Imperial College London

BEng Individual Project

Imperial College London

Department of Computing

Reverse Engineering Cellular Automata

Author:
Manuj Mishra

Supervisor:
Prof. Andrew Davison

Second Marker:
Dr. Edward Johns

June 1, 2022

Abstract

Evolutionary algorithms are effective tools for black-box optimisation problems.

Acknowledgements

[Acknowledgements here]

Contents

1 Introduction **5**
- 1.1 Motivation . . . 5
- 1.2 Objectives . . . 5
- 1.3 Contributions . . . 6
- 1.4 Technical Challenges . . . 6

2 Preliminaries **7**
- 2.1 Cellular Automata . . . 7
  - 2.1.1 Neighbourhood Functions . . . 7
  - 2.1.2 Conway's Game of Life . . . 8
  - 2.1.3 Wolfram's Classification . . . 9
- 2.2 Evolutionary Algorithms . . . 11

3 Related Works **12**
- 3.1 Binary Outer-Totalistic CA . . . 12
  - 3.1.1 Exploration . . . 12
  - 3.1.2 Learning . . . 13
- 3.2 Continuous Reaction-Diffusion CA . . . 16

4 Design **17**
- 4.1 Evolutionary Algorithm Toolkit . . . 17
- 4.2 Cellular Automata Simulator . . . 17

5 Method **18**
- 5.1 Life-like CA . . . 18
  - 5.1.1 Genetic Algorithm . . . 18
  - 5.1.2 Species Analysis . . . 18
- 5.2 Reaction-diffusion CA . . . 18
  - 5.2.1 Evolutionary Strategies . . . 18
  - 5.2.2 Particle Swarm Optimisation . . . 18

6 Application **19**
- 6.1 Maze Generation . . . 19
  - 6.1.1 Region Merging Algorithm . . . 19
  - 6.1.2 Fitness Evaluation . . . 19
  - 6.1.3 Selection and Quality-Diversity . . . 19

7 Evaluation **20**

8 Conclusions **21**

9 Ethical Considerations **22**

List of Figures

2.1 (a) von Neumann Neighbourhood and (b) Moore neighbourhood on a 2D square lattice [1] 8
2.2 The glider pattern in the Game of Life [2] 9
2.3 Two possible configurations of a Life-like CA[CITE] 9
2.4 Rule 110 progression with random initialisation [3] 10

3.1 Configurations generated from P-class (a,b) and O-class (c,d) rules [4] 12
3.2 Map of fertile, infertile, mortal, and immortal regions in binary-state RDCA rulespace [5] 13
3.3 20-cell, two step neighbourhood in space and time 14

List of Definitions

2.1 Definition (Cellular automaton) . . . 7
2.2 Definition (Inner-totalistic) . . . 8
2.3 Definition (Outer-totalistic) . . . 9
2.4 Definition (Birth-survival notation) . . . 9

3.1 Definition (Majority Problem) . . . 15
3.2 Definition (GKL Classifier) . . . 15

List of Algorithms

1 Schematic Evolutionary Algorithm . . . 11

Chapter 1

Introduction

1.1 Motivation

Predicting effects is easier than predicting causes. This is the statement of the Inverse Problem. In science, we find it easier to estimate observations from a parameterised model of the world than to deduce parameters from observations. This is a result of the causal opacity of time which eliminates information through the unforgiving forces of selection and entropy. Given knowledge of dinosaur evolution, we may have strong hope of predicting where fossils of certain species lie but to build a rigorous taxomony based on fossils alone is insurmountably harder. A physical model may allow us to predict the subatomic particles ejected when two protons collide at high speed but building physical theories based on these collisions is significantly more difficult, especially when there are multiple equally valid explanations. Regardless, the pursuit of the Inverse Problem is critical to advancing scientific theory around a system's behaviour. Observations can only validate or falsify but prediction paves the way for novel scientific models.

In this thesis, we tackle the inverse problem for cellular automata (CAs). These are simple yet powerful models of computation in which multiple "cells" on a discrete lattice are simultaneously updated at regular time steps. The state of each cell depends exclusively on the state of the cells in a local neighbourhood around it in the previous time step. This localised interaction makes CAs a useful abstract representation of physical and biological systems in the real world from gas molecule interactions[CITE] to human land behaviour[CITE]. Much like these systems, CAs can exhibit chaos, nonlinear dynamics, and the emergence of complexity. As well as simulatory models, CAs are powerful computational engines due to their inherently parallel structure. This makes their study a useful endeavour in the field of distributed computation too.

Top-down investigations into CA behaviour are vast and varied. Mathematical analyses seek to classify CAs and prove general results about long-term behaviour from intrisic properties. In the natural and social sciences, CAs are designed to model real world systems. Both of these endeavours seek to analyse the behaviour of a CA from its structure, transition function, and possibly initial conditions. In this thesis, we explore a bottom-up approach where we deduce the underlying properties of a CA by observing its behaviour. In particular, we utilise evolutionary algorithms to search across several classes of CA. Evolutionary algorithms (EAs) have long been held as effective tools for black-box optimisation problems. Grounded in the principles of Darwinian evolution, EAs traverse over a search space by performing selection, mutation, and crossover on a population of candidate solutions. Increasingly strong solutions are discovered as the fitness of the population grows. Using EAs, we tackle multiple optimisation problems from the imitation of particular CA behaviour to generation of desirable long-term states.

