Using Compression to Find Interesting One Dimensional Cellular Automata

Ahmed, Nadim; Teahan, William

Complex and Intelligent Systems

DOI:
10.1007/s40747-019-00121-7

Published: 01/04/2020

Publisher's PDF, also known as Version of record

Complex & Intelligent Systems
https://doi.org/10.1007/s40747-019-00121-7

ORIGINAL ARTICLE

Using compression to find interesting one-dimensional cellular automata

Nadim Ahmed[1] · **William J. Teahan**[1]

Received: 25 January 2018 / Accepted: 29 August 2019

Abstract
This paper proposes a novel method for finding interesting behaviour in complex systems based on compression. A new clustering algorithm has been designed and applied specifically for clustering 1D elementary cellular automata behaviour using the prediction by partial matching (PPM) compression scheme, with the results gathered to find interesting behaviours. This new algorithm is then compared with other clustering algorithms in Weka and the new algorithm is found to be more effective at grouping behaviour that is visually similar in output. Using PPM compression, the rate of change of the cross-entropy with respect to time is calculated. These values are used in combination with a clustering algorithm, such as k-means, to create a new set of clusters for cellular automata. An analysis of the data in each cluster is then used to determine if a cluster can be classed as interesting. The clustering algorithm itself was able to find unusual behaviours, such as rules 167 and 181 which have output that is slightly different from all the other Sierpiński Triangle-like patterns, because their apexes are off-centre by one cell. When comparing the new algorithm with other established ones, it was discovered that the new algorithm was more effective in its ability to group interesting and unusual cellular automata behaviours together.

Keywords Machine Learning · Clustering · Interestingness · Cellular Automata · PPM Compression

Introduction

The analysis of the behaviour of complex systems is a difficult problem, not least because of the difficulties that arise due to emergence and self-organisation. Behaviour in this context means the movements and interactions of the agents and how this also affects the environment. Often, a system can be defined by simple rules, but those rules can lead to many different behaviours depending on the initial conditions, and often, the observer is never certain that they may have missed some interesting behaviour even after running many simulations or making many observations in the case of a real system. Clearly, it would be extremely useful to be able to automatically classify interesting behaviours within a complex system. However, to find such behaviours, it is important first to define what we might mean when we describe a behaviour as being interesting. Within the data mining and knowledge discovery community, there are several definitions of the term "interestingness". As there is no previous analysis of interestingness within complex systems, this paper explores whether it is possible to automatically classify when interesting behaviour has occurred within a complex system. As a case study, it will specifically use one-dimensional (1D) elementary cellular automata as an example of a complex system to find interesting behaviour within such systems.

Cellular automata were invented in the 1940s. There has been substantial research, since they are still just as relevant today in fields ranging from its use as a pseudo-random sequence generator [1] to designing levels in mazes [2]. This paper will examine 1D cellular automata which produce different patterns as a result of their behaviour depending upon the initial conditions and the rules used. In a 1D elementary cellular automata, the cell can be inactive (white) or active (black) [3] with 256 basic rules available. The activation state of the cell and the two on either side of it affects the cell on the next iteration. This paper will use cellular automata with a single activated cell in the centre of the initial row (with subsequent rows being generated each time step when

✉ Nadim Ahmed
nadim.ahmed@bangor.ac.uk

William J. Teahan
w.j.teahan@bangor.ac.uk

[1] School of Computer Science, Bangor University, Dean Street, Bangor, UK

Published online: 20 September 2019

visualising the behaviour of the system). There are many permutations of rules and initial conditions that generate many different behaviours that are difficult to explore manually.

Amongst the potential patterns available, there could be undiscovered interesting patterns concealed amongst the myriad of different initial conditions and rule permutations. Within those patterns produced, many may be referred to as "uninteresting" patterns. These uninteresting patterns include the empty spaces and diagonal lines moving from the centre to a corner, or repeated patterns of horizontal stripes. For an initial condition of a single cell, there are interesting patterns that are produced by some of the rules, such as those that look similar to a Sierpiński Triangle. Currently, there is no list of cellular automata rules that are classified by their interesting output. This paper will propose such a list, and also a new way of automatically classifying interesting cellular automata patterns based on compression.

This paper is organised as follows. It first examines the idea of "interestingness" in terms of complex system behaviour, which is the main focus of this paper, based on how it has been defined in other fields. The next section then examines the current ways that cellular automata have been classified. As this paper uses compression to classify 1D elementary cellular automata behaviour, a previous example of using compression with classifying cellular automata is discussed. The new clustering algorithm which makes use of compression is then introduced, and then used to find interesting behaviours in 1D elementary cellular automata. Finally, the new clustering algorithm is compared with other clustering algorithms available on Weka.

Interestingness

Most of the available literature on interestingness concentrates on its use in data mining and knowledge discovery. Surprisingly, as the concept of interestingness is well accepted in these fields, there is no literature for its use within complex systems. Interestingness may be defined using subjective (human controlled) or objective (statistical data) measurements [4]. Subjective measures rely on comparing previously observed behaviour with the current behaviour and noticing any changes. It can also be subjected to human emotion at the time of deciding whether something is classed as interesting. Objective measures rely on statistical analysis of the data describing the behaviour, and are independent from humans and are, therefore, generic.

A behaviour is deemed interesting if it strays from what has been previously observed [4], is often classed as something novel or surprising [5] or that interesting discoveries are surprising [6]. However, in data mining and knowledge discovery, there is an additional requirement; it has to be useful. McGarry states that a pattern may be unexpected and, therefore, could be deemed as noise or an outlier, and, thus, may not be useful [7]. Data mining also requires patterns to be relevant and useful, so their interestingness measures are designed to reduce the number of patterns that need to be checked [8]. This means that applying the same interestingness measures used within the data mining community may not be what is needed for analysis of complex systems.

As mentioned above, one of the definitions of interestingness uses a notion of surprise. A large surprise is one where an action that was predicted with high confidence did not take place, but a different action was performed instead [9].

Another definition uses Shannon's entropy to measure interestingness. Hall and Morton state that entropy estimators may be used to construct measures of interestingness [10]. Blanchard mentions that Shannon conditional entropy is one of the most commonly used measures to calculate rule interestingness [11]. Schmidhuber states that associating Shannon's entropy directly with interestingness would be incorrect, due to random noise such as white noise, resulting in a high entropy value, which would not be interesting [12]. Although if in a visual system, the output from the inputs was mainly black, and then suddenly, there was white noise, it could be classed as interesting from a surprising point of view if it was unexpected.

Schmidhuber has a different idea of what constitutes interestingness [12]. He states that when something is beautiful and newly observed, then it is interesting; however, over time, a beautiful item that has been observed many times loses its interestingness value [12]. An example is the photos of the Horsehead Nebula in the constellation of Orion. For someone who has never seen it before, it looks beautiful and interesting, but after seeing dozens of photos, the interestingness factor reduces.

Whether a behaviour of a system is interesting or not depends upon the context. As Schmidhuber states, beauty is an important factor; however, there will be some exceptions to this idea. For example, he mentioned that when a visual sensor that stays in the dark experiences white noise, then there is a sudden increase in entropy. This is due to the previous input of completely zero values suddenly changing into random values of 0s and 1s, and therefore, it becomes uncompressible. Schmidhuber states that in both of these cases, complete darkness and random white noise are boring, and, therefore, not interesting [12]. This is a good case for defining what is interesting in comparison to beauty; however, from a different point of view, the sudden appearance of white noise in what was a dark environment should be classed as interesting. It may not be beautiful, but it could be an indicator that something in the environment has changed, and is, therefore, interesting.

Keeping humans in the loop when trying to determine if something is interesting is another way for determining whether something is interesting [13]. One system that relies

on humans in the loop is the "Conceptual Knowledge Discovery in Databases" system which creates plots as output and relies on experts to test out hypotheses by inspecting the plot structures [7,14]. Another demonstration of the possible importance of having a "human in the loop" aspect to help with finding interesting behaviour is Schmidhuber's claim that beauty is important in defining interestingness. In the case of cellular automata when relying on visual observations, certain patterns may not be classed as interesting, mainly because there is no aspect of beauty within them.

Hudson [15] looks at how ease of compression determines whether a pattern is interesting or boring. His conclusion is that if compression is trivial (producing a very small output file), or is almost impossible (producing an output almost equal in size to the input file), then these situations can be considered as boring. All other situations where compression is challenging are deemed to be interesting [15]. The ease in compression indicates that very little change is occurring when compared to previously, whereas when the compression is almost impossible, then it is as though random noise like white noise is being compressed. In these two situations, they would not be classed as interesting in a general sense.

Current classifications of 1D elementary cellular automata behaviour

Cellular automata behaviour has been classified using different criteria by a number of researchers. One classification, by Wolfram [3], describes four classes:

Class 1 The pattern produced by the cellular automata eventually culminates with all cells having the same value; for example, all values become 0.

Class 2 The pattern produced by the cellular automata consists of solid simple structures or repeated patterns. Examples are vertical or diagonal lines or a ladder type of structure, otherwise repeating patterns alternating between 0 and 1.

Class 3 The pattern produced by the cellular automata is chaotic in nature; for example, patterns that grow to fill out the width of the screen, such as Sierpiński Triangle-like triangles [16].

Class 4 The pattern produced by the cellular automata is constructions of high complexity that do not disintegrate until the distant future [16]. This includes narrow structures that do not fill the whole screen width. They do not grow in width, but rather in height, although they may stop after a while.

Wolfram summarises the classes in his book, "A New Kind of Science" [17] where he states that Classes 1 and 2 will "rapidly settle down" until there is to all intents and purposes no further activity. Class 3 has cells that continually change at each step, such that they "maintain a high level of activity forever". Class 4 systems are in between Class 2 and Class 3 as the pattern produced does not die out as quickly as Class 2, but it does not have the complexity of Class 3. Wolfram also says that Class 4 systems "waver between Class 2 and Class 3 behaviour". Finally, there are also some borderline cases which can be defined as one or the other class [17].

Further classifications aim to refine Wolfram's classes [18,19]—Li has one variation having six classes [20,21], as shown in Table 1.

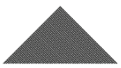
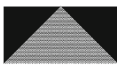
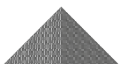
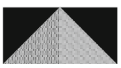
Table 1 The six classes defined by Li and Packard with their descriptions

Null	The produced pattern contains either all 0s or all 1s
Fixed-point	The pattern is formed of a single activated cell that goes down to the next line. The activated cell on the next row may be displaced to the left or the right of the above activated cell or directly below it. This includes vertical or diagonal lines with a width of a single cell
Two-cycle	This is similar to the fixed-point class as above, but the activated cell is repeated. As before, it can have a spatial shift on the proceeding row, and the main structure is repeated, either next to one another, looking thicker, or wider apart, and looking similar to a ladder structure
Periodic	The pattern is a repeated oscillation of a single rule in a vertical direction
Edge of Chaos	A persistent complex pattern is produced until eventually a state of oscillation occurs
Chaotic	The pattern produced does not oscillate, and typically rapidly digresses from the earlier behaviour within the rule, and is subjected to instability due to the jittering caused by the system

Compression and classification of 1D elementary cellular automata behaviour

This section discusses the compression of patterns created by different cellular automata rules. The idea is to find a way to cluster all cellular automata rules that produce similar output into similar groups using compression, and from this to find interesting cellular automata. The first section below looks at one previous study where cellular automata patterns were compressed and grouped according to compressed file size. The second section looks at the new method of compressing the patterns to help retrieve useful information which will group cellular automata rules that produce similar patterns together, and from this, we can determine whether a type of

Table 2 Excerpts of the classification results produced by Zenil's [22] method for 1D cellular automata

 Rule 151 1026 bytes	Rule 136 1026 bytes	Rule 128 1026 bytes	 Rule 127 1026 bytes	 Rule 119 1026 bytes	Rule 104 1026 bytes	Rule 96 1026 bytes	 Rule 95 1026 bytes
 Rule 78 2386 bytes	 Rule 252 2434 bytes	 Rule 220 2434 bytes	 Rule 79 2442 bytes	 Rule 13 2458 bytes	 Rule 238 2470 bytes	 Rule 206 2470 bytes	 Rule 69 2470 bytes
 Rule 230 2726 bytes	 Rule 225 2762 bytes	 Rule 197 2786 bytes	 Rule 246 2798 bytes	 Rule 92 2806 bytes	 Rule 190 2818 bytes	 Rule 169 2818 bytes	 Rule 147 2830 bytes
 Rule 18 3778 bytes	 Rule 22 3794 bytes	 Rule 118 3814 bytes	 Rule 145 3826 bytes	 Rule 161 3978 bytes			

Each row represents the rules that are classified as the same in Zenil's LZW classification, even though they may look different. They have been placed in the same order as given by Zenil's compression-based classification

pattern is interesting depending on which group a rule falls into.

