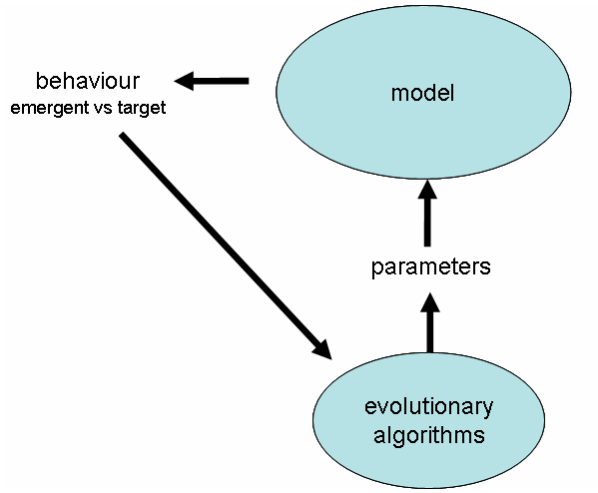


Figure 4: Interaction diagram for the evolution of a given target behaviour. The CA model receives the parameters obtained by the evolutionary algorithm and the emergent pattern is evaluated against the target.

We use a real-coded GA, that is to say, each individual's chromosome (parameter set) comprises a set of real numbers, as opposed to the more traditional bit string. Not only is this a more intuitive representation in this context, but research such as [25] suggests real-coded GAs may be more efficient than their binary counterparts. During the evolution process, offspring are obtained using uniform crossover [26] where, for each gene, the algorithm determines randomly from which parent to draw an allele. Usually each parent has a 0.5 probability of being chosen. Mutation is implemented using the BGA operator introduced in [27] that selects a value from a constant-size distribution either side of the initial value.

Individuals are selected to be parents using the *roulette wheel* selection method [28] which essentially assigns each individual a 'slice' of the roulette wheel whose size is proportional to the individual's fitness. Thus, fitter parents have a greater probability of being selected when the virtual wheel is spun, but all individuals in the population have some chance of selection. The $(\mu+\lambda)$ replacement strategy is employed, where the children and parents are considered together and the best (fittest) μ individuals are chosen to form the next generation's population. The GA is run over 100 generations with a population size of 20, i.e. $\mu=20$. At each generation, 10 offspring are created, i.e. $\lambda=10$. Crossover between parents occurs with 0.7 probability; mutation with 0.3.

Our GA system has been specially developed through the CHELLnet project (`http://www.chellnet.org`) for optimising a range of design and manufacture problems. It is a server-based system and can be tailored to solve a broad range of problems. The number and data type of genes in the chromosome, along with the parameters for the GA (including the user's choice of a range of selection, replacement, recombination and mutation operators and rates) can be specified in the web-based configuration module which builds an XML script as output. This script, along with a plug-in-style problem specification class (which, most importantly, defines the fitness function), configures the GA to the specific problem to be optimised. The execution of the GA can then be started and observed over the Internet through a Java servlet. As shown in Figure 5, this system enables a number of users to run tailored instances of the GA in parallel on different problems.

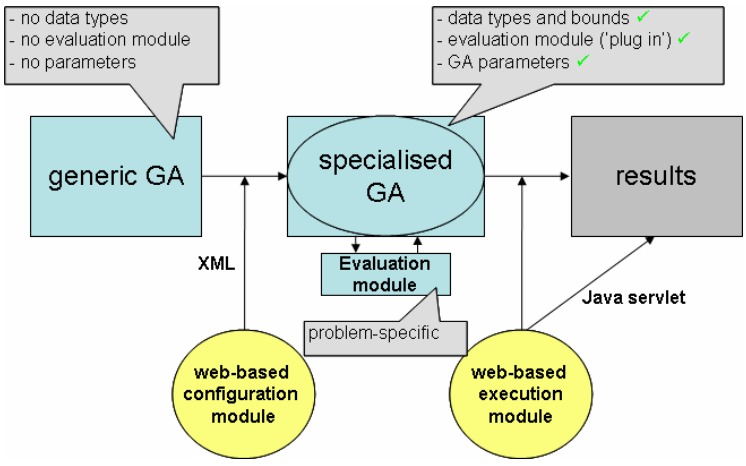


Figure 5: Component diagram of the GA system.

2.2 The Universal Similarity Metric

The Universal Similarity Metric [12] is a measure of the similarity between two objects based on Kolmogorov complexity. This function is a robust compression-based mechanism and has been widely used in different topics of research (see [13] and references therein).

The information distance between two objects is the amount of information required to compute one object given the other. For our purposes, the information distance was calculated with the USM formulae presented in [13] and shown in equation 1, where $K(o_i)$ is the Kolmogorov complexity of the object o_i. The Kolmogorov complexity of an object is defined as the length of the shortest program for computing o_i by a universal Turing machine [14]. Since Kolmogorov complexity is not computable, we have used the same approximation approach as proposed in [13]. The fitness distance correlation and clustering processes described below are used to show that the USM is an effective measure for comparing the spacio-temporal images generated by our CA systems, and thus a robust fitness function for a genetic algorithm.

$$d(o_1, o_2) = \frac{\max\{K(o_1.o_2), K(o_2.o_1)\}}{\max\{K(o_1), K(o_2)\}} \quad (1)$$

2.3 Fitness distance correlation

Finding the most robust way for predicting when a GA will be an effective method of optimisation is still an open topic of research in evolutionary computation theory. One possible methodology, fitness distance correlation (FDC), is proposed in [17]. FDC is a statistical-based methodology which performs a correlation analysis given a known target solution and samples from the search space. Faced with a maximisation problem, a large positive correlation value indicates that the problem may be effectively optimised by a GA, whereas a large negative value suggests that GA optimisation might not be as effective. Correlation values around zero indicate that a more detailed analysis on a scatter plot of fitness versus distance should be performed. The formula for the derivation of a correlation value is shown in equation 2, where r is the correlation coefficient, n is the number of individuals under consideration, f_i is the fitness of individual i, and d_i is its distance to the nearest global optimum, $\bar{f}$ and S_F are the mean and standard deviation of the set of fitnesses, and $\bar{d}$ and S_D are the mean and standard deviation of the set of distances. We performed fitness distance correlation analysis of the USM as applied to the CA generated images in order to extend our study of how effective it could be as a fitness function in a GA when trying to generate a target spatio-temporal behaviour.

$$r = \frac{(1/n)\sum_{i=1}^{n}(f_i - \bar{f})(d_i - \bar{d})}{S_F S_D} \quad (2)$$