1.2 Objectives

We develop a system to discover cellular automata that are highly likely to exhibit a desired behaviour. Key aims include:

1. **Learning Life-Like CA**
   Deduce the underlying transition rule behind cellular automata that exhibit chaotic and complex behaviour similar to that of Conway's Game of Life (see Subsection 2.1.2).

2.

1.3 Contributions

The key contributions of this project are as follows:

1. **Evolutionary Algorithm Toolkit**
   A versatile toolkit that implements multiple evolutionary algorithms to train and optimise different classes of CA. We use this to successfully predict the update rule of binary outer-totalistic CA from observations of the CA running on random initial conditions. We show that this can be extended to continous automata by predicting the parameters of diffusion-reaction equations from simulations of chemical reactions.

2. **Cellular Automaton Simulator**
   A system that can efficiently simulate discrete and continuous cellular automata. This allows a broad range of fitness functions to be implemented in the EA toolkit. This can also render snapshots of the CA directly during simulation which allows animations to be efficiently generated afterwards.

3. **Procedural Maze Generator**
   A CA-based maze generation program that uses the EA toolkit to produce difficult mazes with characteristics optimised to user preference.

1.4 Technical Challenges

Chapter 2

Preliminaries

2.1 Cellular Automata

A cellular automaton (CA) is a computational model that performs multiple parallel computations, each depending on local interactions, to produce complex global behaviour. We define a CA formally as follows.

Definition 2.1 (Cellular automaton)**.** *A cellular automaton is an n-dimensional finite grid of computational units called cells. Each cell c_i is characterised by:*

- *A discrete state variable $\sigma_i(t) \in \Sigma$, where i indicates the index of the cell in the lattice, t indicates the current time step, and Σ denotes the finite set of all state variables.*
- *A finite local neighbourhood set $\mathcal{N}(c_i)$ with cardinality N.*
- *A transition function $\phi : \Sigma^N \to \Sigma$ which takes local neighbour states as input. This is also known as the CA "update rule".*

At each time step, the state of each cell is simultaneously updated according to the transition function. That is, $\sigma_i(t+1) = \phi(\ \{\sigma_j(t) \mid c_j \in \mathcal{N}(c_i)\}\ )$

Due to the breadth of systems studied in CA literature, the constraints of this definition are often altered to produce interesting arrangements. For example:

- The structure need not be a square grid. CA have been studied on hexagonal grids[6], aperiodic tessellations such as as the Penrose tiling[7], and even randomly generated structures like the Voronoi partition[8].
- The system need not be deterministic. Probabalistic cellular automata (PCA) have stochastic transition functions which describe a probability distribution of possible outcomes for any given input. PCA are able to model random dynamical systems in the real world from stock markets[9] to infectious diseases[10].
- The state space Σ need not be finite. In this thesis we will explore multiple possible state variable representations including bit arrays and continuous vectors.

For the purpose of this thesis, we will assume the original definition of CA unless otherwise stated.

2.1.1 Neighbourhood Functions

We consider a "neighbourhood function" for each cell $c_i \mapsto \mathcal{N}(c_i)$. This makes it easier to discuss neighbourhood sets of cells in the CA, each of which are typically homogenous.

There are many possible neighbourhood functions for any given CA geometry. When defining the neighbourhood function, we select a distance metric $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ to measure the proximity of two cells and we set a threshold T under which we consider two cells to be within each other's neighbourhood.

$$c_i \in \mathcal{N}(c_j) \iff d(c_i, c_j) \leq T$$

There are two neighbourhoods that are frequently used on Euclidean lattices. The *von Neumann neighbourhood* contains all cells within a Manhattan distance of 1. For a 2D square lattice, this contains the cell itself and the 4 cells in the cardinal directions. For a 3D cubic lattice, it contains the central cell and a 6-cell octahedron around it. The *Moore neighbourhood* contains all cells at a Chebyshev distance of 1. For a 2D square lattice, this is the central cell with the 8 neighbouring cells in a square around it. In the 3D case, it is a cube.

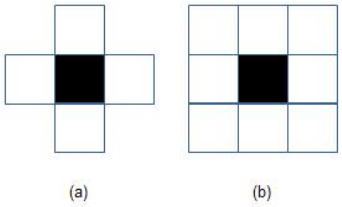


Figure 2.1: (a) von Neumann Neighbourhood and (b) Moore neighbourhood on a 2D square lattice [1]

In a finite grid CA, border cells must be given special consideration since they do not have the same number of neighbours as interior cells and therefore cannot share the same neighbourhood function. One option is to define a case-wise neighbourhood function with different behaviour for border cells. Another option is to freeze the state of border cells. In the field of partial differential equations, this is known as setting "fixed boundary conditions". The problem can also be circumvented entirely by relaxing the finite grid assumption and allowing cells to "wrap around" the grid. This is known as setting "periodic boundary conditions" and can be imagined visually as running the CA on an infinite periodic tiling or, alternatively, on a torus.