Previous use of compression with cellular automata

There has been one previous study by Zenil [22] using compression to investigate dynamical properties of cellular automata, using a compression technique that is comparable to the LZW algorithm. The rules are arranged in order from the smallest compressed file size to the largest. The ordering of the rules shows a clustering of the different types of cellular automata behaviour, which Zenil states agrees with Wolfram's four classes [22].

His method is able to separate Wolfram classes 1 and 2 from classes 3 and 4 using the compressed file size alone. The problem with this technique, however, is that even though the ordering agrees with Wolfram's classes, the output produced by rules with similar compression file sizes often looks visually different. There are cases when either there are no visual similarities between behaviour produced by the rules that are placed together, or the similarities are vague. Table 2 shows excerpts from this ordering which shows the output from the rules, rule number, and the compressed file size in bytes. The ordering is the same as shown in Zenil [22]. The table clearly shows a number of examples that are placed together that are not visually similar such as Rule 225 next to Rule 197, and Rule 190 next to Rule 169. The latter pair even has exactly the same compression size, even though they visually look very different.

A new approach to clustering cellular automata behaviour using compression

This section describes a new approach to clustering cellular automata behaviour using compression. The goal is to make it easier to group visually similar clusters of cellular automata rules, to find interesting behaviours. Algorithm 1 is a summary of the tasks required for clustering cellular automata output using our method. The first task is to calculate the *divergence* in cross-entropy $\Delta H(r)$ for each row r for each rule. This is done using the codelength values produced when compressing each symbol that has been output using the Prediction by Partial Matching (PPM) text compression scheme (line 2). The second part of the algorithm is a loop that creates different numbers of clusters (line 3). The second main task is applying a clustering algorithm to create different numbers of clusters (line 4). The final line calculates the Silhouette value as described below, for the current clustering, so the optimal number of clusters may be selected at the end by choosing the lowest Silhouette value.

Information divergence, like compression, has been found to be a useful means for comparing changes in state or behaviour, and therefore, we have investigated it for this paper. A further explanation of each aspect of this algorithm will now be discussed in the next few sections. The next section explains how the PPM algorithm works and the particular variant of the algorithm and implementation we adopted for our experimental evaluation. Section 5.2 then discusses how we adapted PPM to specifically compress the 1D cellular automata output.

input : Cellular automata represented as a series of 0 and 1 to represent inactive and active cells, respectively. Each rule is stored in an individual file.
output: Clusters of cellular automata arranged into similar looking behaviour.
foreach cellular automata rule **do**
  `Create` $\Delta H(r)$ values `using PPM compression`
for numClusters = *1* **to** *20* **do**
  `Create` numClusters clusters `by applying a Clustering Algorithm` on $\Delta H(r)$ values
  `Calculate` Silhouette value for current set of clusters

Algorithm 1: Overview of the new clustering system for cellular automata.

Prediction by partial matching

The Tawa Toolkit [23] based on the earlier Text Mining Toolkit [24] was used which implements the Prediction by Partial Matching (PPM) text compression algorithm, which is regarded as one of the most effective text compression algorithms available [25]. The other reason for using PPM is that it supports streaming compression, and that it is possible to note the progress of the compression as the compression proceeds.

PPM is an example of an adaptive statistical-based compression system, which has the characteristics of carrying out two processes: modelling and coding. The model builds a table of probabilities of all symbols encountered so far, and uses that to predict what the next symbol to appear will be. The compressor encodes the actual symbol using the probability distribution produced by the model. It uses a Markov-based approach where the last few characters in the input stream (called the context) are used to predict what the next character will be. The number of characters used in the context is used to define the order of the model. For example, a context of length 1 is used for an order 1 model. It has been observed that orders higher than 5 often produce worse compression in many applications.

As each symbol is encoded, probability distribution models for each order are created. These models are then combined into a single one using the escape mechanism, which predicts the upcoming character using a highest fixed order first, but backs off to lower order models when the upcoming character has not been seen yet. For this paper, a maximum fixed order of 6 was used.

The compression codelength h for encoding a symbol s_i in bits using an order 5 PPM model can be represented by Eq. 1:

$$h(s_i) = -\log_2 P(s_i|s_{i-5}s_{i-4}\ldots s_{i-1}). \tag{1}$$

PPM normally starts with the highest order, k, as requested by the user. When a new symbol is observed that has not been seen before, an escape symbol is issued; the escape sequence tells PPM to reduce the order by 1 until the symbol is no longer novel. However, if the order k reaches -1, then the same probability of $\frac{1}{|A|}$ is assigned to all characters, with A representing the size of the alphabet. The effect of the escape mechanism is to "smooth" the probability estimates. PPM also uses the technique of "full exclusion" where when the escape mechanism is used, all the symbols that were already predicted by higher orders are excluded.

Another notable difference of PPM compared to other language modelling methods is that of "update exclusions" which was introduced by Moffat [26]. This technique deals with the way which the counts are updated for the context. When update exclusions are enabled, the symbol count for each context is only incremented if it has not been predicted by a higher order context.

Two prominent variants to PPM have been crafted that use different methods for calculating the escape probabilities, method C and method D. (These are called PPMC and PPMD in the literature). Two equations are used for calculating the prediction probabilities, one for calculating the escape probability, e, and one to calculate the probability of a symbol occurring, $p(s)$. The equations for PPMC are shown in Eqs. 2 and 3 and the equations for PPMD in Eqs. 4 and 5:

$$e_{\text{PPMC}} = \frac{t}{n+t} \tag{2}$$

$$p(s)_{\text{PPMC}} = \frac{c(s)}{n+t} \tag{3}$$

$$e_{\text{PPMD}} = \frac{t}{2n} \tag{4}$$

$$p(s)_{\text{PPMD}} = \frac{2c(s)-1}{2n}, \tag{5}$$

where, t is the number of types that follows the context; n is the number of times a context has occurred; and $c(s)$ is the number of times a context was followed by the symbol s.

PPMC estimates the probability for each symbol using its raw frequency and assigns the number of types t to the escape count for estimating the probability of an escape occurring when an upcoming symbol is unseen in the context. In contrast, PPMD increments the symbol count by 2 when a previously seen symbol is encountered, but increments the escape count by 1 and assigns an initial symbol count of 1 for symbols that have not been seen before in the context. It has been found that PPMD outperforms PPMC in most compression experiments, and therefore, this paper uses PPMD

Table 3 The mean file sizes in bytes for the compressed 1D cellular automata output

Order	PPMC	PPMD
3	571.0677	565.9692
4	519.4867	513.0579
5	504.1639	496.2098
6	486.7357	**477.0722**
7	490.0108	478.5286
8	491.2515	477.7705
9	498.7499	483.6288
10	505.0899	489.0198

The input file size for each rule is 33,282 bytes
Bold indicates the best compression

to compress the data produced from the elementary cellular automata output.

Compressing 1D cellular automata output

A 1D elementary cellular automata with a width of 257 cells and a total of 130 rows including the initial condition of a single activated cell in the middle of the first row was used in our experimental analysis. The output behaviour for each rule was saved into a file as a stream of 0s and 1s terminating each row with a newline character. Each row has a total of 258 characters, including the newline character. The incremental PPM compression size was collected for each row of the 1D cellular automata and the differences in compression were used to find interesting behaviour.

As mentioned in the previous section, PPM has two main types, C and D, with each being able to use different orders for calculating the maximum context size. The data produced by all the 256 cellular automata rules have been collated, and the total compressed file sizes, memory used, and the time taken have been compiled. The method and order that produces the smallest compressed file size was determined to use in our experiments.

Table 3 shows the mean compressed file sizes for all 256 1D cellular automata rules when using PPM with both methods and different orders. This shows that overall, PPMD gives smaller compressed file sizes than PPMC. The file sizes reduce as one increases the order size, although the file sizes increase from order 7 onwards. The smallest file size is achieved using PPMD with order 6.

Table 4 shows the mean amount of memory used by each of the order sizes in bits for each cellular automata rule; there is no difference in memory usage with either method. The memory requirements are small, being between 16.3 and 17.3 KB per rule. Table 5 shows the mean number of seconds, across 10 runs, for calculating the divergence data for all 256 rules on an Intel Core i5-2400 CPU at 3.10 GHz. The results show that the difference in time taken between different orders and methods is almost negligible.

Table 4 Mean memory usage in bits used by the different orders for each cellular automata rule

Order	Bits
3	133,894
4	134,300
5	134,819
6	135,484
7	136,353
8	137,540
9	139,234
10	141,743

Table 5 The run time in seconds for calculating the divergence values

Order	PPMC	PPMD
3	35.0972	34.3036
4	35.0772	34.9584
5	35.3324	34.9992
6	35.7592	35.4992
7	35.9908	35.8408
8	35.8440	35.9024
9	35.7248	35.9088
10	35.3480	35.0196

To better illustrate how PPM works using a relevant example and also to illustrate some important aspects of the algorithm, Table 6 shows a dump of the PPM model after reading the first ten lines of output produced for rule 18 (as shown in Fig. 1). The model has been stored in a trie data structure with a maximum depth of 7 (since an order 6 model is being used) which stores the suffixes contained in the cellular automata output data. Each set of columns separated by the vertical bar in the table contains information concerning the nodes in the trie; the nodes are arranged using a preorder traversal. There are separate columns for the node counts, the first being for PPMC without update exclusions (which we have labelled as PPMC′ as normally PPMC would perform update exclusions), while the second set of counts are for standard PPMD with update exclusions. The column labelled 'Path to trie node' shows the path down the trie to each node and the column labelled 'Depth' indicates the depth of each node in the trie. As stated earlier, the cellular automata output produces three output symbols, '0', '1', and a newline character, as indicated in the table by the letter 'n' in the trie path.

Table 6 shows a clear difference in the counts that are stored for PPMC′ and PPMD. The PPMC′ counts accurately reflect the raw counts. For example (in the first ten lines of the output for rule 18), referring to the counts in the second, sixth, and tenth columns: the number of times any symbol occurs is 2580 times (at depth 0 of the trie); '000001' occurs 14 times; '00n00' occurs 9 times (i.e. a '00' ends one row and also starts the next row after it); '100' occurs 18 times; 'n00'

Table 6 Dump of dynamic PPM trie for the first ten lines of the 1D cellular automata rule 18 and the counts for both order 6 PPM models PPMC′ (PPM using escape method C without update exclusions) and PPMD (PPM using escape method D with update exclusions, i.e., standard PPMD)

Depth	PPMC′ counts	PPMD counts	Path to trie node	Depth	PPMC′ counts	PPMD counts	Path to trie node	Depth	PPMC′ counts	PPMD counts	Path to trie node
0	2580	8		3	18	1	001	4	9	1	0n00
1	2537	7	0	4	18	1	0010	5	9	1	0n000
2	2494	7	00	5	11	1	00100	6	9	1	0n0000
3	2466	7	000	6	11	1	001000	7	9	17	0n00000
4	2438	7	0000	7	7	13	0010000	1	33	1	1
5	2414	7	00000	7	4	7	0010001	2	33	1	10
6	2390	7	000000	5	7	1	00101	3	18	1	100
7	2367	4733	0000000	6	7	1	001010	4	18	1	1000
7	13	25	0000001	7	5	9	0010100	5	14	1	10000
7	9	17	000000n	7	2	3	0010101	6	14	1	100000
6	14	3	000001	3	9	1	00n	7	13	25	1000000
7	14	27	0000010	4	9	1	00n0	7	1	1	1000001
6	9	1	00000n	5	9	1	00n00	5	4	1	10001
7	9	17	00000n0	6	9	1	00n000	6	4	1	100010
5	14	1	00001	7	9	17	00n0000	7	4	7	1000100
6	14	1	000010	2	33	3	01	3	15	1	101
7	7	13	0000100	3	33	3	010	4	15	1	1010
7	7	13	0000101	4	18	3	0100	5	7	1	10100
5	9	1	0000n	5	18	3	01000	6	7	1	101000
6	9	1	0000n0	6	14	3	010000	7	7	13	1010000
7	9	17	0000n00	7	14	27	0100000	5	8	1	10101
4	18	3	0001	6	4	1	010001	6	8	1	101010
5	18	3	00010	7	4	7	0100010	7	6	11	1010101
6	11	3	000100	4	15	3	0101	7	2	3	1010100
7	11	21	0001000	5	15	3	01010	1	9	1	n
6	7	1	000101	6	7	3	010100	2	9	1	n0
7	7	13	0001010	7	7	13	0101000	3	9	1	n00
4	9	1	000n	6	8	3	010101	4	9	1	n000
5	9	1	000n0	7	8	15	0101010	5	9	1	n0000
6	9	1	000n00	2	9	1	0n	6	9	1	n00000
7	9	17	000n000	3	9	1	0n0	7	9	17	n000000

The new line character is indicated by 'n'