2.4 Clustering

In order to further assess the proficiency of the USM as a fitness function, we use clustering. For this to be effective, the definition of a 'similarity' measure or, ideally, a metric is required. Thus, to cluster the spatio-temporal images for each data set described in the next subsection, the distance between each pair of spacio-temporal patterns was measured and recorded in a distance matrix, M. Then, using this matrix, clustering [15] occurs by processing the distance matrix produced above with the clustering algorithm outlined in the Pseudocode 1. A number of different clustering methods and representations are available. In our case, we have used the unweighted arithmetic average method (UPGMA) and logarithmic tree representation respectively. One such implementation of clustering can be found at [16].

```
Clustering()

  Input: F set of spacio-temporal behaviour patterns
  Foreach o_i,o_j∈F where 0≤i,j≤|F|
     P_ij=(o_i,o_j)
     M[i,j] = d(P_ij)
     M[j,i] = d(P_ji)
  T = makeCluster(M)
  Output: T tree of clustered behaviour patterns
```

where
P_{ij} is a pair of patterns o_i and o_j belonging to F
$d(P_{ij})$ calculates the distance between of a pair of objects
M[i,j] is a cell of matrix M storing distance between o_i and o_j
makeCluster(M) implements UPGMA applied to M

```
MakeCluster(M)

  Input: M a distance matrix
  let Minimum = maxNumber
  While dimension(M) > 2x2
      Foreach i,j where i≠j
         Minimum = Min(M[i,j], Minimum)
      mergeRows(M,i,j)
      mergeColumns(M,i,j)
      M[k,ij] = avg(M[k,i], M[k,j])
      M[ij,k] = avg(M[i,k], M[j,k])
      Node = makeNode(i, j)
      insertNode(T, node, minimum)
  Output: T a hierarchical structure
```

where
mergeRows joins the content of rows i and j
mergeColumns joins the content of columns i and j
node associates i and j
insertNode adds a new node in T

Pseudocode 1: Clustering procedure to assess the proficiency of the USM as a fitness function. Distances between all possible pair of objects belonging to a set of spacio-temporal behaviour patterns is stored in a matrix subsequently used in a clustering calculator.

2.5 Models and Data Sets

Here, we describe the CA systems in detail, and the data sets we used to demonstrate our methodology.

2.5.1 The Models

The *CA continuous* model is a general dynamic system for the study of complex behaviours such as boiling a liquid. It consists of a discrete time-space representation whose cells take infinite continuous values, but limited by the precision of the computer. For this system, two variables were considered for generation of the spacio-temporal behaviour: *PRECISION-LEVEL* which gives the precision of the state values, ranging from 1 to 16 decimal places and *ADD-CONSTANT* which is a constant used in the calculation of each cell's value.

The *Turbulence* model is a system designed for understanding how a transition from order to disorder takes place when fluids flow through pipes. Subject to the value of the cells around it, a cell in the lattice maps to either a laminar or a turbulent behaviour. This model also consists of two variables: *COUPLING-STRENGTH* which ranges continuously from 0 to 1 and determines the extent to which cells influence their neighbours and *ROUGHNESS* which ranges from 0 to 0.025 and controls the friction acting on the modelled fluid.

Finally, the *Gas Lattice* model is a complex system modelling how circular propagation of particles takes place. Each cell of the underlying space hosts a number of particles which are subject to collision and propagation as governed by the rules of the automaton model. The variables controlling the system are *DENSITY* which controls the number of particles per cell and *RADIUS* which defines the size of the initial circular wave.

The *Meta-automaton* is a one-dimensional binary cellular automaton composed of a toroidal square lattice of 100 rows by 100 columns where each cell is associated with one of the so-called "256 elementary rules" defined in chapter 3 of [10]. Each rule is encoded with a binary array of length 8 where each bit is associated with a possible configuration given by the state of the cell containing the rule and the state of its two immediate surrounding cells. During runtime, the state of the first row of cells is randomly initialized with either 0 or 1. After that, at every timestep *t* each cell executes its rule considering its internal state plus the state of its two adjacent cells, and changes the state of the cell at timestep *t+1* according to the related value to that configuration. An example of the elementary rule 145 is shown in Figure 6.

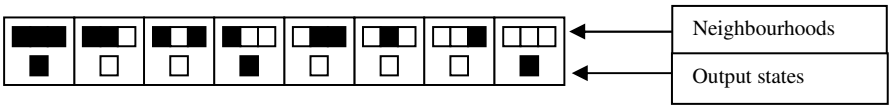


Figure 6: Illustration of the elementary rule 145. Each possible configuration of three cells is associated with an output state used as the new state for the next timestep.

The Meta-automaton is a particular instance of the so called non-uniform automaton [29]. In our case, the *Meta-automaton* also allows the user to partition the system's spatial and temporal dynamics using the variables *K-TIMES* and *T-LIMIT*. That is, groups of k-consecutive cells can be associated to the same rule and re-assignation of rules is allowed to take place every *t* timesteps.

2.5.2 Data Sets

For each of the four CA systems described above, we compiled a data set comprising a number of groups of spacio-temporal patterns. For a given model, all the parameters were fixed except one and a number of different groups of images were produced such that the variable parameter was altered for each group, and all the patterns in a given group were generated using the same parameters. The notation used to identify a particular spacio-temporal pattern in the paper is of the form xyz_pQ where xyz refers to the model itself, p to the group and Q to the pattern within that group. For example, *turb_e5* refers to the fifth image in group *e* (i.e. the group generated using parameter set *e*) using the Turbulence model. Table 1 lists all the models, the number of data sets for each model, their parameters and associated reference to the *NetLogo* implementation.

Model	**Number of Groups**	**Number of Images**	**NetLogo Library**	**Parameters**
CA continuous	11	5 per group	*CA Continuous*	*PRECISION-LEVEL* *ADD-CONSTANT*
Gas Lattice	12	5 per group	*Gas Lattice Automaton*	*COUPLING-STRENGTH* *ROUGHNESS*
Turbulence	10	5 per group	*Turbulence*	*DENSITY* *RADIUS*
Meta-automaton	3	10, 9 and 4		*K-TIMES* *T-LIMIT*

Table 1: Data set names, number of obtained groups per data set, number of images produced per group, name of the NetLogo library, and name of the parameters used for the generation of the spacio-temporal patterns.

3. EXPERIMENTS AND RESULTS

This section describes the analysis and experiments performed using the methodology outlined above. The first part presents an analysis of the USM as a fitness function through the Fitness Distance Correlation method. Then, we present the results of clustering the spatio-temporal images using the USM. Finally, the results of the GA applied to a set of ten target spatio-temporal behaviour patterns will be described.