2.1.2 Conway's Game of Life

A popular example of a CA is the Game of Life (henceforth "Life") formulated by John Conway in 1970 [11]. It consists of a 2D grid of cells, each with a boolean state variable signifying that the cell is either "alive" or "dead". The transition rule takes as input the cell's own state $\sigma_i(t)$ and the number of living individuals in the cell's Moore neighbourhood (excluding itself), denoted n. This is as follows:

$$\phi(\sigma_i(t), n) = \begin{cases} 0 & \sigma_i(t) = 1 \text{ and } n < 2 \text{ (Death by "exposure")} \\ 0 & \sigma_i(t) = 1 \text{ and } n > 3 \text{ (Death by "overcrowding")} \\ 1 & \sigma_i(t) = 1 \text{ and } n \in \{2, 3\} \text{ (Survival)} \\ 1 & \sigma_i(t) = 0 \text{ and } n = 3 \text{ (Resurrection)} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Despite its simple setup and update rule, Life can exhibit the emergence of complex patterns. It is possible to simulate a fully universal Turing machine within Life [CITE] and, as a corollary of the Halting Problem [CITE], this means that Life is undecidable. Given two configurations, it is impossible to algorithmically determine whether one will follow the other.

Patterns found within Life include still lifes like the *block* which are fixed-point solutions to the transition function as well as periodic oscillators like the *beacon* which has period 2. There are also periodic patterns that move across the lattice such as the *glider* pattern. It is possible to discover new stable patterns by repeatedly running specific rules on random initial patterns of a pre-determined density (called soups) and classifying the objects remaining after transient reactions have dissipated. Large-scale experiments of this nature are called "soup searches"[CITE].

A CA is considered "Life-like" if it exists on a 2D lattice, has binary state, uses the Moore neighbourhood function. Life-like cellular automata exist in two varieties: inner-totalistic and outer-totalistic.

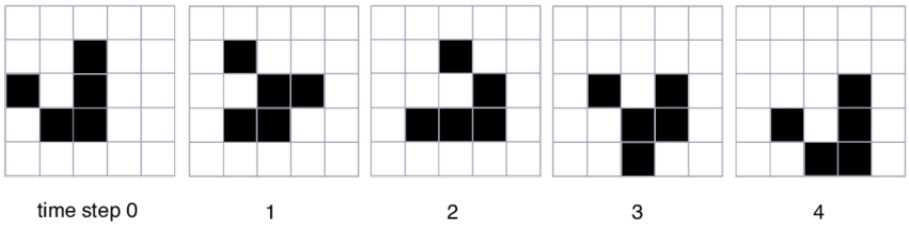


Figure 2.2: The glider pattern in the Game of Life [2]

Definition 2.2 (Inner-totalistic)**.** *A Life-like CA is inner-totalistic if the output of the transition function depends only on the number of living cells in a cell's neighbourhood (including the cell itself).*

$$\sigma_i(t+1) = \sigma_j(t+1) \iff \sum_{c_p \in \mathcal{N}(c_i)} \sigma_p(t) = \sum_{c_q \in \mathcal{N}(c_j)} \sigma_q(t)$$

Definition 2.3 (Outer-totalistic)**.** *A Life-like CA is outer-totalistic if the output of the transition function depends on both the number of living cells in a cell's neighbourhood and the state of the cell itself.*

$$\sigma_i(t+1) = \sigma_j(t+1) \iff \sum_{c_p \in \mathcal{N}(c_i)} \sigma_p(t) = \sum_{c_q \in \mathcal{N}(c_j)} \sigma_q(t) \quad \text{and} \quad \sigma_i(t) = \sigma_j(t)$$

As an example of the subtle difference here, consider the configurations shown in Figure 2.3. An inner-totalistic CA would yield identical configurations in the next time step since both input configurations have 3 active cells in the neighbourhood set. However, an outer-totalistic CA would treat both configurations separately as one has an live centre cell and the other has a dead centre cell. This discrepancy corresponds to a great difference in the size of search spaces. There are $2^{10} = 1024$ inner-totalistic CA but $2^{18} = 262144$ outer-totalistic CA. A B/S rulestring represents the transition function of an outer-totalistic CA in a form called birth-survival notation.

Figure 2.3: Two possible configurations of a Life-like CA[CITE]

Definition 2.4 (Birth-survival notation)**.** *Let N_b and N_s be sets of integers. We say an outer-totalistic CA has rulestring* BN_b/SN_s *if it has transition function:*

$$\phi(\sigma_i(t), n) = \begin{cases} 1 & \sigma_i(t) = 0 \text{ and } n \in N_b \text{ (Birth)} \\ 1 & \sigma_i(t) = 1 \text{ and } n \in N_s \text{ (Survival)} \\ 0 & \text{otherwise} \end{cases}$$

Using this notation, we can represent the Game of Life as B3/S23. In this thesis, when we refer to Life-like CA, we implicitly assume the outer-totalistic variety.