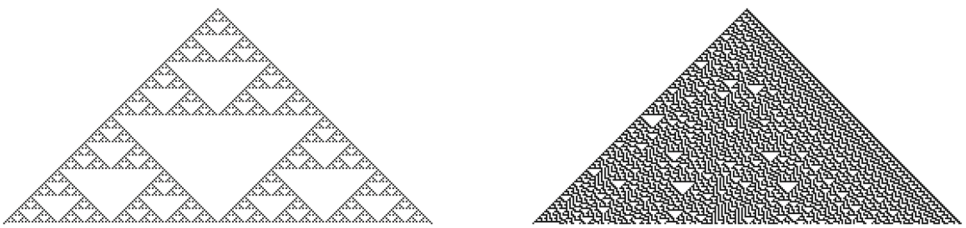


Fig. 1 1D elementary cellular automata output for Rules 018 and 086 with single activated cell in the centre of the initial row run for 128 iterations (i.e., with 128 rows). The $\Delta H(r)$ plots for these rules are shown in Figs. 2 and 3

occurs 9 times (i.e., a row starts with '00'); and '1n' never occurs (meaning none of the first 10 rows end with a '1'). In contrast, the PPMD counts are markedly different—most of the PPMD counts for nodes with the longest paths are twice the equivalent PPMC′ counts −1, as per Eq. 5. For nodes with shorter paths, the counts are much less than the PPMC′ counts due to the way which the update exclusion mechanism works (which, as stated, only updates count for the longest

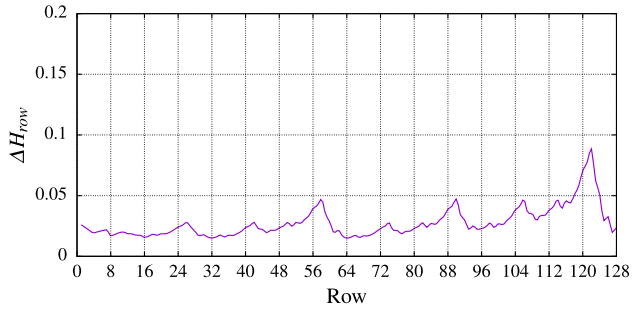


Fig. 2 Plot for Rule 18 showing $\Delta H(r)$ for the cellular automata output

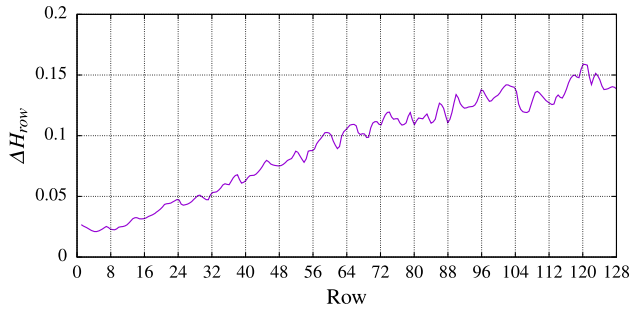


Fig. 3 Plot for Rule 86 showing $\Delta H(r)$ for the cellular automata output

context if the symbol being predicted in that context has not already been seen).

The *cross-entropy* $H(r)$ for each row r was calculated by dividing the summation of the codelengths for encoding each symbol in the row r by the number of characters c in each row, as shown in Eq. 6, where $c = 258$ in this case. The *divergence* in cross-entropy $\Delta H(r)$ was calculated by subtracting the cross-entropy of the current row from the cross-entropy of the previous row as defined by the following Eq. 7:

$$H(r) = \frac{1}{c} \sum_{i=cr}^{c(r+1)-1} h_i(r) \tag{6}$$

$$\Delta H(r) = H(r) - H(r-1). \tag{7}$$

The divergence in cross-entropy was plotted against the row number for each rule. For example, Fig. 1 shows the output for Rules 018 and 086 with a single activated cell in the centre of the initial row when it is run for 129 iterations (i.e., with 129 rows). Figures 2, 3 plot the divergence in cross-entropy $\Delta H(r)$ values for each row for these two rules, respectively. The first plot shows a repeating pattern of increasing peaks of changes in cross-entropy that clearly corresponds to the pattern of triangles exhibited by the rule in the output for Rule 18 to the left of Fig. 1. The second plot reflects the increasing chaotic texture of triangles in the output for Rule 86 to the right of Fig. 1.

After all the plots of each rule were created, it was noted that different rules produced similar plot shapes for the plots and that there were only a small number of distinct plot shapes (around 10 or so). This is perhaps to be expected as a visual inspection of the output from 1D elementary cellular automata with a single central cell set initially will confirm that there are about a similar number of re-occurring patterns of behaviour.

Clustering 1D cellular automata output

The first part to clustering is to finding out the optimal number of clusters, so the next section will look at how to find the optimal number of clusters. Different clustering algorithms are then compared to see which ones produce an optimal number of clusters. The final part of this section will examine the clusters that were produced by the chosen clustering algorithm.

Finding an optimal number of clusters

When deciding on how many clusters are required, there are several techniques available such as Pairwise Precision and Recall, Matching Index, and Rand Index which rely on a gold standard [27]. For our 1D cellular automata case study, there is no gold standard available as we wish to cluster the output without any preconceived ideas to eliminate the potential of bias. However, there are several other methods available such as the elbow method and using the Silhouette value [27] that do not require a gold standard.

Concerning the latter, Eqs. 8, 9, and 10 describe how to calculate the Silhouette value. This involves two parts to be calculated, $a(x)$ and $b(x)$, where $a(x_i)$ is the average distance between object x_i and all the other objects in the current cluster. The second part, $b(x_i)$ calculates the average distance from the object x_i to the next nearest cluster. The Silhouette value $sil(x_i)$ is calculated using Eq. 10. Assume that the average distance from x_i to the objects in the current cluster is a value of 7 ($a(x_i) = 7$), and that the average distance from x_i to the objects in the next nearest cluster is 5 ($b(x_i) = 5$). When subtracting $a(x_i)$ from $b(x_i)$ as shown in the numerator of Eq. 10, a negative value is given, indicating that x_i is in the incorrect cluster. The denominator chooses the larger value between $a(x_i)$ and $b(x_i)$:

$$a(x_i) = \frac{1}{n_{C_A} - 1} \sum_{x_j \in C_A, x_j \neq x_i} d(x_i, x_j) \tag{8}$$

$$b(x_i) = \min_{C_B \neq C_A} \frac{1}{n_{C_B}} \sum_{x_j \in C_B} d(x_i, x_j) \tag{9}$$

$$\text{sil}(x_i) = \frac{b(x_i) - a(x_i)}{\max(a(x_i), b(x_i))}. \tag{10}$$

The Silhouette value ranges from -1 to $+1$. When the Silhouette value of an item is close to $+1$, then it is con-

sidered to be "well clustered". When the value is 0, then it does not matter which cluster which it belongs to, although a zero value is also used when an item is alone in a cluster. When the value is close to -1, then that indicates that the item is misclassified and should be in a different cluster [28]. Rousseeuw [28] states that to choose the optimal number of clusters, the overall average Silhouette width $\overline{s(k)}$ has to be calculated by taking all the Silhouette values for all the different clusters, and calculating the average value. The number of clusters is chosen as the one with the higher $\overline{s(k)}$ value.

Having defined a method for calculating whether a clustering algorithm gives good results, we will now look at a clustering algorithm to cluster the cellular automata output.

Comparing clustering algorithms

Different clustering algorithms were used to cluster the cellular automata behaviour using the divergence in cross-entropy data using two different toolkits: `scikit-learn` [29] and Weka [30]. These toolkits provide suites of different clustering algorithms, which make it easier to choose the best for clustering 1D cellular automata behaviours. The algorithms featured in `scikit-learn` will be covered first, followed by the algorithms in Weka.

`scikit-learn` is a programming library for Python designed for machine learning, and amongst the tools it provides is a collection of clustering algorithms [29]. We used various clustering algorithms with the PPM compression divergence data with results described below.

The BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [31] clustering algorithm uses hierarchies and is designed for very large data sets. The BIRCH algorithm takes in two parameters, the number of clusters and the threshold value. A range of thresholds were set for this algorithm, because the default setting meant that no clusters were created. It was discovered that a very small value was required to increase the likelihood of clusters being populated. This achieved the highest Silhouette value of 0.8296 when 14 clusters were created with a threshold value of 0.02.

The DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [32] clustering algorithm uses density as its method of clustering by checking for the minimum number of points at a certain radius, so the density of points has to exceed a certain threshold [32]. In `scikit-learn`, the maximum distance that can exist between two points grouped as being in the same cluster can be set; the difference in this value determines how many clusters are created. The highest Silhouette average was 0.7287 for three clusters.

Hierarchical agglomerative clustering (HAC) is also based on hierarchies. Each object is kept as its own cluster and then using a distance measure, the closest objects are brought together to form a bigger cluster. The selected Silhouette average was the same as BIRCH, 0.8296 with 14 clusters.

Table 7 Summary of clustering algorithms showing the highest Silhouette average with the number of clusters which they represent

Clustering algorithm	Silhouette average	No. of clusters
Birch	0.8296	14
DBScan	0.7287	3
HAC	0.8296	14
K-means	0.8296	14
Mean shift	0.8094	21
Spectral	0.7138	4

The algorithm and the reasons for choosing the number of clusters are described in detail in the next section.

A well-known clustering algorithm is k-means clustering [33] that has been implemented for many machine learning clustering tools [34]. This algorithm requests only one parameter, the number of clusters required, and then, it will create that many clusters by arranging the values with the nearest mean values together. As with the BIRCH and HAC algorithms, we found that the highest Silhouette average was 0.8296 with 14 clusters being created.

Mean shift clustering [35] is an iterative algorithm that works by shifting the data points to the mean of the data points in its vicinity. One of its parameters is the bandwidth which is estimated using an estimator function that takes in a quantile value and the number of samples required. By changing the quantile value, the number of clusters created changes, so the highest Silhouette average was 0.8094 when 21 clusters were created.

Spectral clustering [36,37] creates eigenvectors from a matrix derived from the similarity measures within the data. `scikit-learn`'s documentation states that spectral clustering works better when the number of clusters specified is small [38]. The highest Silhouette average was 0.7138 when four clusters were created.

A summary of the clustering algorithm results is given in Table 7. Mean shift clustering was not chosen, because it created too many clusters, and was too fine-grained, so rules that would be counted as being similar were split into separate clusters, as shown in Table 8. The clusters produced by BIRCH, HAC ,and k-means were the same and gave the highest Silhouette average overall. The next section describes choosing the specific linkage and distance measure for HAC in more detail to illustrate important aspects of the clustering process.

Hierarchical agglomerative clustering

Hierarchical agglomerative clustering is an example of hierarchical clustering, where the clustering occurs in a series of steps of cluster sizes starting with a single cluster containing all the elements to n clusters with each cluster containing a

Table 8 Some of the clusters produced by the mean shift clustering algorithm were too fine-grained

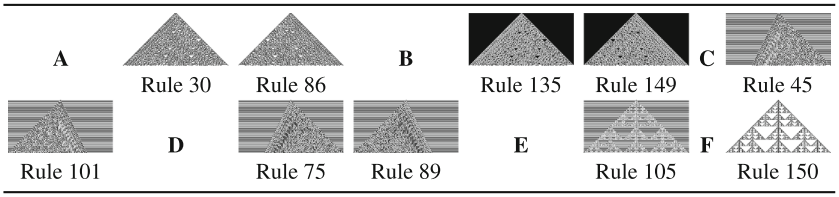
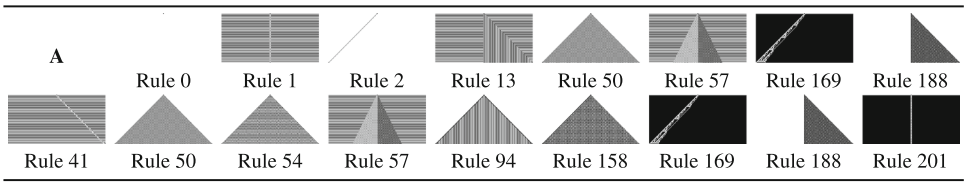


Table 9 Results of performing hierarchical agglomerative clustering with different linkages and distance measures

Distance measure	Linkage type	No. of clusters	Silhouette average	Distance measure	Linkage type	No. of clusters	Silhouette average
Canberra	Single	14	0.8296	Chebyshev	Single	9	0.8540
Canberra	Complete	14	0.8296	Chebyshev	Complete	3	0.8555
Canberra	Average	14	0.8296	Chebyshev	Average	3	0.8646
Canberra	Weighted	14	0.8296	Chebyshev	Weighted	3	0.8574
Euclidean	Single	2	0.8328	Manhattan	Single	2	0.8350
Euclidean	Complete	2	0.8463	Manhattan	Complete	2	0.8509
Euclidean	Average	2	0.8463	Manhattan	Average	2	0.8509
Euclidean	Weighted	2	0.8463	Manhattan	Weighted	2	0.8509

Table 10 Partial list of the 220 rules in one of the clusters produced using Chebyshev distance measure



single element. Agglomerative clustering starts with n clusters, which joins up the clusters to eventually become a single individual cluster, while divisive clustering goes in the opposite direction [39].