3.1 Fitness distance correlation

For each group of patterns obtained from the *CA Continuous*, *Turbulence* and *Gas Lattice* models, each image in turn was considered as a target designoid while the remaining images of the group were compared to it using the USM. The resulting plots and correlation coefficients for some of the groups are shown in Figure 7, with further results available at `http://www.cs.nott.ac.uk/~gzt/edbcs`. Overall, the correlation coefficients for the three systems range from 0.2140 to 0.5342, showing that the USM has a relatively high correlation with the genotype of the spacio-temporal behaviour pattern. However, some scatter plots indicate that the USM may only be effective in certain areas of the search space. For instance, we found that plots of the *Turbulence* system suggest that the USM will be effective only for target designoids with coupling-strength values between 0.400 and 0.800. In the second experiment, we considered all the patterns in the data set as a *single* group. As before, each image, in turn, was considered as a target designoid and the remaining images compared to this target. In these results, some correlation coefficients were around zero (and thus present an inconclusive result) from –0.07 to 0.14, with certain structures appearing in the scatter plots showing no relationship between fitness and distance. However, other coefficients range from –0.2018 to –0.2579 and from 0.1642 to 0.6695 with scatter plots suggesting that the problem is suitable for optimisation by an evolutionary algorithm.

Consequently, we can conclude from these results, that FDC is not always a completely reliable and clear indication of a fitness function's suitability, especially when the geneotype – phenotype mapping is as complex as it is in these systems.

3.2 Clustering

The *CA Continuous*, *Turbulence* and *Gas Lattice* data sets were run four times using the algorithm outlined in pseudocode 1. Clustering the eleven groups of spacio-temporal patterns for the *CA continuous* model generated the expected eleven clusters, corresponding with the eleven groups of spacio-temporal patterns. The logarithmic cluster tree for this model can be found at Figure 8. For *Turbulence* behaviour patterns, a similar result was achieved. In this case, there was one outlier pattern, although its location is close to the family group to which it belongs. This can be seen in Figure 9. Full cluster trees, as well as further supporting material for this research can be found in the website referenced in subsection 3.1.

Another positive result was obtained with *Gas Lattice* patterns - eleven groups were created. As a final experiment, the three data sets were processed together. As expected, the characteristics of the three different collections were detected and hence different clusters were created for each group of our data set. Three different logical partitions locating the *Turbulence* model objects in the top, the *CA Continuous* in the middle and the *Gas Lattice* model instances at the bottom were easily identifiable (see Figure 9).

Thus, we conclude that using the USM as part of the clustering process of complex system patterns was successful. It captured not only the change of behaviour that different values of a parameter produce in a complex system but also the dissimilarities between different spacio-temporal behaviour patterns.

As a general conclusion, we can state that the FDC plus clustering analysis presented in subsection 3.1 and 3.2 indicate that the use of USM – with both the chosen representation and genotype to phenotype mapping – are amenable for the evolutionary design of complex systems such as CAs. However, the FDC analysis and scatter plots also reveal that some of the target spatio-temporal patterns might be more difficult to evolve than others. So we expect that the evolutionary algorithm will, in some cases, find it difficult to evolve suitable patterns. In the next section we perform evolutionary experiments to test this methodology.

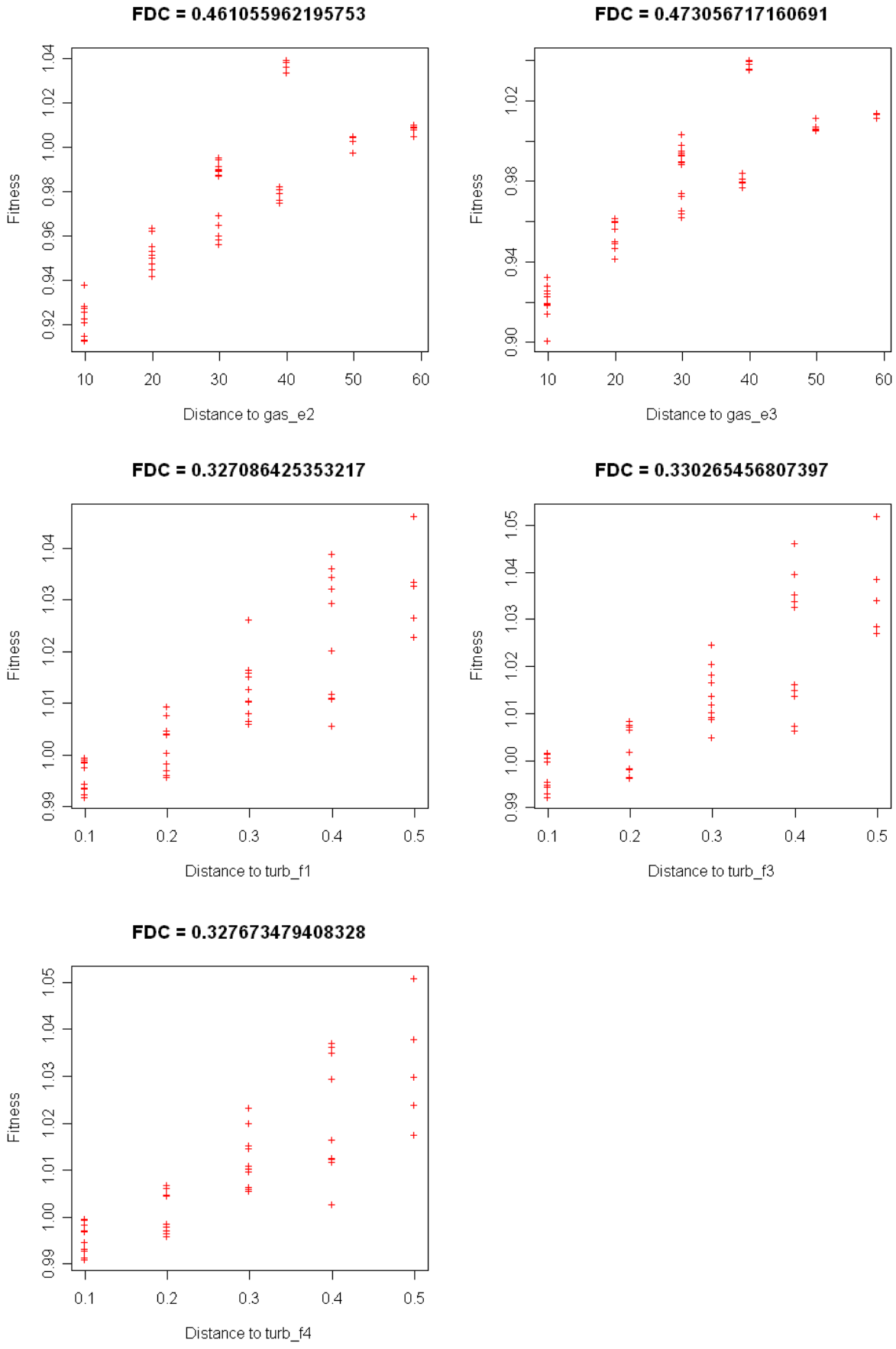


Figure 7: Graphics of the resultant scatter plots and correlation coefficients for the group *e* in the Gas Lattice model and group *f* in the Turbulence model showing that the USM value has a relatively high correlation with the genotype of the spacio-temporal behaviour pattern.