2.1.3 Wolfram's Classification

The choices of lattice geometry, neighbourhood function, state variable, and transition rule define the behaviour of a CA. Fixing the former three factors, Wolfram [12] classified CAs based on transition rules as follows:

1. Class 1 (Null) : Rules that lead to a trivial, uniform state
2. Class 2 (Fixed-point / Periodic) : Rules that lead to stable or periodic patterns
3. Class 3 (Chaotic) : Rules that lead to chaotic patterns
4. Class 4 (Complex) : Rules that lead to complex, long-lived impermanent patterns

Elementary cellular automata are defined on the simplest nontrivial lattice, a finite one-dimensional chain. The neighbourhood of each cell contains the cell itself and the two cells adjacent to it on either side. The state variable is a boolean which means there are $2^3 = 8$ possible neighbourhood state configurations. A transition rule maps each of these neighbourhood states to a resultant state and can therefore be represented as an 8-digit binary rule table $(t_7t_6t_5t_4t_3t_2t_1t_0)$ where configuration (000) maps to t_0, (001) maps to t_1, ..., and (111) maps to (t_7). Consequently, there are $2^8 = 256$ possible transition functions for elementary CA.

The Wolfram code, a number between 0 and 255 obtained by converting the binary rule table to decimal, is the standard naming convention for these rules. Rule 110 is particularly notable as it can exhibit class 4 behaviour [3] and is Turing complete [13]. Figure 2.4 shows an example progression of a Rule 110 system. Each row of pixels represents the state of the automaton at one snapshot in time with the topmost row representing the randomized initial state. It shows the emergence, interaction, and subsequent dissipation of multiple long-lived impermanent patterns.

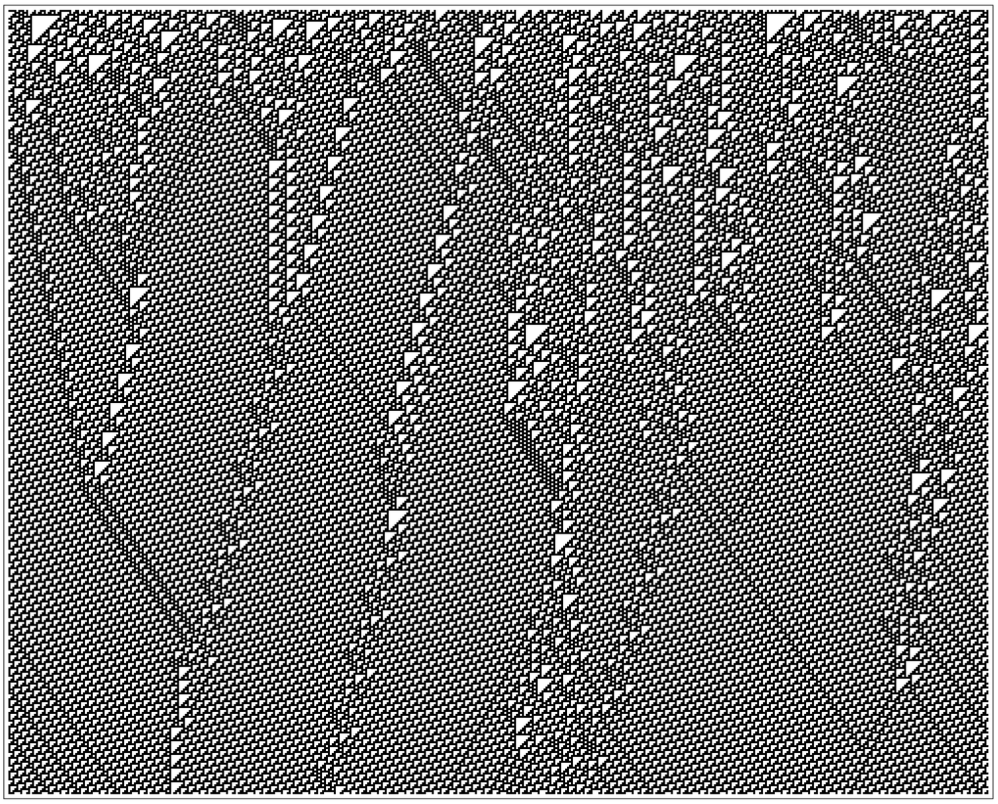


Figure 2.4: Rule 110 progression with random initialisation [3]

2.2 Evolutionary Algorithms

Evolutionary algorithms (EAs) are a family of heuristic-based search algorithms for black-box optimisation problems. They are inspired by biological evolution. A population of candidate solutions is initialized and modified through repeated selection, mutation, and recombination. We define an EA formally in Algorithm 1.