Agglomerative clustering can use different methods of "linkages", such as single, complete, average, and weighted. To use agglomerative clustering, a method of calculating the distance between objects needs to be established, and there are different distance measures that may be used, such as Canberra, Chebyshev, Euclidean, or Manhattan. Table 9 shows the results of trying out different types of linkages and distance metrics and the number of clusters that gives the highest Silhouette average with the 1D cellular automata data.

Table 9 shows that Chebyshev produces the highest Silhouette value for nine clusters with a Silhouette score of 0.8540. However, one of the clusters has 220 rules, which is filled with several different patterns (see Table 10): plain backgrounds, diagonal lines, simple triangles, and others. For this reason, it has been discounted as a suitable distance measure. The next highest Silhouette value is when using the Canberra distance measure, with a Silhouette score of 0.8296 when there are 14 clusters (see Fig. 4). Initial observations indicate that it has a better assignment of behaviours into

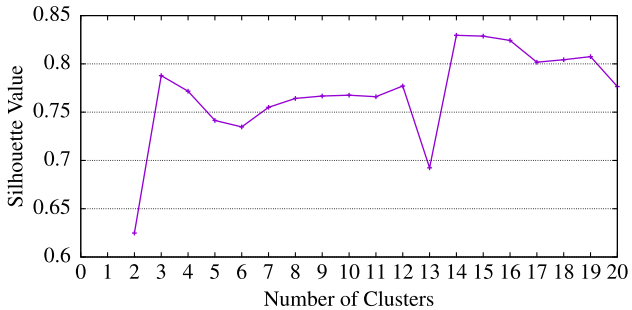


Fig. 4 Silhouette error plot using the Canberra measure

clusters. Note that when there are 15 clusters, then the Silhouette value is slightly different at 0.8287, also with a good cluster assignment.

The equation for the Canberra distance metric is shown in Eq. 11, where there are two objects being measured a and b; for example, a could contain the divergence data for rule 18, while b could contain the data for rule 22. Canberra distance is known to be very sensitive in detecting differences when the values involved are close to zero [40]. This is an important factor, since many of the values in our data are very close to zero:

Table 11 The $\Delta H(r)$ plots for cellular automata behaviours clustered using PPM compression data with a clustering algorithm

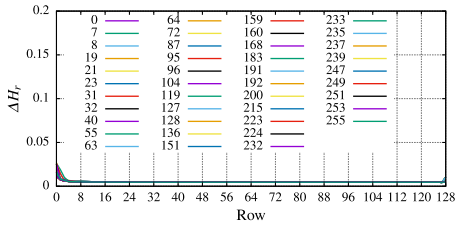
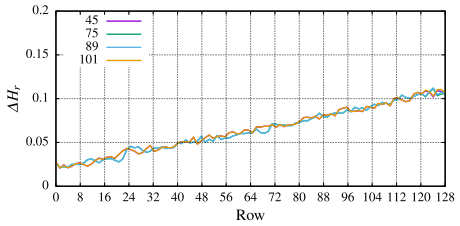
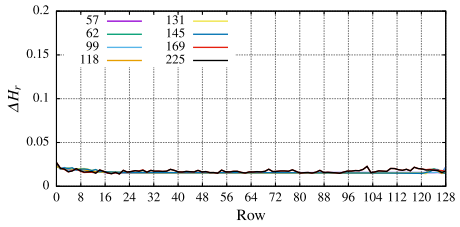
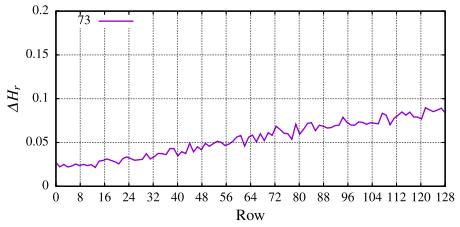
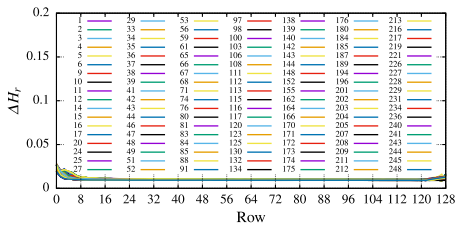
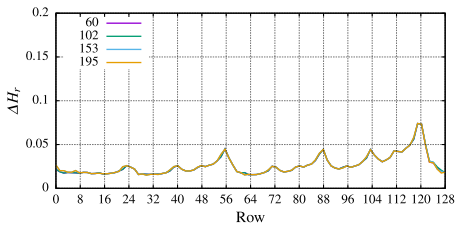
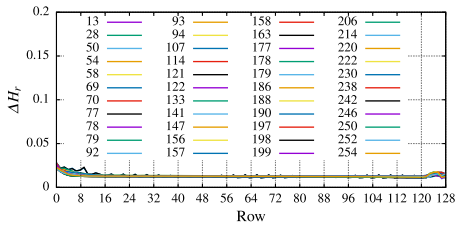
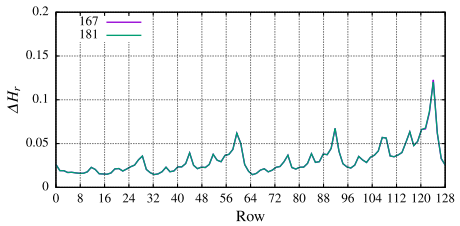
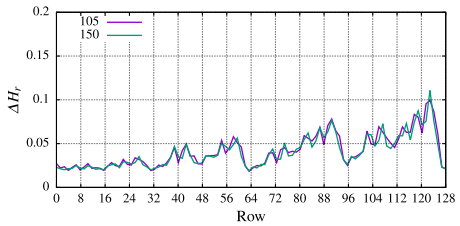
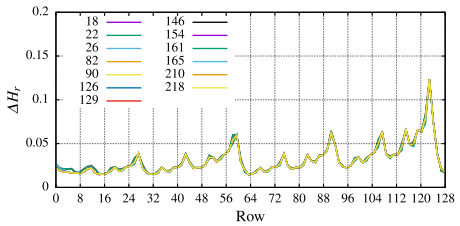
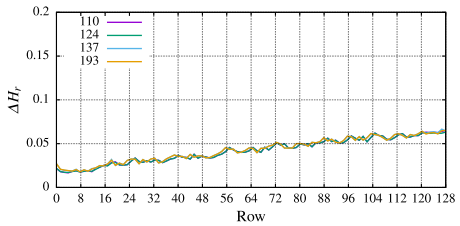
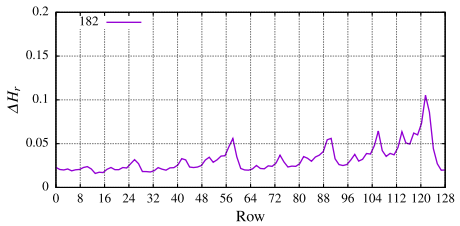
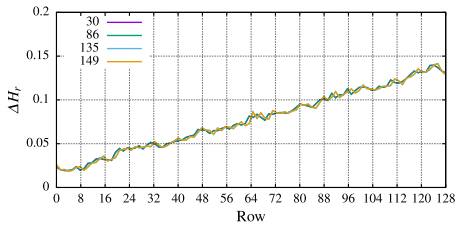
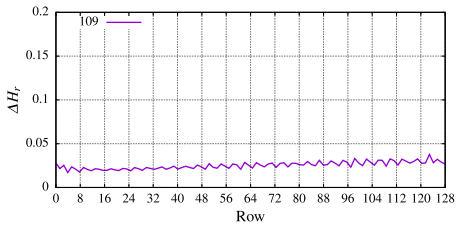
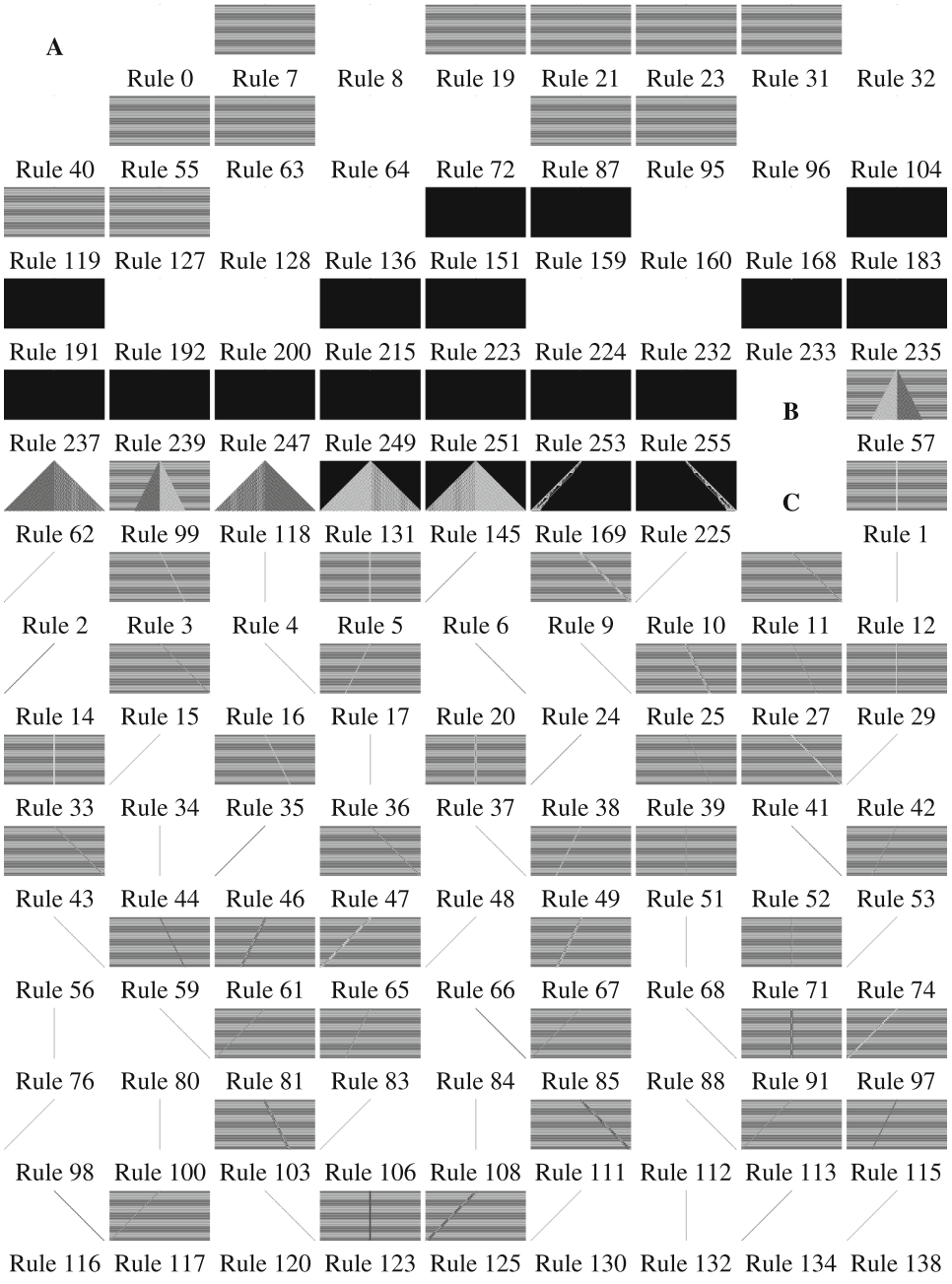
Cluster	$\Delta H(r)$ Plot	Cluster	$\Delta H(r)$ Plot
A		H	
B		I	
C		J	
D		K	
E		L	
F		M	
G		N	