Figure 8: Illustration of the logarithmic cluster tree for patterns belonging to the *CA continuous* model. Clustering the fifty five images of patterns for the *CA continuous* model have generated the expected clusters, corresponding with the eleven groups of spacio-temporal patterns.

Figure 9: Illustration of the logarithmic cluster tree for patterns belonging to the *Turbulence*, *CA Continuous* and *Gas Lattice* models. The characteristics of the three different collections were detected giving three different logical partitions locating the *Turbulence* model objects in the top, the *CA Continuous* in the middle and the *Gas Lattice* model instances at the bottom.

3.3 Genetic algorithm

Here we present the results of the GA experiments using the *Turbulence* model as a first instance and the *Meta-automaton* model as a second test case. These two models are particularly well-suited to evolutionary design as the resultant pattern captures the entire spacio-temporal behaviour of the system, i.e. each subsequent row of the image represents the system behaviour at the next time step. This is not the case for the *Gas Lattice* model, where each time step is associated with an entire image, which would make similarity calculations prohibitively compute- and time-intensive. The *Turbulence* model was run in preference to the *CA continuous* model as it is a more complex, and therefore more interesting model.

The *Turbulence* Model

For this experiment, the *Turbulence* model was initialized with the parameter *initial turbulence* ranging from 0 to 100, *coupling strength* ranging from 0 to 1 and *roughness* from 0 to 0.025. All three parameters are free to take any value in their range, each to an accuracy of 64 bits. The generation of the spacio-temporal behaviour patterns from these parameters is automated through a simple wrapper using *NetLogo*'s Java API [7].

We ran the GA five times (capped at 100 iterations) on two groups (*e* and *f*) of *Turbulence* patterns which each comprise five target images. As explained in section 2.5.2, each target in a given group was generated using the same parameters, but as the *Turbulence* model is stochastic, each spatio-temporal pattern generated is somewhat different. Table 2 shows the values for the most successful of the five runs for each target image in each group.

Given the three parameters for the *Turbulence* model, initial turbulence *i*, coupling strength *c*, and roughness *r*, for a given spacio-temporal behaviour pattern, $F = \{i_F, c_F, r_F\}$ and a given target, $T = \{i_T, c_T, r_T\}$, we define the error for each gene, $e(g) = \frac{abs(g_T - g_F)}{g_T}$ and the average error for a given individual, $E(F) =$.

Visual inspection of Figure 10 shows that for groups *e* and *f*, the USM values below 0.96 show that the resultant images are similar to the target in terms of information distance, and there are certain visual features which can be picked out; both resultant images share the targets' density of light pixels, with a number of dark triangles dispersed throughout the image and the larger triangles at the top of T_{e5} have been succesfully represented in the F_{e5}.

Looking at Table 2, it is evident that the GA has mixed success in approximating the actual target parameters. The error margins range from a most satisfactory 4.3% (c_{f3}) to a widely inaccurate 1277% (r_{f2}). It is interesting to note that the worst errors are all for parameter *r* (roughness). This suggests that it may be the case that *r* is the least influential in the generation of the images, and indeed a brief experimentation with the *Turbulence* program reveals that this is indeed the case, at least when combined with these values of *i* (50.5) and *c* (0.5). It appears that when *i*, the initial turbulence, is high, as in this case, a change in *r* makes little difference, but when *i* is low, *r* is far more influential. Indeed, this agrees with the physical dynamics of fluid flow which the system is modelling – if the fluid is initially perturbed, we can intuitively surmise that the roughness of the pipe will have a lesser effect than when the fluid is initially undisturbed. These observations show that, although an interesting indication, exact approximation of the parameters is not necessarily a good indication as to the similarly (or lack thereof) of two spacio-temporal behaviours. This is just a further confirmation of the highly complex, non-linear and stochastic nature of the genotype – phenotype mapping.

With images such as T_{e5} and T_{f3}, whose parameters are such that they are not visually very different, the quality of the results is difficult to see. However, if we use a target pattern which is more visually distinctive, results are much clearer: A further target, T_P was defined, giving a much more distinctive image (with a low value for *i*, giving more influence to *r*). The results for this target (shown in Figure 10) are noticeably better - not only is the resultant image visually very similar to the target (as indicated by a USM value of 0.91980 - substantially lower than that of the previous experiments), but the parameters have all been approximated to within about 35% of the target with a combined error value $E = 0.22569$, and interestingly, *r* is now the most accurate approximation ($e(r) = 0.05552$).

Target	i_F	c_F	r_F	i_T	c_T	r_T	$usm(F,T)$	$e(i)$	$e(c)$	$e(r)$	E
e1	62.90586	0.52350	0.00074	50.50000	0.50000	0.00100	0.95657	0.24566	0.04699	0.25948	0.18405
e2	69.47874	0.48819	0.00094	50.50000	0.50000	0.00100	0.95738	0.37582	0.02361	0.06288	0.15410
e3	51.97374	0.51587	0.01130	50.50000	0.50000	0.00100	0.95725	0.02918	0.03173	10.29811	3.45301
e4	71.83373	0.46997	0.00080	50.50000	0.50000	0.00100	0.95817	0.42245	0.06005	0.19738	0.22663
e5	43.03929	0.51941	0.00838	50.50000	0.50000	0.00100	0.95653	0.14774	0.03883	7.38250	2.52302
f1	55.92625	0.56941	0.00791	50.50000	0.60000	0.00100	0.95785	0.10745	0.05098	6.91322	2.35722
f2	65.98295	0.58869	0.01377	50.50000	0.60000	0.00100	0.95657	0.30659	0.01885	12.77470	4.36672
f3	68.37877	0.60256	0.01140	50.50000	0.60000	0.00100	0.95608	0.35404	0.00426	10.40107	3.58646
f4	53.86632	0.60256	0.00673	50.50000	0.60000	0.00100	0.95915	0.06666	0.00426	5.73318	1.93470
f5	48.21686	0.60334	0.00549	50.50000	0.60000	0.00100	0.95781	0.04521	0.00556	4.48623	1.51234
p	8.85724	0.54241	0.00356	6.98285	0.83854	0.00377	0.91980	0.26843	0.35314	0.05552	0.22569

Table 2: GA results for the most successful of the five runs for each target image in groups *e* and *f* of *Turbulence* patterns.