Algorithm 1 Schematic Evolutionary Algorithm

Require: S - the set of possible chromosome values
Ensure: $s^* \in S$
  $t \leftarrow 0$
  $M_0 \leftarrow \mu$ random individuals from S
  while stopping condition is false **do**
    EVALUATE(M_t)
    $P_t \leftarrow$ SELECTPARENTS(M_t) ▷ Parents
    $\Lambda_t \leftarrow$ RECOMBINE(P_t) ▷ Children
    $Pmod_t \leftarrow$ MUTATE(P_t)
    $\Lambda mod_t \leftarrow$ MUTATE(Λ_t)
    $M_t + 1 \leftarrow$ SELECTPOPULATION($Pmod_t$, Λmod_t)
    $t \leftarrow t + 1$
  end while
  $s^* \leftarrow$ FINDBESTCANDIDATE(P_t)

The initial selection phase (SELECTPARENTS()) uses a objective function, also known as a fitness function, to compare and select the top candidates. Recombination produces a set of children that have similar properties to some subset of the parents. This exploits the cumulative progress of the evolutionary process embedded in the parent candidates. Mutation explores new areas of the search space by perturbing properties of the parents and children. The latter selection phase (SELECTPOPULATION()) produces a new population from the modified parents and children. Population-wide selection criteria can be enforced in this phase. For example, certain parents can be eliminated if they have survived for too many generations or, symmetrically, children can be granted immunity for a particular number of generations.

EAs are valued for their broad applicability as they require no information about the constraints or derivative of the objective function. In fact, an explicit representation of the objective function is not even neccessary to run an EA as long as candidates can be compared to each other. Selection pressure can then be introduced in the form of tournament-based elimination.

Typically, an EA acts on a population of "chromosomes" which are indirect encodings of candidate solutions. The structure of the chromosome is called the "genotype" and the structure of the corresponding solution is called the "phenotype". In this thesis, the genotype will usually be a set of parameters that characterise the transition function for a particular class of CA. The corresponding phenotype is the cellular automaton with that transition function.

Chapter 3

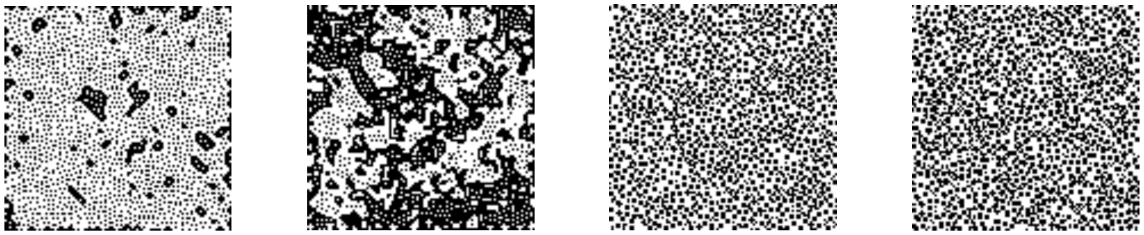
Related Works

This chapter summarises recent work on the analysis and learning of cellular automata. We focus our exploration on two classes of automata. The first are binary outer-totalistic CA, also known as life-like CA (see Def 2.3). The second are continuous reaction-diffusion CA which model simple chemical reactions. As we will see, these are a natural extension of life-like CA which, under certain constraints, themselves can be interpreted as discrete reaction-diffusion simulations with each cell accommodating the reactant or the substrate - a binary choice.

3.1 Binary Outer-Totalistic CA

3.1.1 Exploration

Early attempts to categorise 2D cellular automata by Packard and Wolfram[14] extend Wolfram's original 4 categories. They classify rules based on information content and rate of information transmission measured using Shannon entropy and Lyapunov exponents respectively. However, these metrics do not translate to clear global decision boundaries between Wolfram's classes. As proven by Yaku[15], many questions about global properties of 2D CA are formally undecidable which makes the construction of definitions based on long-term outcomes difficult.



(a) B12345678/S23 (b) B12345678/S234 (c) B1/S345 (d) B1/S345678

Figure 3.1: Configurations generated from P-class (a,b) and O-class (c,d) rules [4]

Adamatzky et al.[4] produces a systematic analysis of life-like CA in which the birth and survival sets are contiguous intervals. These are dubbed "binary-state reaction-diffusion cellular automata (RDCA)" as they provide a discretized model of simple two-chemical reactions with substrate '0' and reagent '1'. The birth set is analagous to diffusion rate and the survival set is analagous to reaction rate. The analysis includes categorisations based on qualitative factors like the features and density of resulting configurations and quantitative factors like the outcome of glider collisions within each universe. For example, the **P**-class contains rules with high diffusion rate (i.e. wide birth interval) and low reaction rates (i.e. narrow survival interval) which produce large regions of 0-state and 1-state each containing scatterings of the other within them. These patterns are qualitatively distinct from, for example, **O**-class rules which have low diffusion rate and high reaction rate producing irregular spotted patterns. Despite the depth of this investigation, the 1296 CA rules analysed cover less that 0.05% of all life-like CA. The broader issue in both Wolfram's and Adamatzky's classifications is the lack of objective distinction between class boundaries which makes it difficult to predict the behaviour of rules *a priori*. Indeed, some automata have been

proven to span multiple classes[16].

This dilemma is alleviated to some degree by Eppstein's four-way classification[5] which is based on strict definitions of *fertility* and *mortality*. A rule is fertile if there exists a finite pattern that eventually escapes any bounding box B. Note this is symmetrically opposite to the definition of periodicity since any infertile rule can only iterate through $2^{|B|}$ steps before repeating a previous state. A rule is mortal if it supports a pattern which transitions to the *quiescent* state (i.e no live cells) in the next time step. Eppstein conjectures that "interesting" behaviour arises out of rules that are both fertile and mortal. Figure 3.2 depicts a schematic map of his analysis.