Table 12 The clusters produced using the PPM clustering algorithm for 1D cellular automata rules



$$d(a,b) = \sum_{i=1}^{n} \frac{|a_i - b_i|}{|a_i| + |b_i|}. \tag{11}$$

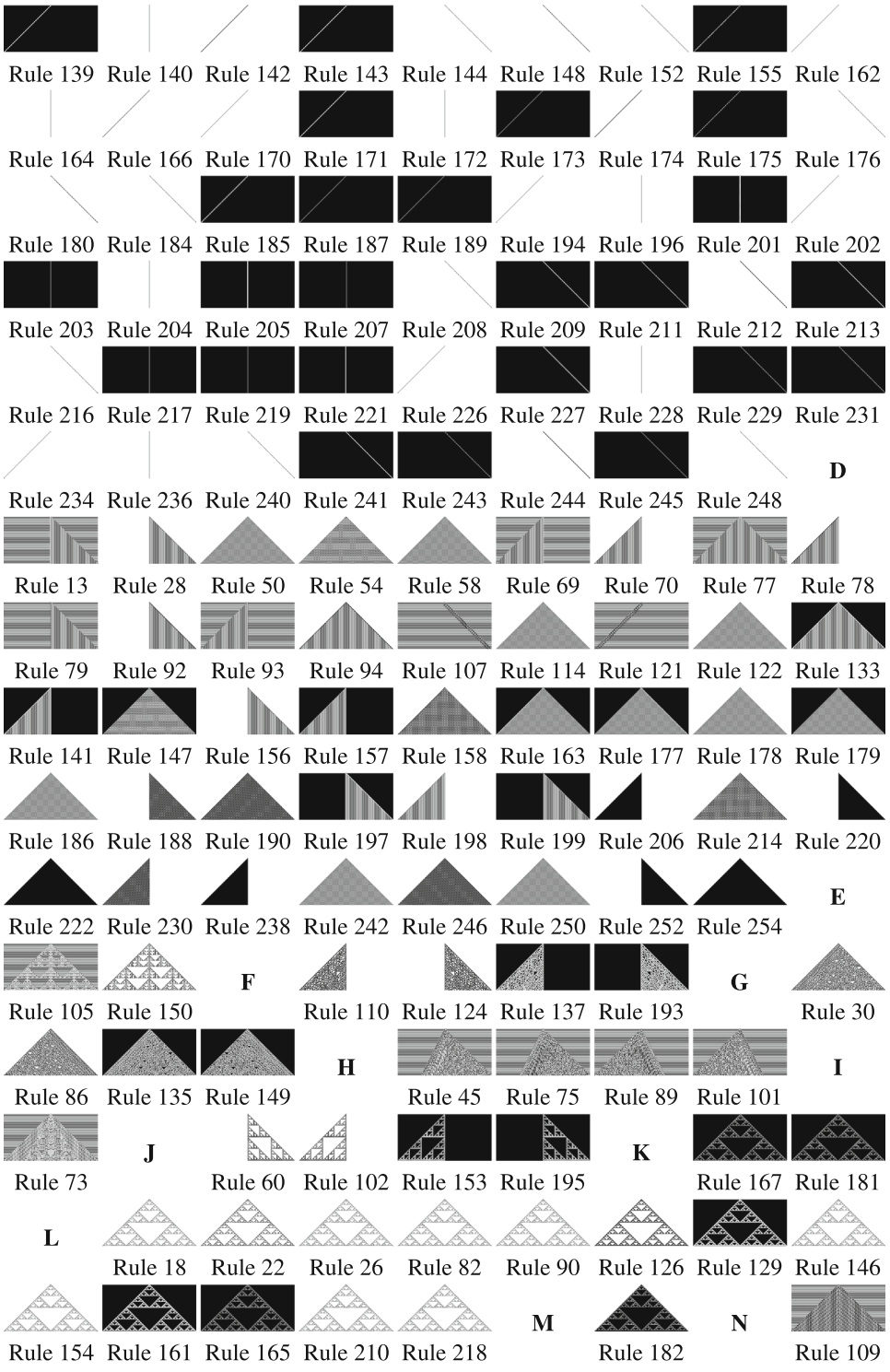
To see how the Silhouette average changes according to the number of clusters when using the Canberra measure, Fig. 4 shows the value of $\overline{s(k)}$ for various numbers of clusters. The graph indicates that the optimal number of clusters to use in our case is 14 or 15, as these have the highest overall average Silhouette widths.

Experimental results using PPM clustering

The divergence plots of the 1D cellular automata rules collated for each cluster produced by our PPM clustering algorithm are shown in Table 11. The table shows that each cluster contains rules that produce similar looking plots in most cases and also that the plot shapes compared to those for the other clusters are notably different to each other.

We can also confirm how well the clustering has worked by examining visually the output of the individual cellular

Table 12 continued



automata rules as this is a manifestation of the behaviour and compare whether the rules in each cluster are visually similar. Table 12 lists the outputs of the rules within each cluster as given by the PPM clustering algorithm with 14 clusters.

A description of the rules in each cluster is discussed below.

Cluster A consists of rules that produce three distinct types of patterns:

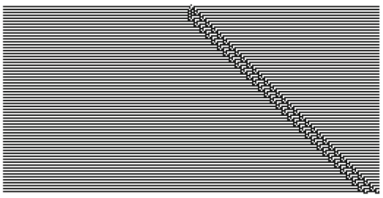


Fig. 5 Rule 107 has been placed in Cluster D

1. the output is uniformly black;
2. the output is uniformly white; or
3. the output consists of black and white horizontal stripes.

Cluster B consists of cellular automata rules that produces triangles that have two different textures split down the centre of the triangle. However, it also contains two rules that produces an output that consists of an unusual diagonal of lines and small triangles which are increasing in width.

Cluster C has many cellular automata rules that produce either vertical or diagonal lines on various backgrounds from cluster A.

Cluster D consists of cellular automata that produce triangles that have either a regular textured pattern, or a plain black texture. However, there are two exceptions where the rules produce an output of a thick-textured double-diagonal line (107; shown in Fig. 5; and 121).

Cluster E consists of rules that produce triangles that look vaguely similar to those in cluster L; however, the pattern produced is different to those in cluster L. For example, cluster L has a large triangular-shaped empty space in the middle, with smaller empty triangles above it to the top. In cluster E, the largest triangles are those which are the same size as the second largest ones in cluster L.

Cluster F contains four cellular automata rules that produces a right-angled triangle with a texture consisting of many small triangular shapes that resemble chaotic fish scales, not too dissimilar to those in cluster G.

Cluster G consists of cellular automata that produce a single triangle that has a texture of many small triangular shapes that looks similar to chaotic fish scales with a plain background of either black or white.

Cluster H consists of a horizontally striped background with a single scalene triangle that has a chaotic texture.

Cluster I consists of a triangle that contains a complex internal structure on a black and white horizontal background, as shown in Fig. 6. There are no other rules that produce a pattern like this.

Clusters J–M are based around Sierpiński-like triangles. The rules in cluster J have right-angled Sierpiński-like triangles on plain background. Cluster K has two Sierpiński-like triangles that are off-centre, and, therefore, are different from the others. Cluster L contains Sierpiński-like triangles on plain backgrounds. Cluster M contains one cellular automata rule with a Sierpiński-like triangle which is internally black on a white background.

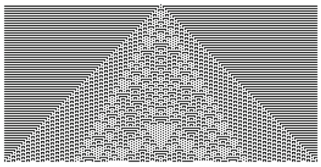


Fig. 6 Cluster I contains one rule, 73

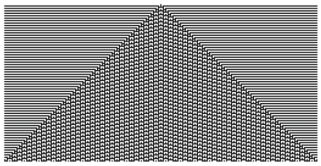
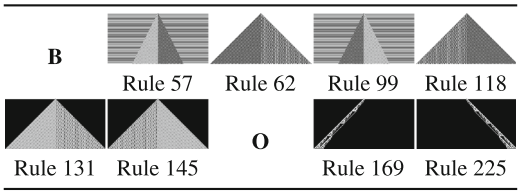


Fig. 7 Cluster N contains one rule, 109

Table 13 The changes in the clusters when 15 clusters are used instead of 14



Cluster N consists of a triangle with a horizontally striped background, but with the triangle made up of many small white squares with each square having a black dot in the middle, as shown in Fig. 7. This pattern is unique, since no other rules produce anything similar to it.

It was mentioned earlier (in Sect. 5.3.3) that the Silhouette scores for 14 and 15 clusters were very similar, with a difference of only 0.0009. When 15 clusters are created, cluster B bisects, with rules 169 and 225 being diverted into a separate cluster of their own, as shown in Table 13; referred to as cluster O in this paper. This, on visual inspection, provides a better cluster separation as these two rules produce interesting behaviour that is distinct from any other rules. Aman Ahuja [41] has argued that visual inspection often provides a better method for determining the number of clusters as opposed to using just the Silhouette score. Therefore, in this case, we argue that using the 15 clusters generated by the clustering algorithms is the best choice. This is because it produces clusters of cellular automata rules that are more fine grained and better grouped than the ones defined by Wolfram or Zenil.

Table 14 Mean values of the analysis of the $\Delta H(r)$ values for each cluster

Criteria	A	B	C	C	C	C	C	D
Cluster	Variance	Number of peaks	Level slope	Gentle slope	Moderate slope	Strong slope	Steep slope	Cluster size
A	0.0	0.0	126.5	1.4	0.1	0.0	0.0	42
B	0.0	0.0	125.3	2.7	0.0	0.0	0.0	6
C	0.0	0.0	125.5	2.5	0.0	0.0	0.0	126
D	0.0	0.0	122.8	5.2	0.0	0.0	0.0	44
E	**0.0003**	**8.0**	40.5	58.0	19.5	**10.0**	0.0	**2**
F	**0.0002**	0.0	69.8	58.3	0.0	0.0	0.0	4
G	**0.0012**	0.0	59.0	69.0	0.0	0.0	0.0	4
H	**0.0007**	0.0	71.5	56.0	0.5	0.0	0.0	4
I	**0.0004**	0.0	59.0	59.0	10.0	0.0	0.0	**1**
J	**0.0001**	**2.0**	84.0	40.0	3.0	**1.0**	0.0	4
K	**0.0003**	**8.0**	47.5	61.0	13.0	**6.5**	0.0	**2**
L	**0.0003**	**8.0**	47.2	61.2	11.6	**8.1**	0.0	13
M	**0.0002**	**6.0**	52.0	61.0	11.0	**4.0**	0.0	**1**
N	0.0	0.0	64.0	64.0	0.0	0.0	0.0	**1**
O	0.0	0.0	112.0	16.0	0.0	0.0	0.0	**2**

Values that make a cluster interesting are in bold
The criteria are explained in Sect. 6

Clustering interesting elementary cellular automata

To find any interesting behaviours and anomalies, further investigation of the data and also that produced by the PPM clustering algorithm was carried out. As mentioned earlier, Hudson [15] has stated that easily compressed files are not interesting, which implies that those that have very low $\Delta H(r)$ values that remain low would be classed as uninteresting. Therefore, interesting behaviours would be identified by sharp or sudden changes in the $\Delta H(r)$ divergence values.

The following criteria, which refer to the plots in Table 11, have been used to establish whether a cluster produces interesting behaviour or not.

(a) There should be significant variance in the divergence data (high-variance criterion).
(b) The divergence values produced should have significant peaks in the graph (frequent jaggedness criterion).
(c) There should be some gradients in the graph that can be considered to be strong or steep (strong gradients criterion).
(d) If the patterns produced are rare or unusual, even if they do not produce significant graphs, then they may be considered to be interesting (small cluster criterion).

The investigation described below uses these criteria to identify behaviours as being interesting in Table 14. The top row indicates which criterion the column describes. With criterion C, all types of gradients are shown; however, the columns that indicate whether a behaviour can be classed as interesting are the "Strong" and "Steep" slopes.

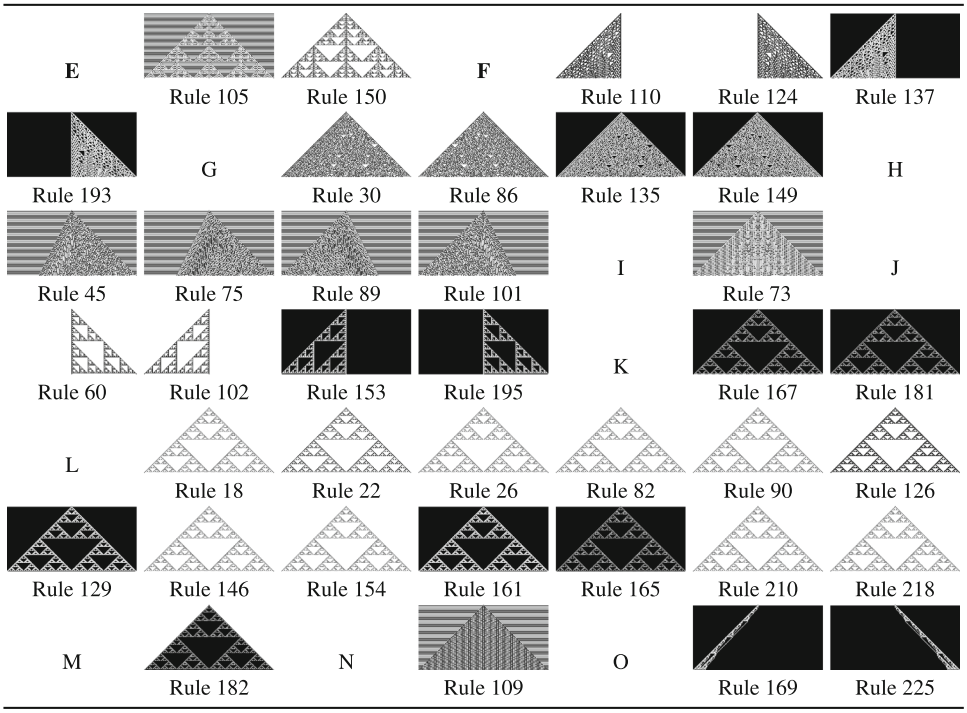
Criterion A: high variance

First, the mean variances of the $\Delta H(r)$ values for each cluster was examined with the idea being that if there is very little change in the data, then that could be used as evidence of uninteresting behaviour. The variance of $\Delta H(r)$ values for each rule was calculated, which were collated to compute the mean variance for each cluster. Upon examination of the variances in column two of Table 14, clusters A, B, C, D, N, and O have variances of 0, which indicates that the variation of the $\Delta H(r)$ values from the mean was minuscule. If there is not much variance from the mean, then this indicates that very little change in behaviour occurs over the observed lifetime of the cellular automata. This is one indicator which can be used to identify a cluster as not being interesting. The clusters which registered variance values, shown in bold text, indicates that these clusters are potentially visually interesting: clusters E–M.