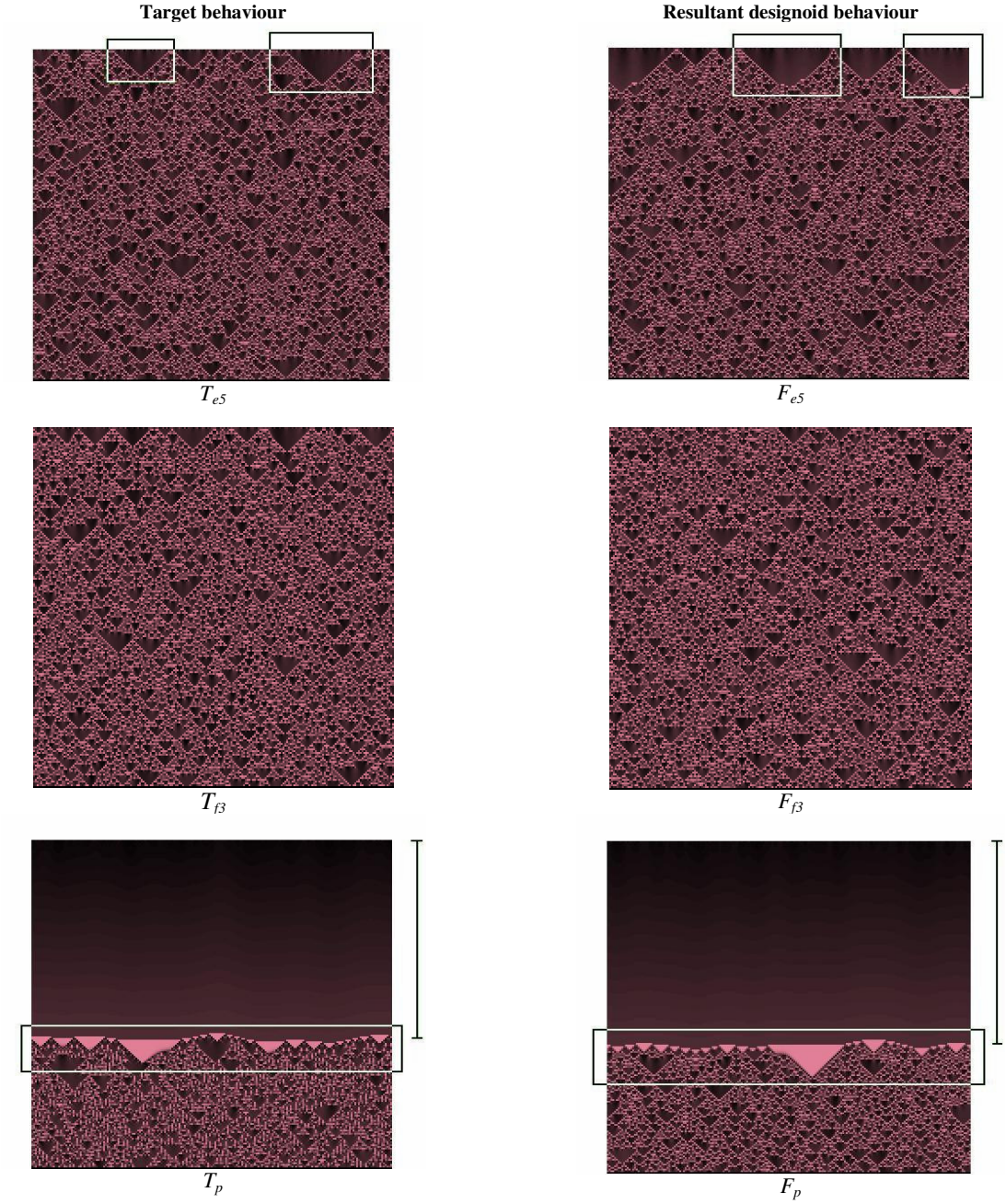


Figure 10: Snapshots of the target and designoid spacio-temporal behaviour patterns with annotations showing particularly well-produced features for the *Turbulence* model.

The *Meta-automaton* Model

Three groups of target patterns were defined using the meta-automaton system. In the first set, all the cells have been initialized with the same rule without considering dynamic rule reassignment. The target patterns produced by the automaton are shown in Figure 11.

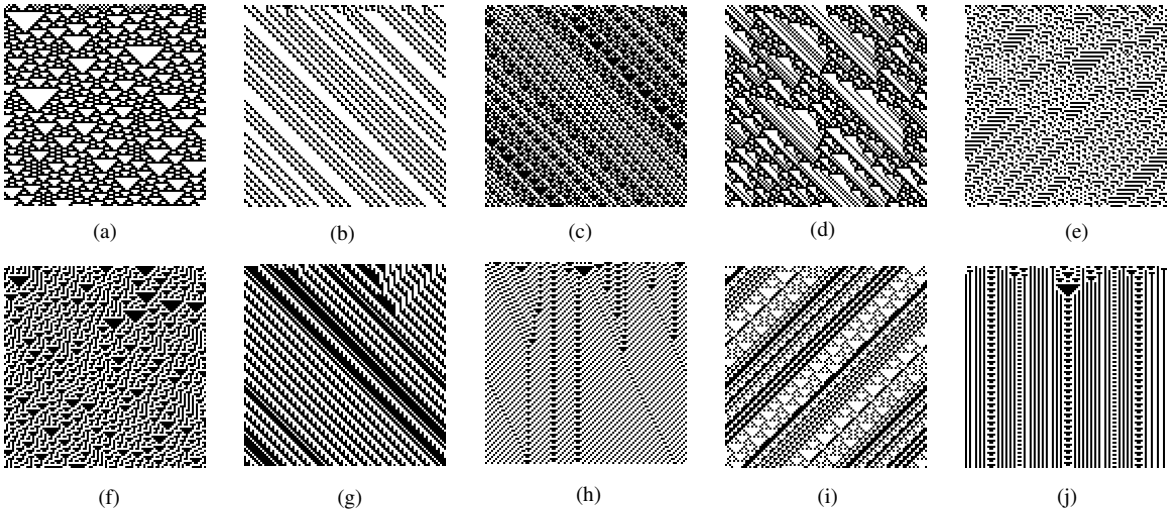


Figure 11: Target patterns produced by the *Meta-automaton* using the same rule in all cells. (a) meta_a1, (b) meta_a2, (c) meta_a3, (d) meta_a4, (e) meta_a5, (f) meta_a6, (g) meta_a7, (h) meta_a8, (i) meta_a9, (j) meta_a10.