This work provides a strong theoretical foundation to guide our search of life-like CA and to verify that our techniques are effective on different varieties. However, they are not grounded in a systematic statistical search which makes it difficult to ascertain the proportion of each category that exist in contested regions. For example, we may be interested in the ratio of fertile to infertile configurations for rule B3/S01. Although a closed-form solution for this ratio is infeasible, it is possible to come to an approximation through simulation. As mentioned in the preliminaries, large scale simulations of random initial conditions on particular rules have proven to be an effective way of identifying new patterns [CITE]. This is called soup searching. In a similar vein, we will use soup searches to approximate the fertility and periodicity of all rules in the life-like CA rulespace.

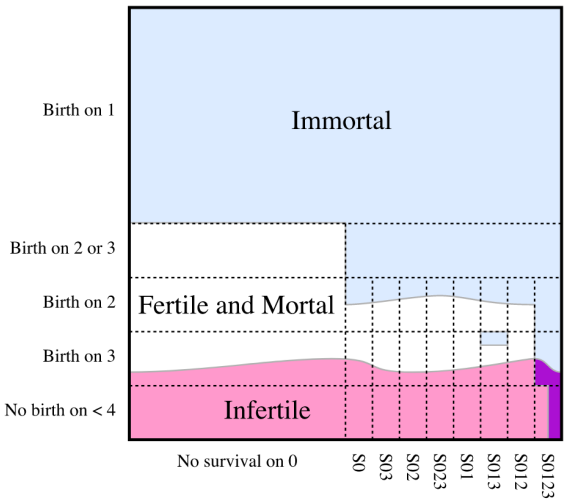


Figure 3.2: Map of fertile, infertile, mortal, and immortal regions in binary-state RDCA rulespace [5]

3.1.2 Learning

A seminal work by Meyer et al.[17] looks at learning 2D CA neighbourhood functions using genetic algorithms. Later works by Mitchell et al. [18] explore the effectiveness of genetic algorithms in learning entire transition functions but only in the domain of elementary cellular automata. Around the same time, Koza et al. make leaps by applying genetic programming to a broad variety of tasks including the CA majority classification problem [19]. Incremental improvements have been made to since then with Breukelaar and Bäck notably delivering experimental evidence that the CA inverse design problem using evolutionary computation is more tractable in higher dimensions[20]. We delve briefly into some of these papers to compare their aims, methods, and outcomes.

Learning Neighbourhood Functions

In *Learning Algorithm for Modelling Complex Spatial Dynamics (Meyer et al., 1989)*[17], the neighbourhood function of a binary probabilistic cellular automaton (PCA) was evolved to model artificially generated datasets. The motivation was to establish a CA architecture capable of codifying patterns in physical interactions directly from experimental data. It was successful to

this end as Richards et al.[21] used results from this work to predict the dendritic solidification structure of NH_4BR.

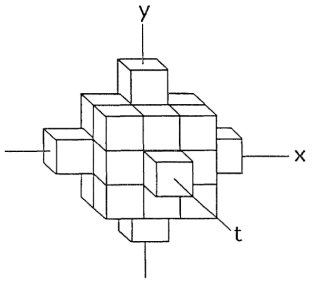


Figure 3.3: 20-cell, two step neighbourhood in space and time

Meyer's genetic algorithm seeks solutions within a 20-cell vicinity where each cell can be included or excluded from the neighbourhood set. It is the intersection of the Moore neighbourhood in time step $t-1$ and the von Neumann neighbourhood of range 2 in time step $t-2$ as visualised in Figure 3.3. The full 20-cell neighbourhood is called the master template and each chromosome encodes some subtemplate $s_1, ..., s_m$. The fitness function used is

$$F = I - \frac{2^m}{N}$$

$$\text{where } I = \sum P(s, s_1, ..., s_m) \log_2 \frac{P(s, s_1, ..., s_m)}{P(s)P(s_1, ..., s_m)}$$

Here, I is the mutual information of the subtemplate and represents the amount of information, measured in Shannon bits, that can be obtained about the value of the central cell from subtemplate states. It is calculated by summing across all 2^m configurations of the subtemplate in the data and across both values of $s \in \{0, 1\}$. The second term in the fitness function ensures that subtemplates of varying sizes are treated appropriately by proportionately penalising large subtemplates that, by nature, will contain more information. $N = 20$ is the size of the master template

The genetic algorithm initialises the population at a randomly chosen subset of possible subtemplates. Selection is performed using a truncated linear ranking. Crossover is applied using an arbitrary cut in space-time on the master template as the crossover point. Point mutation is applied by either adding or removing a single cell from each candidate. This process is iterated to converge towards an optimum.

As the first notable exploration of learning CA properties with genetic algorithms, this paper demonstrates the ability of GAs to efficiently traverse an opaque search space. The algorithm precisely learns neighbourhoods interior to the master template such as the 1 time step Moore neighbourhood and even when the objective neighbourhood lies partially outside the master template, the algorithm successfully finds a close approximation. For example, when given data produced by a 1 time step von Neumann neighbourhood, the algorithm learns a neighbourhood set that produces correct behaviour 96% of the time.