Criterion B: frequent jaggedness

Second, the mean number of significant peaks in the plots was also examined for each cluster. Clusters A, B, C, D, F, G, H, I, N, and O have zero peaks, while clusters E, J, K, L, and M have significant peaks in their data. The number

Table 15 Clusters of interesting cellular automata rules



of peaks was calculated using a peak detection algorithm [42]. Those clusters that have peaks are shown in bold font, indicating that they can be classified as visually interesting clusters: clusters E, J, K, L, and M.

Criterion C: strong gradients

Third, the slope steepness for the $\Delta H(r)$ values was also examined by calculating the gradient of the slopes of the plots. This was done using values adopted by geographers to describe the gradient of a mountain slope [43]. The number of level, gentle, moderate, strong, and steep slopes was collected for each rule, and the mean number of each type of gradient was calculated for each cluster. Strong and steep slopes were deemed to be interesting, as these show spikes in the plot data, although there were no rules that produced a steep slope. Strong slopes in the graphs were detected in clusters: E, J, K, L, and M. It is no coincidence that these clusters are the same ones that have peaks in the data.

The proportion of level slopes can also indicate whether a cluster is interesting or not. Clusters A, B, C, D, and O have a high ratio of level slopes, with over 100 points along the plots being level, which indicates that they lie fairly flat, indicating no change in compression, and therefore, they can be classified as not being interesting.

Criterion D: small cluster size

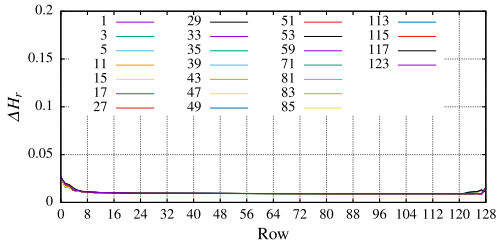
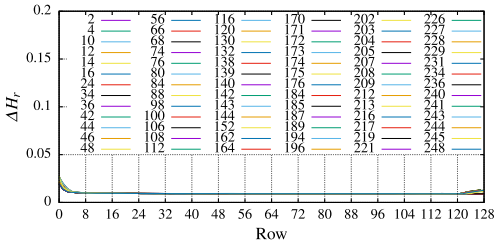
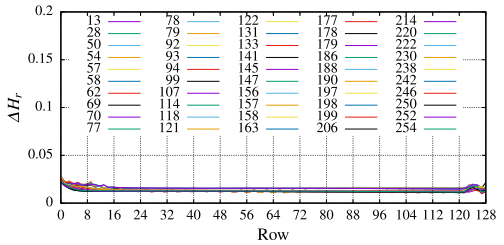
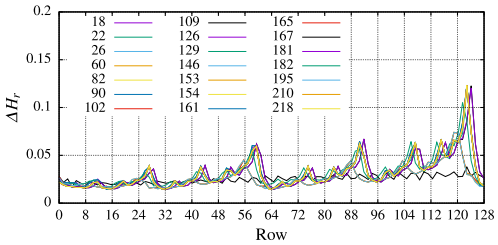
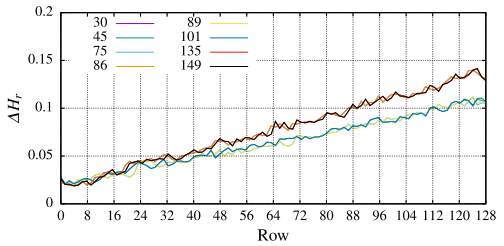
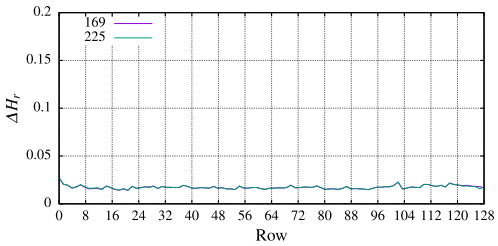
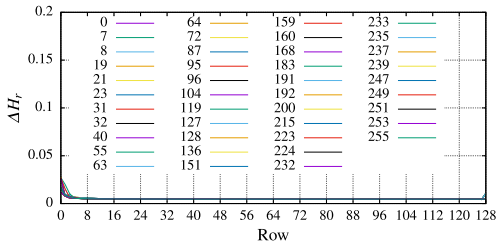
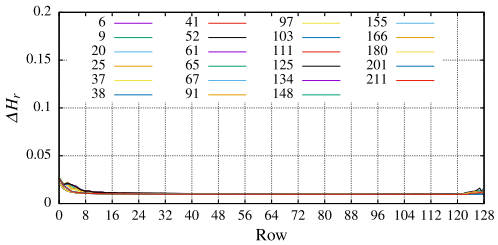
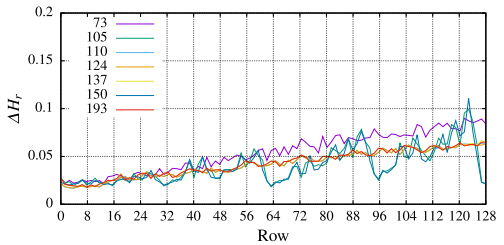
Another aspect of interestingness which was mentioned in Sect. 2 is the idea that interesting behaviour is often novel or different from what was observed before. We will describe this type of interesting behaviour as "unusual" behaviour. The PPM clustering technique can also be useful in discovering novel or unusual behaviours by identifying clusters with few entries as being a rare type of occurrence and, therefore, interesting. In this case, the ones being classified as rare or unusual are those clusters that only have one or two rules in them: clusters E, I, K, M, and N.

One example of an unusual set of behaviours is cluster K, which has two rules (167 and 181) that produce Sierpiński-like triangles. At first glance, the rules in cluster K look exactly like rule 165 (cluster L); however, due to the clustering algorithm, it has detected that these two rules are different, since they are offset by one cell on either side of the centre. This would not have been noticed without a meticulous analysis of each rule.

Another example where unusual behaviour has been discovered is the entries in cluster E. Rules 105 and 150 contain uniquely patterned triangles, which look similar to the Sierpiński-like triangles in clusters L and K, but have different internal patterns.

As mentioned earlier, clusters I and N have triangles with unique patterns that are not replicated by any other rules, and can also be counted as being interesting.

Table 16 The $\Delta H(r)$ graphs for cellular automata behaviours clustered using expectation maximisation clustering algorithm and PPM compression divergence data

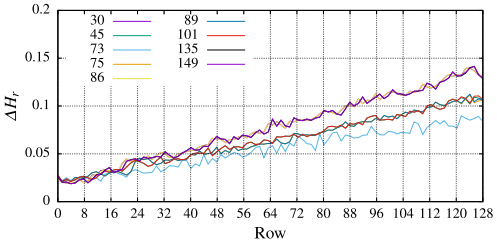
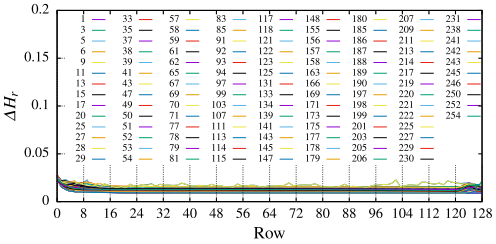
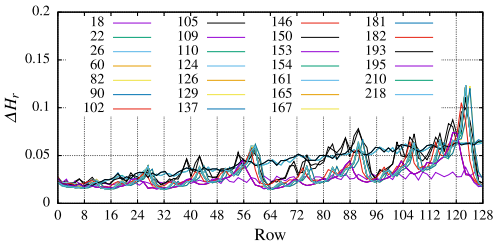
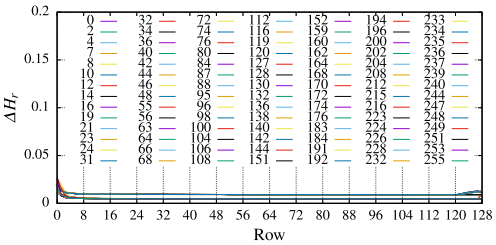
Cluster	$\Delta H(r)$ Graph	Cluster	$\Delta H(r)$ Graph
A		F	
B		G	
C		H	
D		I	
E			

Another cluster with unusual behaviour is cluster O which contains rules 169 and 225. This is an interesting cluster, because the patterns produced are unlike any of the other 256 rules seen in Table 12.

Cluster B produces triangles which have been split into two with different textures on either side. This cluster has been deemed uninteresting using the criteria listed above, since its plots did not vary enough, nor was it rare enough. However, since the clustering algorithm has managed to separate out this cluster, it is available for examination on its own, instead of being hidden amongst the other rules.

In summary, visually interesting clusters were found by analysing the data by examining the variance of the $\Delta H(r)$ values and choosing those that are significantly higher than

Table 17 The $\Delta H(r)$ graphs for cellular automata behaviours clustered using XMeans clustering algorithm and PPM compression data

Cluster	$\Delta H(r)$ Graph	Cluster	$\Delta H(r)$ Graph
A		C	
B		D	

the lower values; in this case, the uninteresting clusters did not register a variance value. The other way of finding visually interesting clusters is by counting the number of significant peaks or by examining the gradient of the plots produced. Unusual clusters were identified as being those which have a small number of rules in the clusters; in this case, those with one or two rules were also deemed to be interesting.

Interesting behaviours have been discovered using the PPM clustering. The visually interesting ones were deemed to be clusters E, F, G, H, I, J, K, L, and M. The unusual clusters were determined as being clusters E, I, K, M, N, and O. The clustering technique has made it easier to find unusual behaviours of cellular automata easily, especially for those in clusters K and O (Table 15).

Comparing the PPM clustering algorithm with other algorithms implemented by Weka

In this section, we describe experiments which we conducted to compare our clustering algorithms with previously established clustering algorithms as implemented by Weka (Waikato Environment for Knowledge Analysis) [44]. For our specific task of finding interesting behaviour in cellular automata, the requirements are that the clustering algorithm can automatically determine the number of clusters without any intervention from the user, and that it should not leave any rules unclustered. Two different clustering algorithms fitted into this category: expectation maximisation (EM) and XMeans.