For the second group of target patterns, the spacial dynamics were divided in two. That is, given two random rules chosen from the pool of 256 rules, the first consecutive 50 cells were associated with one rule and the remaining 50 with the other. As in the previous data set, there is no reassignment of rules during runtime. The obtained patterns are shown in Figure 12.

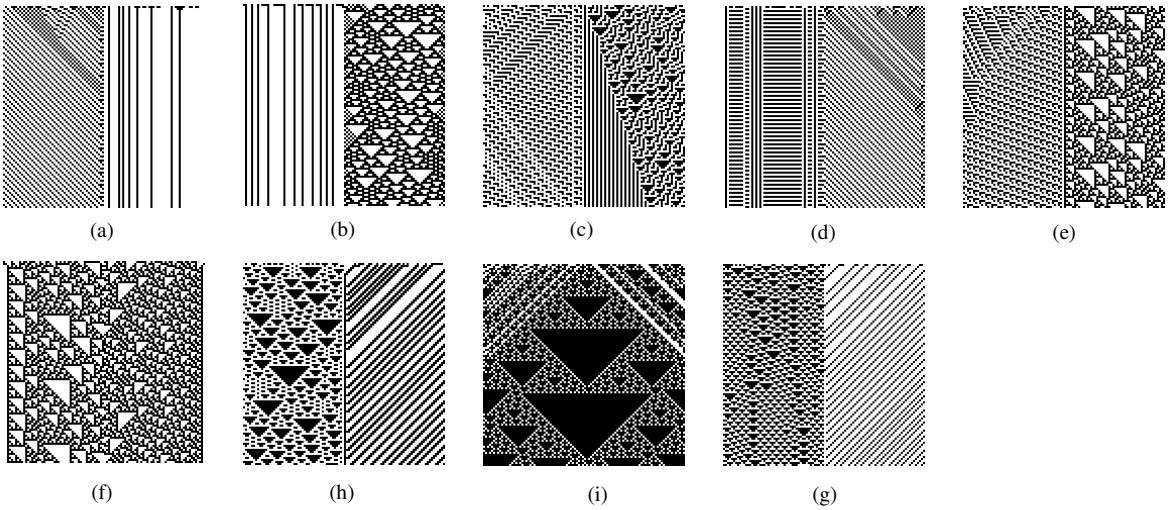


Figure 12: Target patterns produced by the *Meta-automaton* changing dynamics over space with two different rules: (a) meta_b1, (b) meta_b2, (c) meta_b3, (d) meta_b4, (e) meta_b5, (f) meta_b6, (g) meta_b7, (h) meta_b8, (i) meta_b9.

In order to provide the GA with an even more challenging data set, the spacial dynamics were divided into four for the last group of target patterns. In this case, cells were divided in groups of 25 consecutive cells and a randomly selected rule le from the pool of 256 was associated to every cell belonging to the same group. The obtained patterns for this data set are shown in Figure 13.

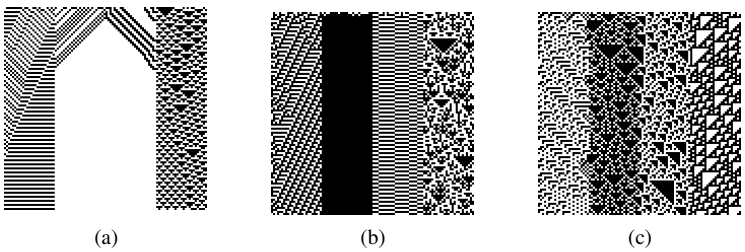


Figure 13: Target patterns produced by the *Meta-automaton* changing dynamics over space with four different rules: (a) meta_c1, (b) meta_c2, (c) meta_c3.

We ran the evolutionary algorithm on the meta-automaton once per target pattern, each individual was evaluated 10 times per generation and the mean USM value was taken as fitness. The results for the first, second and third experiments are shown in Table 3, Table 4 and Table 5 respectively. The *Patterns* column identifies the target pattern, *OR* refers to the set of original rules used to generate the target pattern and *ER* shows the rules evolved by the GA. The *USM* column contains the fitness values of the best individuals whilst the *ST* column (similarity type) classifies the resemblance of the evolved pattern to the target image.

As shown in Table 3, five out of ten experiments have evolved the expected rule for the first data set. However, if we further analyse the remaining results we see that in most of the cases where the expected rule was not achieved, the evolved rule often results either in a mirror or otherwise similar image. For instance, as Figure 14 (a, b) show the designoid target patterns found for *meta_a2* and *meta_a4* are mirrors. Moreover, as depicted in Figure 15 (a), it is clear that the diagonal black strips appearing in *meta_a9* were well captured by the USM. We can argue, therefore, that certain similarities between the target pattern and the designoid were found in most cases. Contrarily, in the case of *meta_a6* and *meta_a7,* no similarities at all appear despite USM values close to 1.

Patterns	OR	ER	USM	ST
meta_a1	[122]	[122]	0.993958124	Correct
meta_a2	[148]	[6]	1.042744644	Mirror
meta_a3	[181]	[181]	0.984504855	Correct
meta_a4	[120]	[106]	0.983119009	Mirror
meta_a5	[97]	[97]	0.985429776	Correct
meta_a6	[135]	[195]	0.976343879	None
meta_a7	[229]	[195]	1.048922986	None
meta_a8	[131]	[131]	1.00218998	Correct
meta_a9	[154]	[169]	0.987069886	Captured
meta_a10	[133]	[133]	0.950053315	Correct

Table 3: Genetic algorithm results for the *Meta-automaton* patterns using one rule.

Patterns	OR	ER	USM	ST
meta_b1	[177 132]	[164 177]	0.818578016	Mirror
meta_b2	[68 122]	[122 100]	0.885830497	Mirror
meta_b3	[65 135]	[215 146]	0.948304844	None
meta_b4	[5 57]	[115 192]	0.870995252	None
meta_b5	[25 60]	[26 125]	0.96081944	None
meta_b6	[60 102]	[183 20]	0.964207074	None
meta_b7	[147 2]	[130 147]	0.905361748	Mirror
meta_b8	[129 46]	[126 16]	0.958283213	Captured
meta_b9	[167 180]	[91 167]	0.993560531	Captured

Table 4: Genetic algorithm results for the *Meta-automaton* patterns using two rules.

Patterns	OR	ER	USM	ST
meta_c1	[49 34 84 147]	[73 141 188 230]	0.907788419	Captured
meta_c2	[61 251 23 165]	[38 140 105 234]	0.917228868	Captured
meta_c3	[41 183 195 110]	[61 120 146 196]	0.940763235	Captured

Table 5: Genetic algorithm results for the *Meta-automaton* patterns using four rules.