This work also raises many questions for future research. The most pertinent is whether it is possible to link learned rules to existing and future theoretical models. Moreover, this work only explores binary state CA but application of similar techniques on continous-state CA could closer approximate the partial differential equations that underlie the physical processes being modelled.

Finally, this paper focuses on optimising the neighbourhood set of the CA model only. It aims to establish *which* parameters in a local vicinity of a current cell are most relevant to predicting the future state, not *how* those parameters are combined and transformed to produce the result. In this thesis, we are interested in going beyond this and approximating the full transition function. In some cases we will fix the neighbourhood function used to reduce our search space under the assumption that techniques from this paper can be used to find optimal sub-neighbourhoods if they exist.

Learning 1D Transition Functions

There are a number of problems in elementary CA computation that have piqued academic interest from both analytical and computation angles. One example is the firing gun synchronisation problem[22] which seeks a rule that minimizes the time to get a CA from a quiescent state (all 0) to a firing state (all 1). Another is the density classification problem, or majority problem, which aims to find a binary elementary CA rule that accurately performs majority voting. That is, all cells converging to the state that dominates the initial condition. Despite their simple formulation, both of these problems require the transfer of information through compressed, latent representations and a global consensus based on localised computations. This makes them useful benchmarks when measuring the capability of CAs and the algorithms used to design them. For the sake of brevity, we focus only on the majority problem in this section. We formalise it as follows.

Definition 3.1 (Majority Problem)**.** *An elementary CA of size N solves the majority problem for some initial conditions* $\{\sigma_i\}_{i=1}^{N}$ *if* $\exists T$ *s.t.* $\forall t > T$*:*

$$\sigma_i(t) = \begin{cases} 0, & \sum_{i=1}^{N} \sigma_i(0) < \frac{N}{2} \\ 1, & \sum_{i=1}^{N} \sigma_i(0) > \frac{N}{2} \end{cases}$$

The desired result is undefined if the initial state contains an equal number of 0 and 1 cells.

The Gacs-Kurdyumov-Levin (GKL) rule is a human-designed solution to solve this problem. The function, as defined below, allows consensus to be reached in O(n) time and, for n=149, acheives success on 81.6% of inputs[23]. Modifications throughout the 1990s incrementally improved this classifier[24]. Although these were very promising, the human designed aspect of these algorithms meant there was little to support their optimality compared to others in the rulespace.

Definition 3.2 (GKL Classifier)**.** *A GKL density classifier is an elementary CA on periodic boundary conditions with transition function*

$$\sigma_i(t+1) = \begin{cases} Mo(\sigma_{i-3}(t) + \sigma_{i-1}(t) + \sigma_i(t)), & \sigma_i(t) = 0 \\ Mo(\sigma_i(t) + \sigma_{i+1}(t) + \sigma_{i+3}(t)), & \sigma_i(t) = 1 \end{cases}$$

where $Mo(.)$ *returns the mode of its arguments.*

A seminal series of work by Mitchell, Crutchfield, and Das[18] tackled this issue by automating the process of CA transition function design through evolutionary computation. These works made effective use of genetic algorithms operating on fixed length bitstrings. Rules with radius $r = 3$ were considered leading to chromosomes of length $2^{2r+1} = 128$. The size of the rulespace was therefore 2^{128} which eliminates the possibility of any exhaustive search. The size of the CA itself was $N = 149$, chosen to be odd so that the solution to the majority problem is well defined. Upon initialisation, 100 chromosomes are chosen from a distribution that is uniform over chromosome density. This can be viewed as picking the binary representations of 100 samples from the $Binomial(64, [TOCALCULATE])$[PROOF] distribution. This is markedly distinct from the usual unbiased distribution which assigns each bit in the chromosome to 0 or 1 with probability 0.5, equivalent to picking 100 samples from the $Uniform(0, 128)$ distribution. The choice of binomial initialisiation has been shown to considerably improve performance[CITE].At each generation, 100 new initial conditions (ICs) were created and fitness was defined as the percentage of correctly classified ICs. This stochastic fitness function was effective at reducing overfitting. A $(\mu + \lambda)$ selection method was employed with $\mu = 20$ and $\lambda = 80$ and mutation was performed with a two-point crossover. Although not as accurate as GKL, the discovered solution still achieves a 76.9% accuracy. However, the evolved solutions were not very sophisticated, mostly falling into the category of "block-expanding algorithms"[FIGURE].

A later work by Andre et al.[19] uses genetic programming to achieve superior results qualitatively and quantitatively. The obtained solution uses various internal representations of density[FIGURE] to transfer and collate information across the automaton. It attains an accuracy of ∼82.3%. Recently, it was proven that a perfect density classification rule for an infinite CA in any dimension, stochastic or determistic, is impossible[25]. However, evolutionary computation still surprises in its ability to find approximations of ever-increasing performance.