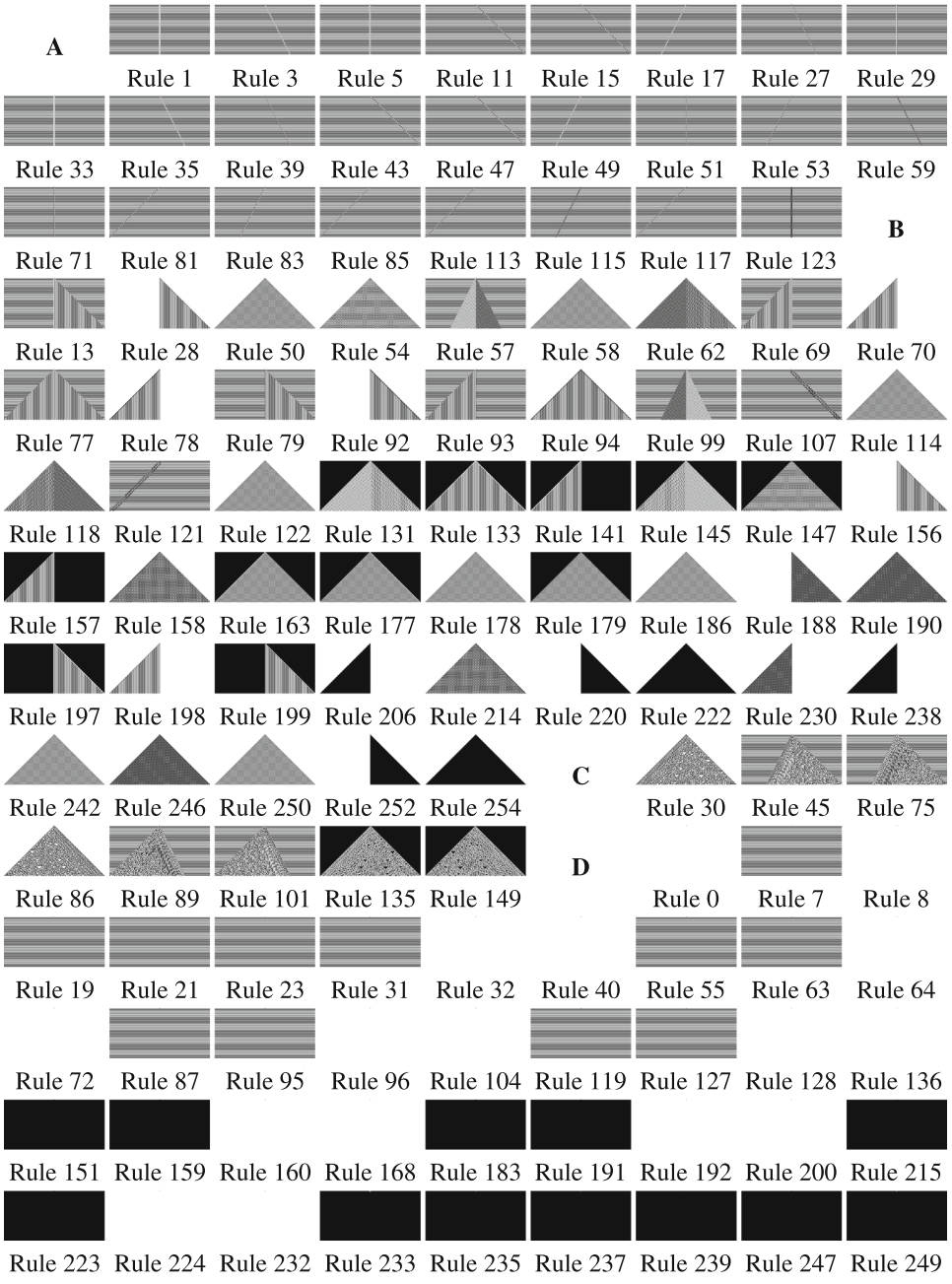
Weka uses a file format called attribute-relation file format (ARFF) [45] which is used to describe data represented by relations and attributes. The types of data that are accepted for each attribute are described in the ARFF file, which could be integers, floating point values, otherwise a specific set of strings (enumerations) [46]. The $\Delta H(r)$ divergence values used for the plots, such as in Figs. 2 and 3, for each rule were compiled into a Weka ARFF file with a relation being created for each rule. Each relation contained 129 numeric attributes. These attributes were used to store the divergence values for each row in the cellular automata that were calculated from the PPM compression codelengths.

Clustering using expectation maximisation

The EM clustering algorithm as implemented by Weka produced nine clusters. This was by selecting the EM clustering algorithm in Weka and using default settings. A table of cluster plots was created, as shown in Table 16. Examining this, clearly, there are some problematic clusters produced. For example, in clusters C, E, and G, there are at least two different types of patterns in each cluster. In addition, the plots of clusters A, F, and J look very similar. Cluster C's rules start in the same area, but then diverge into two distinct streams, although they do follow a similar pattern (Table 17).

The cellular automata clustered according to the EM clustering algorithm is shown in Table 18. When looking at the nine clusters produced by the EM clustering algorithm, some rules that exhibit similar behaviour have been split into different clusters, and rules that exhibit visually different behaviours have been grouped together into the same cluster.

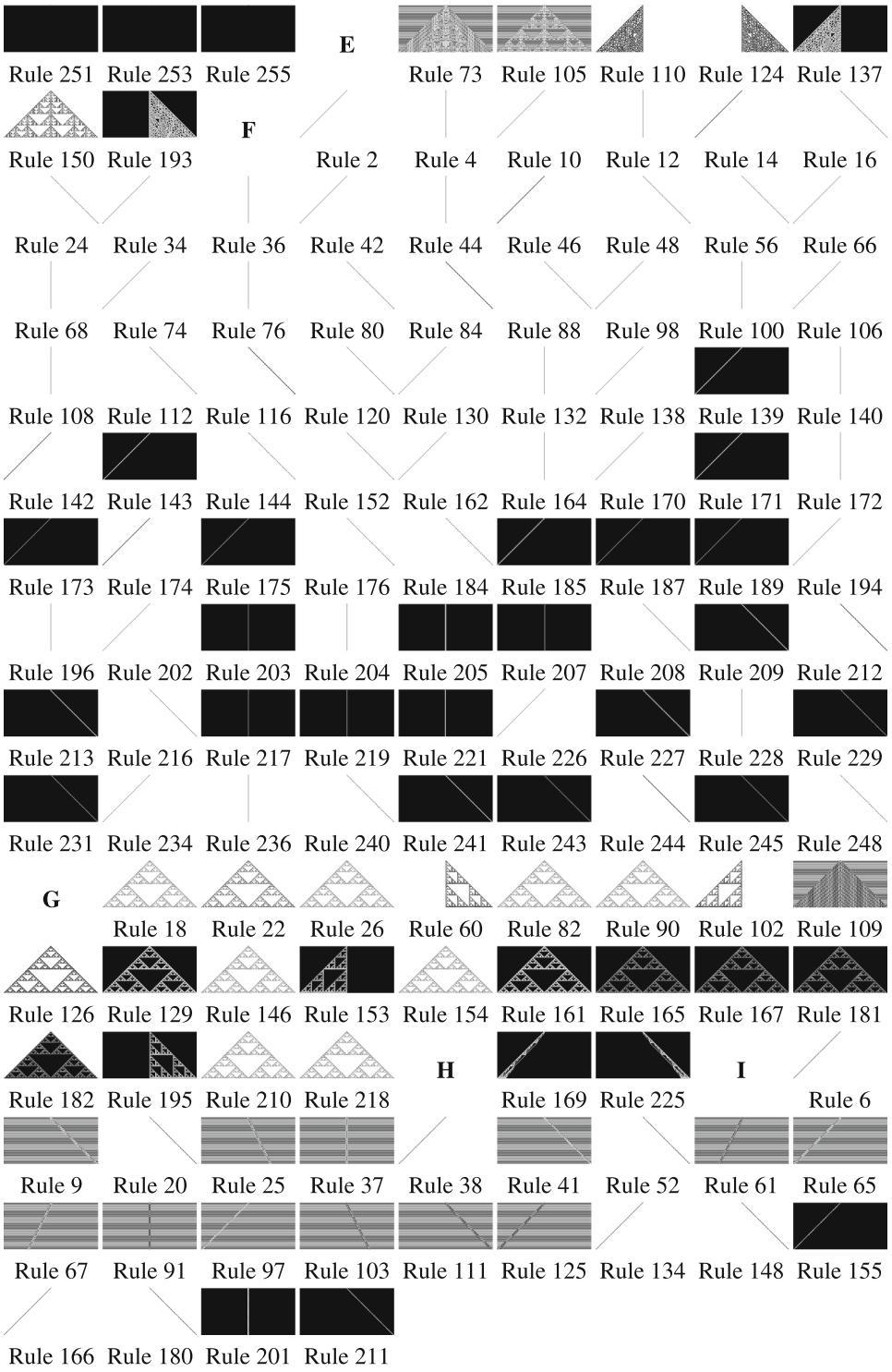
Table 18 The clusters produced by Weka's expectation maximisation clustering algorithm



Cluster A has rules that contain either a diagonal or vertical line on a horizontal black and white background. Cluster B has several types of triangles with various shading patterns on different backgrounds. It also has the two rules that contain thick diagonal lines, as in Fig. 5. Cluster C has cellular automata rules that produce isosceles and right-angled triangles with a chaotic internal pattern. Cluster D produces the same output as PPM's cluster A. Cluster E contains different types of triangles with complex internal patterns, which is a combination of PPM clusters E, I, and F. The rules in cluster F contains a thin diagonal or vertical line on a plain background. Cluster G contains Sierpiński-like triangles, which is a combination of PPM clusters J, K, L, and M, and also a triangle from PPM's cluster N

The Silhouette value is one way of determining if all the clusters are well matched with, as stated, a value close to 1 indicating good clustering, whereas badly matched clusters have a Silhouette value approaching −1. For comparison, the PPM clusters in Table 12 produced a Silhouette value of 0.8287; however, EM clustering gave a Silhouette value

Table 18 continued

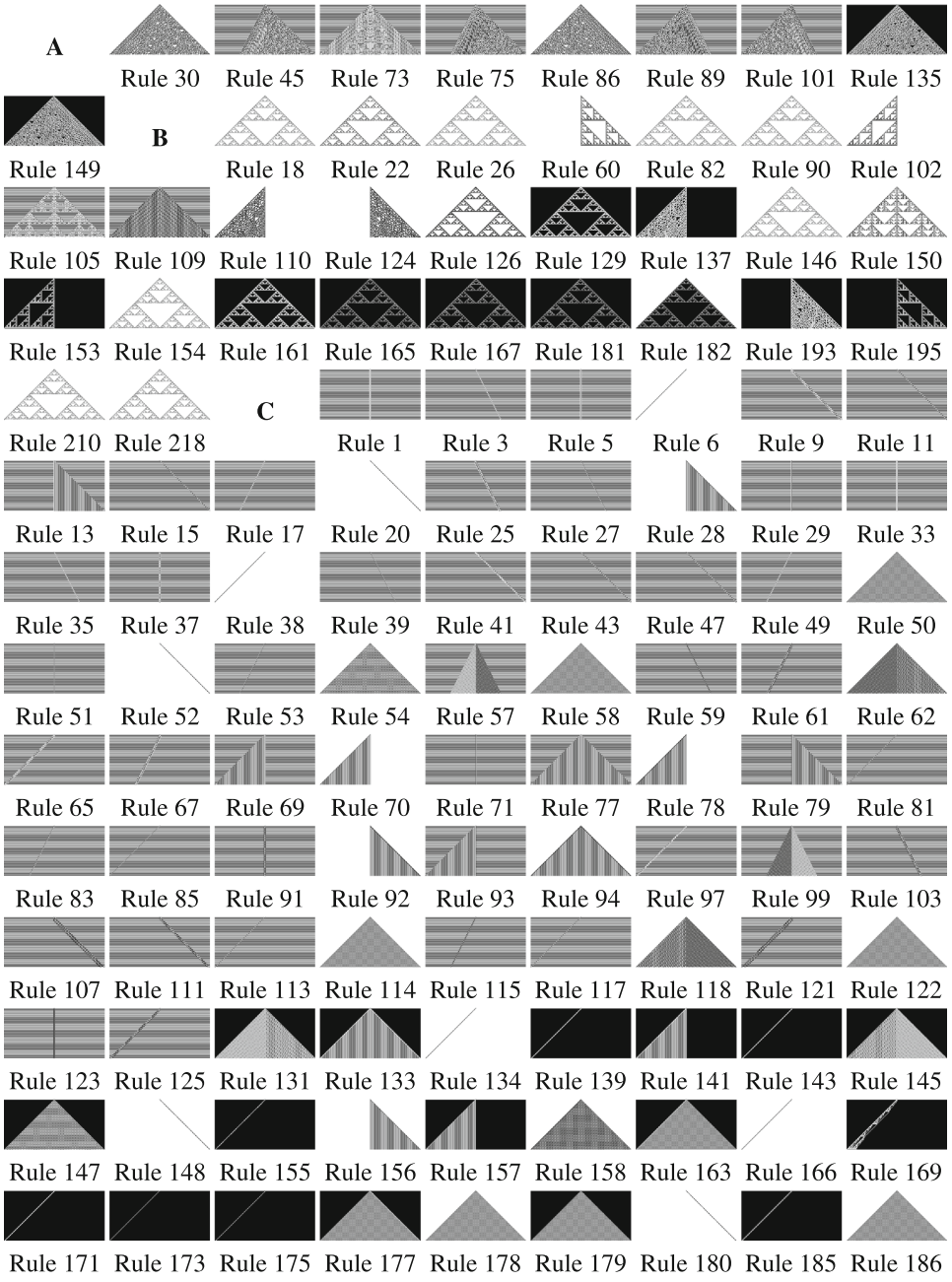


of 0.4811, which is considerably less than PPM's Silhouette value.