In the case of the second data set, equivalent rules, mirrors and close similarities are also found. As shown in table 4, none of the results have reached exactly the correct rules. However, five results out of ten are acceptable - three designoids end up producing mirror images and two reproduce important features appearing in the target patterns. In fact, Figure 14 (c, d, e) show that a target rule plus an equivalent rule were found in the case of the mirrored *meta_b1*, *meta_b2* and *meta_b7*. On the other hand, Figure 15 (b, c) show that in *meta_b8* the lightest areas are shown as the darkest in the left hand side of the image whilst for *meta_b9* the USM was able to capture the small dark triangles at the right hand side of the image.

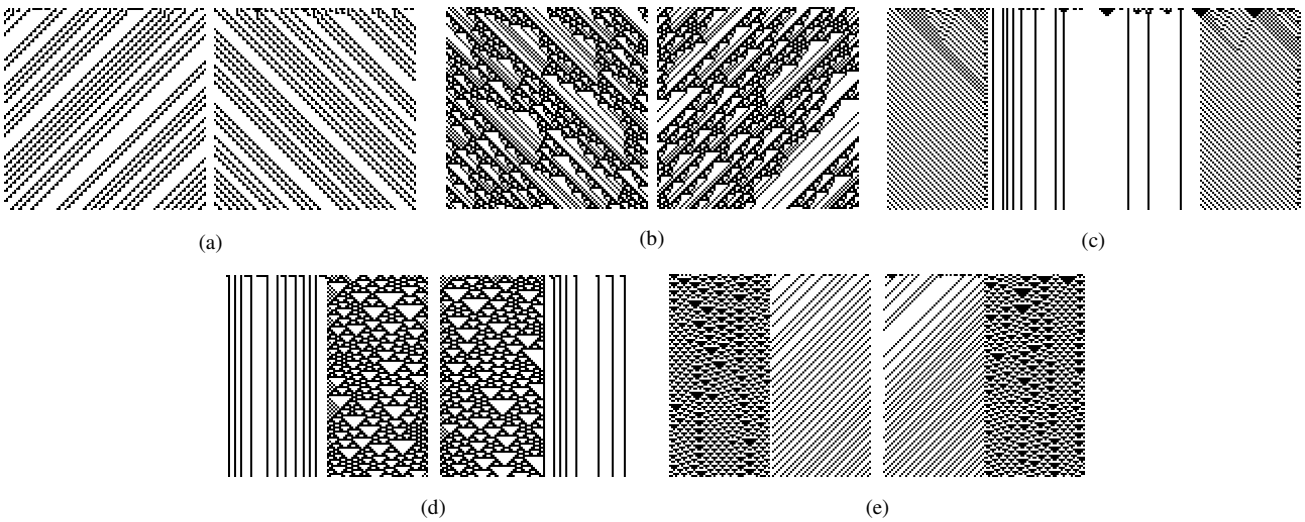
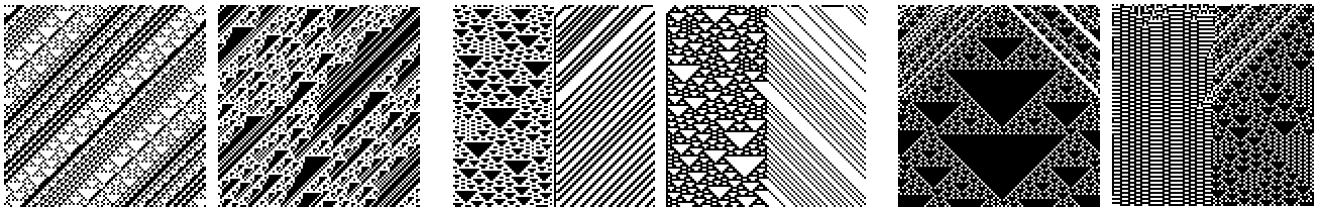
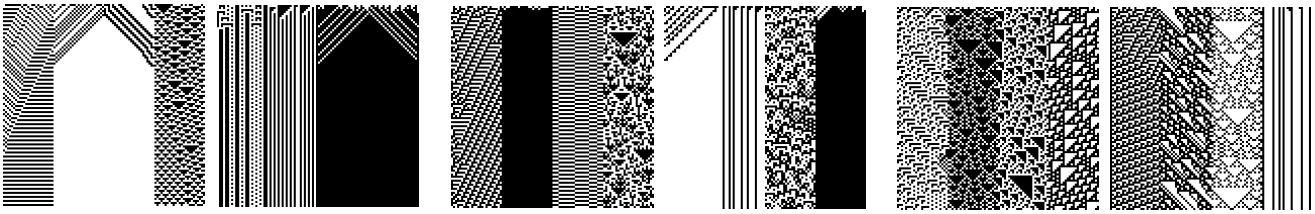


Figure 14: Mirrored designoid patterns found for *Meta-automata* patterns using one and two rules: (a) meta_a2 and its mirror, (b) meta_a4 and its mirror, (c) meta_b1 and its mirror, (d) meta_b2 and its mirror, (e) meta_b7 and its mirror.



(a) (b) (c)

Figure 15: Captured similarities for the *Meta-automata* patterns using one and two rules: (a) meta_a9 and its similar designoid, (b) meta_b8 and its similar designoid, (c) meta_b9 and its similar designoid.



(a) (b) (c)

Figure 16: Captured similarities for the *Meta-automata* patterns using four rules: (a) meta_c1 and its similar designoid, (b) meta_c2 and its similar designoid, (c) meta_c3 and its similar designoid.

In the case of the third data set, mirrors were more difficult to produce and, as Table 5 reveals, none of the results have reached the correct rules. However, a visual analysis of the obtained designoids supports the idea that some relevant features were captured from the target patterns. For example, in Figure 16 (a) it is interesting to note that two rules for producing the inverted v-shape drawing in the middle were discovered (but with inverted colouring and in a different position). Moreover, in Figure 16 (b) an equivalent rule for the second strip was discovered at the fourth position in the designoid and the chaotic behaviour of the last strip is represented in the third position of the designoid. Finally, in Figure 16 (c) a similar effect of colours inversion occurred between the second and third strip of the target pattern and designoid respectively.