3.2 Continuous Reaction-Diffusion CA

Chapter 4

Design

In this chapter we outline the workings of the evolutionary algorithm toolkit.

4.1 Evolutionary Algorithm Toolkit

4.2 Cellular Automata Simulator

Chapter 5

Method

5.1 Life-like CA

5.1.1 Genetic Algorithm

5.1.2 Species Analysis

5.2 Reaction-diffusion CA

5.2.1 Evolutionary Strategies

5.2.2 Particle Swarm Optimisation

Chapter 6

Application

6.1 Maze Generation

6.1.1 Region Merging Algorithm

6.1.2 Fitness Evaluation

6.1.3 Selection and Quality-Diversity

Chapter 7

Evaluation

Chapter 8

Conclusions

Chapter 9

Ethical Considerations

Bibliography

[1] Debasis Das. A survey on cellular automata and its applications. volume 269, 12 2011. ISBN 978-3-642-29218-7. doi: 10.1007/978-3-642-29219-4_84.

[2] Alan Dorin, Jonathan McCabe, Jon McCormack, Gordon Monro, and Mitchell Whitelaw. A framework for understanding generative art. *Digital Creativity*, 23, 12 2012. doi: 10.1080/14626268.2012.709940.

[3] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, 2002. ISBN 1579550088. URL https://www.wolframscience.com.

[4] Andrew Adamatzky, Genaro Juárez Martínez, and Juan Carlos Seck Tuoh Mora. Phenomenology of reaction–diffusion binary-state cellular automata. *International Journal of Bifurcation and Chaos*, 16(10):2985–3005, 2006.

[5] David Eppstein. Growth and decay in life-like cellular automata. In *Game of Life cellular automata*, pages 71–97. Springer, 2010.

[6] L Hernández Encinas, S Hoya White, A Martín Del Rey, and G Rodríguez Sánchez. Modelling forest fire spread using hexagonal cellular automata. *Applied mathematical modelling*, 31(6):1213–1227, 2007.

[7] Adam P Goucher. Gliders in cellular automata on penrose tilings. *Journal of Cellular Automata*, 7, 2012.

[8] Wenzhong Shi and Matthew Yick Cheung Pang. Development of voronoi-based cellular automata-an integrated dynamic model for geographical information systems. *International Journal of Geographical Information Science*, 14(5):455–474, 2000.

[9] Marco Bartolozzi and Anthony William Thomas. Stochastic cellular automata model for stock market dynamics. *Physical review E*, 69(4):046112, 2004.

[10] Armin R Mikler, Sangeeta Venkatachalam, and Kaja Abbas. Modeling infectious diseases using global stochastic cellular automata. *Journal of Biological Systems*, 13(04):421–439, 2005.

[11] Martin Gardner. The fantastic combinations of jhon conway's new solitaire game'life. *Sc. Am.*, 223:20–123, 1970.

[12] Stephen Wolfram. Theory and applications of cellular automata. *World Scientific*, 1986.

[13] Matthew Cook et al. Universality in elementary cellular automata. *Complex systems*, 15(1):1–40, 2004.

[14] Norman H Packard and Stephen Wolfram. Two-dimensional cellular automata. *Journal of Statistical physics*, 38(5):901–946, 1985.

[15] Takeo Yaku. The constructibility of a configuration in a cellular automaton. *Journal of Computer and System Sciences*, 7(5):481–496, 1973.

[16] John T Baldwin and Saharon Shelah. On the classi ability of cellular automata. *TCS*, 1999.

[17] Thomas P Meyer, Fred C Richards, and Norman H Packard. Learning algorithm for modeling complex spatial dynamics. *Physical review letters*, 63(16):1735, 1989.

[18] Melanie Mitchell, James P Crutchfield, Rajarshi Das, et al. Evolving cellular automata with genetic algorithms: A review of recent work. In *Proceedings of the First international conference on evolutionary computation and its applications (EvCA'96)*, volume 8. Moscow, 1996.

[19] David Andre, Forrest H Bennett III, and John R Koza. Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. *Genetic programming*, 96:3–11, 1996.

[20] Ron Breukelaar and Th Bäck. Using a genetic algorithm to evolve behavior in multi dimensional cellular automata: emergence of behavior. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 107–114, 2005.

[21] Fred C Richards, Thomas P Meyer, and Norman H Packard. Extracting cellular automaton rules directly from experimental data. *Physica D: Nonlinear Phenomena*, 45(1-3):189–202, 1990.

[22] Edward F Moore. The firing squad synchronization problem. *Sequential machines, selected Papers*, pages 213–214, 1964.

[23] Péter Gács, Georgy L Kurdyumov, and Leonid Anatolevich Levin. One-dimensional uniform arrays that wash out finite islands. *Problemy Peredachi Informatsii*, 14(3):92–96, 1978.

[24] Rajarshi Das, James P Crutchfield, Melanie Mitchell, and James M Hanson. Evolving globally synchronized cellular automata. 1995.

[25] Ana Bušić, Nazim Fates, Jean Mairesse, and Irene Marcovici. Density classification on infinite lattices and trees. In *Latin American Symposium on Theoretical Informatics*, pages 109–120. Springer, 2012.