XMeans

Another clustering algorithm examined was XMeans, which produced four clusters. The $\Delta H(r)$ plots for the XMeans

Table 19 The clusters produced by the XMeans clustering algorithm in Weka

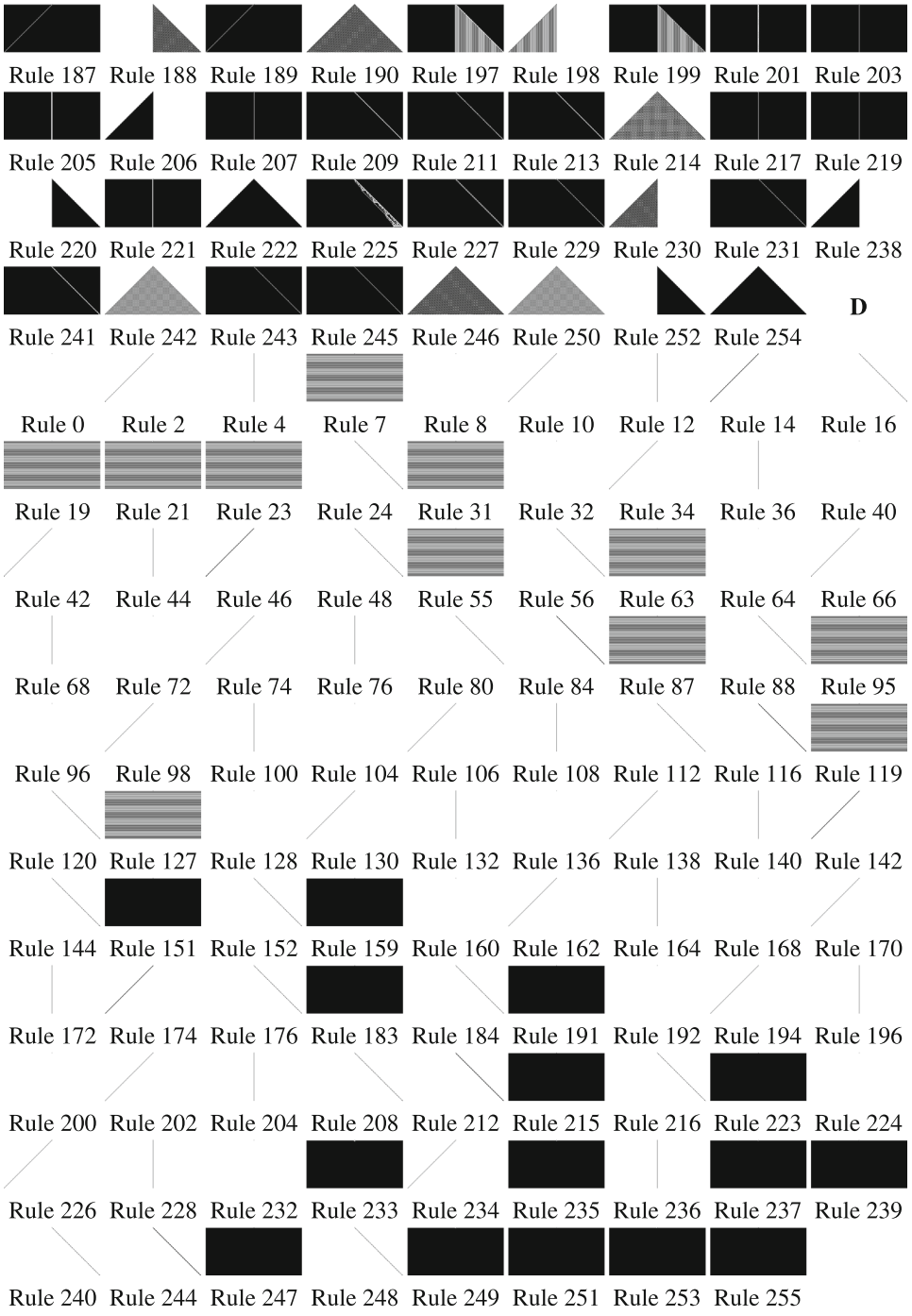


clustering algorithm are shown in Table 17. From the plots, we see that cluster A has three distinct lines, while cluster B has several distinct shapes in the graph, and the plots of clusters C and D look similar to each other.

When examining the clusters created by this algorithm, as shown in Table 19, the first cluster was the equivalent of combining the PPM clusters G, H and I. The second cluster contains a combination of PPM clusters F, J, K, L, M, and N. The next two clusters are larger conglomerations of different types of patterns. XMeans clustering had a Silhouette score of 0.3685, which is worse than both PPM and EM Clustering.

It can be concluded that EM and XMeans clustering algorithms are not fine grained enough to show distinct cellular automata behaviour. However, EM clustering was able to pick out one of the unusual behaviours in EM cluster H. Both EM and XMeans clustering algorithms were able to isolate the visually interesting clusters, which were also highlighted in Table 14; however, they were not able to isolate the unusual or rare behaviours.

Table 19 continued



Conclusion

The study looked at how compression can be used to discover interesting behaviours in complex systems with a specific look at 1D elementary cellular automata as a case study. The new algorithm combines PPM compression with a clustering algorithm which is chosen using the Silhouette average as a guide to give the number of clusters. By calculating the divergence in cross-entropy using the difference in compression between each cellular automata row output, it was possible to cluster effectively the cellular automata rules according to their behavioural output. It was then also possible to find interesting and unusual behaviours quickly based on the cluster data.

Several clustering algorithms were compared against each other using the same data: the divergence in cross-entropy

values of all the 256 rules. Three clustering algorithms implemented by the `scikit-learn` Python toolkit gave the same outcome: BIRCH, Hierarchical Agglomerative Clustering, and k-means. As one evidence of the effectiveness of the approach, they were able to place rules 181 and 167 in a separate group by themselves. These rules produce Sierpiński Triangle output, but they are unique, because their apexes are not on the central cell, unlike the other rules that produce patterns similar to Sierpiński triangles. Rules 169 and 225, which produce an unusual diagonal of lines and small triangles which increase in width, were also isolated using the PPM clustering algorithm.

In Weka, other clustering algorithms were run, with only two of the clustering algorithms, EM and X-Means, automatically setting the number of clusters without user intervention, and had no unallocated rules. The EM clustering algorithm was able to isolate rules 169 and 225 into a separate cluster as it seems appropriate. Even though the two clustering algorithms were able to isolate visually interesting cellular automata rules from the others, they were not fine grained enough to give an insight into the different patterns of behaviour produced by the cellular automata rules.

Using the divergence in cross-entropy values produced by PPM compression gives a useful insight into the life of the complex system for the duration of observation. It can then be used to help identify both visually interesting and unusual cellular automata behaviour.

References

1. Maiti S, Roy Chowdhury D (2017) Study of five-neighborhood linear hybrid cellular automata and their synthesis. In: Giri D, Mohapatra RN, Begehr H, Obaidat MS (eds) Mathematics and Computing. ICMC 2017. Communications in Computer and Information Science, vol 655. Springer, Singapore
2. Adams C, Parekh H, Louis SJ (2017) Procedural level design using an interactive cellular automata genetic algorithm. In: Proceedings of the genetic and evolutionary computation conference companion, GECCO'17, ACM, New York, NY, pp 85–86 (2017). https://doi.org/10.1145/3067695.3075614
3. Wolfram S (1983) Cellular Automata. Los Alamos Science
4. Carvalho DR, Freitas AA, Ebecken N (2005) Evaluating the correlation between objective rule interestingness measures and real human interest. In: European Conference on Principles and Practice of Knowledge Discovery in Databases, vol 9. pp 453–461
5. Freitas AA (2006) Are we really discovering "Interesting" knowledge from data? Expert Update SGAI 9(1):41–47
6. Gaines BR (1996) Transforming rules and trees into comprehensible knowledge structures. In: Advances in knowledge discovery and data mining. American Association for Artificial Intelligence, Menlo Park, CA, pp 205–226
7. McGarry K (2005) A survey of interestingness measures for knowledge discovery. Knowl Eng Rev 20:39–61
8. Hilderman RJ, Hamilton HJ (1999) Knowledge Discovery and Interestingness Measures: A Survey. Tech. rep., University of Regina, Regina, Saskatchewan, Canada. http://www.cs.uregina.ca/Research/Techreports/2000-01.pdf
9. Benureau FCY (2015) Self-exploration of sensorimotor spaces in robots. Ph.D. thesis, Bordeaux University. https://tel.archives-ouvertes.fr/tel-01251324/document
10. Hall P, Morton SC (1993) On the estimation of entropy. Ann Inst Stat Math 45:69–88
11. Blanchard J, Guillet F, Gras R, Briand H (2005) Using information-theoretic measures to assess association rule interestingness. In: Fifth IEEE International Conference on Data Mining. IEEE. https://doi.org/10.1109/ICDM.2005.149
12. Schmidhuber J (2009) Driven by compression progress: a simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. Anticipatory behavior in adaptive learning systems. Springe, Berlin, Heidelberg, pp 48–76
13. Zhong N, Liu C, Ohsuga S (2001) Dynamically organizing KDD processes. Int J Pattern Recognit Artif Intell 15(3):451–473
14. Correia JH, Stumme G, Wille R, Wille U (2003) Conceptual knowledge discovery–a human-centered approach. Appl Artif Intell 17:281–302
15. Hudson NJ (2011) Musical beauty and information compression: complex to the ear but simple to the mind? BMC Res Notes 4(1):9
16. Wolfram S (1984) Cellular automata as models of complexity. Nature 311:419
17. Wolfram S (2002) A new kind of science. Wolfram Media. https://www.wolframscience.com/nks/
18. Li W, Packard N, Langton CG (1990) Transition phenomena in cellular automata rule space. Phys D 45:77–94
19. Li W, Packard N (1990) The structure of the elementary cellular automata rule space. Complex Syst 4:281–297
20. Li W (1992) Phenomenology of non-local cellular automata. J Stat Phys 68:829–882
21. Oliveira GMB, Oliveira PPB, Omar N (2001) Definition and application of a five-parameter characterisation of one-dimensional cellular automata rule space. Artif Life 7:277–301
22. Zenil H (2010) Compression-based investigation of the dynamical properties of cellular automata and other systems. Complex Syst. https://www.complex-systems.com/abstracts/v19_i01_a01/
23. Teahan WJ (2018) A compression-based toolkit for modelling and processing natural language text. Information 9:294
24. Mahoui M, Teahan WJ, Thirumalaiswamy Sekhar AK, Chilukuri S (2008) Identification of gene function using prediction by partial matching (PPM) language models. In: Proceedings of the 17th ACM conference on information and knowledge management, CIKM '08. ACM, New York, pp 779–786. https://doi.org/10.1145/1458082.1458186
25. Mahoney M (2016) Large text compression benchmark. http://mattmahoney.net/dc/text.html. Accessed 15 Jan 2018
26. Moffat A (1990) Implementing the PPM data compression scheme. IEEE Trans Commun 38:1917–1921
27. Im Walde SS (2003) Experiments on the automatic induction of German semantic verb classes. Comput Linguist 32:159–194
28. Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20:53–65
29. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E

(2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
30. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. SIGKDD Explor Newsl 11(1):10–18. https://doi.org/10.1145/1656274.1656278
31. Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: an efficient data clustering method for very large databases. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, SIGMOD'96, vol 25. ACM, ACM Press, pp 103–114. https://doi.org/10.1145/233269.233324
32. Ester M, Kriegel HP, Sander J, Xu X et al (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, vol 96. AAAI Press, pp 226–231
33. Hartigan JA, Wong MA (1979) Algorithm as 136: A k-means clustering algorithm. J R Stat Soc Ser C (Appl Stat) 28(1):100–108
34. Kriegel HP, Schubert E, Zimek A (2017) The (black) art of runtime evaluation: are we comparing algorithms or implementations? Knowl Inf Syst 52(2):341–378
35. Cheng Y (1995) Mean shift, mode seeking, and clustering. IEEE Trans Pattern Anal Mach Intell 17(8):790–799
36. Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 22(8):888–905. https://doi.org/10.1109/34.868688
37. Ng AY, Jordan MI, Weiss Y (2001) On spectral clustering: analysis and an algorithm. In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, NIPS01. MIT Press, pp 849–856
38. Scitkit-Learn: Clustering (2018) http://scikit-learn.org/stable/modules/clustering.html. Accessed 21 March 2018
39. Everitt BS, Landau S, Leese M, Stahl D (2011) Cluster analysis, 5th edn. Wiley, New York
40. Apolloni B, Pedrycz W, Bassis S, Malchiodi D (2008) The puzzle of granular computing. Springer, New York
41. Ahuja A (2013) Interpretation of Silhouette Plots (Clustering). https://pafnuty.wordpress.com/2013/02/04/interpretation-of-silhouette-plots-clustering/. Accessed 15 Jan 2018
42. Endolith: Peak detect (2011). https://gist.github.com/endolith/250860. Accessed 15 Jan 2018
43. Barcelona Field Studies Centre: Measuring Slope Steepness. http://geographyfieldwork.com/SlopeSteepnessIndex.htm. Accessed 15 Jan 2018
44. Frank E, Hall MA, Witten IH (2016) The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques". Morgan Kaufmann
45. University of Waikato: Weka ARFF. http://weka.wikispaces.com/ARFF. Accessed 15 Jan 2018
46. Stephen RG (1995) Weka: the Waikato environment for knowledge analysis. In: Proceedings of the New Zealand Computer Science Research Students Conference, pp 57–64. https://www.cs.waikato.ac.nz/%7Eml/publications/1995/Garner95-WEKA.pdf