Even in those results where the exact rules have not been found, the nature of the rules used in the Meta-automaton mean that a number of different rules can have very similar spacio-temporal behaviour. Hence, as seen in the results for the *Turbulence* models, a significantly different genotype can, in fact, result in a similar phenotype – yet another illustration of the complex, non-linear nature of the genotype-phenotype-fitness mapping in these systems). It is evident from these experiments that, although the USM's information distance-based metric works well in many cases, it has a number of shortcomings. As illustrated by the results above, one of the most obvious is its blindness to negative images. (Intuitively, using the description of conditional Kolmogorov complexity described above in section 2.2, we can see how the amount of information needed to produce, for example, a segment of black pixels, given a segment of white pixels is equal to that needed to produce a segment of white pixels given a segment of black pixels). Similarly, the USM does not differentiate between mirror images – the actual information content of a mirrored image is identical to its unmirrored counterpart. It is a logical progression, therefore, to extend the fitness function using an additional measurement such as hamming distance or an entropy analysis. Using a measure of Hamming distance involves calculating the colour difference between target and designoid on a pixel-by-pixel basis. Alternatively, an image's entropy is an estimate of the distance between two images based on calculating the frequency of the appearance of different sub-blocks or fragments of the images. In either case, the fitness function would need to be extended to either a multi-objective setting or a single weighted function.

4. DISCUSSION AND FUTURE WORK

In this paper we have presented a methodology to control the spatio-temporal evolution of a CA in order to obtain a 'designoid' target spatio-temporal behaviour by combining clustering, fitness distance correlation analysis and an evolutionary algorithm.

We can conclude that the clustering and fitness distance correlation together are good indicators of the quality of the encoding, i.e. genotype, its mapping to phenotype and the fitness function evaluation of phenotypes. The application of this methodology before starting long and expensive evolutionary runs should be considered.

In the cellular automata examples we presented, the methodology gave some support for the use of a compression-based information distance metric such as the USM as a fitness. However, from the analysis of the FDC and

clustering, one could expect (and this was indeed confirmed by the evolutionary runs) that there would be cases where the USM cannot properly inform the evolutionary process. Moreover, an introspective analysis of the cases where the FDC reported no or low correlation and where the USM induced bad clustering can shed light of ways on improving the fitness function used.

There are a vast number of systems that can be modelled by cellular automata and we expect that the methodology described here could be helpful in some of these cases.

5. ACKNOWLEDGEMENTS

We would like to thank Dr. J. Bacardit for his constructive comments on this paper. We also thank the EPSRC for funding P. Siepmann under project EP/D021847/1.

REFERENCES

[1] B. Chopard and M. Droz. *Cellular automata modeling of physical systems*. Cambrige University Press, Cambridge, 1998.

[2] Kier, Lemont B., Seybold, Paul G., Cheng, and Chao-Kun. *Cellular Automata Modeling of Chemical Systems*. Springer, 2005.

[3] M. Mitchell, J. Crutchfield, and R. Das. Evolving cellular automata with genetic algorithms: A review of recent work, 1996.

[4] Emmanuel Sapin et. al. A new universal cellular automaton discovered by evolutionary algorithms. In Kalyanmoy Deb et. al., editor, *Genetic and Evolutionary Computation – GECCO-2004, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 175–187, Seattle, WA, USA, 26-30 June 2004. Springer-Verlag.

[5] G.D. Green and D. Newat. Towards a theory of everything? - grand challenges in complexity and informatics. *Complexity International*, 8, 2001.

[6] R. Dawkins. *Climbing Mount Improbable*. Penguin Books, 1996.

[7] U. Wilensky. *NetLogo*. Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL, 1999. http://ccl.northwestern.edu/netlogo

[8] J. P. Crutchfield and K. Kaneko. Phenomenology of spatio-temporal chaos. *Directions in Chaos*, pages 272–353, 1987.

[9] Kunihiko Kaneko. *Collapse of Tori and Genesis of Chaos in Dissipative Systems*. World Scientific Pub Co Inc, 1986.

[10] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, Inc., 2002.

[11] O. De Pazzis J. Hardy, Y. Pomeau. Time evolution of two-dimensional model system. invariant states and time correlation functions. *J. Math. Phys*, 14:1746–1759, 1973.

[12] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul Vitányi. The similarity metric. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 863–872, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[13] Natalio Krasnogor and David A. Pelta. Measuring the similarity of protein structures by means of the universal similarity metric. *Bioinformatics*, 20(7):1015–1021, 2004.

[14] A. N. Kolmogorov. Three approaches to quantitative definition of information. *Problemy Paredachi Informatsii*, 1:3–11, 1965.

[15] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *J. Intell. Inf. Syst.*, 17(2-3):107–145, 2001.

[16] John Brzustowski. *Clustering Calculator*. U. Wilensky. *NetLogo*. Department of Biological Sciences, University of Alberta, 2000. http://www2.biology.ualberta.ca/jbrzusto/cluster.php

[17] Terry Jones and Stephanie Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[18] D. Schlierkamp-Voosen H. Muhlenbein. Predictive models for the breeder genetic algorithm. i: Continuous parameter optimization. *Evolutionary Compututing*, 1(1):25–49, 1993.

[19] Francisco Herrera, Manuel Lozano, and Jose L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, 12(4):265–319, 1998.

[20] A. Adamatzky and B. de Lacy Costello. Collision-free path planning in the Belousov-Zhabotinsky medium assisted by a cellular automaton. *Naturwissenschaften*, 89:474–478, September 2002.

[21] M. Sipper, Physica D 92 (1996) 193

[22] M. Sipper, Evolving uniform and non-uniform cellular automata networks, in: Annual Reviews of Computational Physics, Vol. V. ed. D. Stauffer (World Scientific, 1997).

[23] J. Werfel, M. Mitchell and J. P. Crutchfield, Resource Sharing and Coevolution in Evolving Cellular Automata. IEEE Transactions on Evolutionary Computation, Vol. 4, No. 4, November 2000

[24] P Siepmann, CP Martin, N Krasnogor and P Moriarty: A genetic algorithm approach to guiding the evolution of self-organised nanostructured systems. In the proceedings of Condensed Matter and Materials Physics, Exeter, UK, April 2006

[25] D. E. Goldberg. Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems*, 5:139-168, 1991

[26] G. Sywerda. Uniform crossover in genetic algorithms. *Proceedings of the third international conference on Genetic algorithms*, pages 2–9, 1989

[27] H. Muhlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm. *Evolutionary Computation*, 1:25–49

[28] D.E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. *Urbana*, 51:61801

[29] M. Sipper, M. Tomassini, M. S. Capcarrere. *Desisgning cellular automata using a parallel evolutionary algorithm*. Nuclear Instruments and Methods in Physics Research A 389 (1997) 278-283

[30] Merz, P. *Advanced fitness landscape analysis and the performance of memetic algorithms*. *Evol. Comput.* 12, 3 (Sep. 2004), 303-325.