Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Algoritmos genéticos para famílias de autómatos celulares em dimensão 1

João Pedro Martins Cardoso

Orientador(es) | Carlos Correia Ramos

Lígia Maria Ferreira

Évora 2023

Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Algoritmos genéticos para famílias de autómatos celulares em dimensão 1

João Pedro Martins Cardoso

Orientador(es) | Carlos Correia Ramos

Lígia Maria Ferreira

Évora 2023

A dissertação foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor da Escola de Ciências e Tecnologia:

Presidente	|	Teresa Gonçalves (Universidade de Évora)
Vogais	|	Carlos Correia Ramos (Universidade de Évora) (Orientador) Irene Pimenta Rodrigues (Universidade de Évora) (Arguente)

Évora 2023

À memória de minha tia, que sempre me apoiou e que queria imenso assistir à minha conclusão do curso

Agradecimentos

A realização deste trabalho contou com importantes apoios, incentivos e ajuda de diversas pessoas, às quais sou eternamente grato.

Primeiramente quero agradecer à minha mãe que foi sempre me deu um apoio e ajuda fundamental durante todo este percurso.

Quero agradecer também aos meus avós por todo o apoio e condições dadas para que pudesse concluir esta etapa da minha vida.

Agradeço também à minha namorada por todo o apoio, motivação, ajuda e por todas as palavras de apoio transmitidas durante este longo percurso, foi mesmo muito importante.

Quero agradecer também ao meu amigo Samuel Tavares, que mesmo longe me ajudou na realização deste trabalho.

Um agradecimento especial também ao professor Carlos Ramos e à professora Lígia Ferreira, pelo total apoio, disponibilidade e ajuda ao longo da realização deste trabalho.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AC Autómato Celular

DH Distância de *Hamming*

EQM Erro Quadrático Médio

IT Iteração Temporal

MSE *Mean Square Error*

SIFT *Scale-Invariant Feature Transform*

Sumário

Os autómatos celulares são sistemas dinâmicos e discretos que podem ser interpretados como programas simples que desempenham funções computacionais ou como organismos digitais que exibem comportamentos característicos. De um ponto de vista biológico, cada autómato celular é caracterizado por um código similar ao código genético biológico - o genótipo - código esse que vai definir as regras para a realização do autómato celular, tal como acontece com os organismos biológicos.

O genótipo do autómato celular vai definir as regras de transição do autómato celular, sendo que a concretização destas regras vai resultar na realização do autómato celular, que é representada num diagrama espaço-tempo. Esta realização vai apresentar, visualmente, padrões e características únicas dependendo do genótipo e das condições iniciais, sendo que esta realização pode também ser, analogamente, relacionada com a biologia, sendo definida como o fenótipo do autómato celular.

O presente trabalho tem como principal objetivo estudar as dinâmicas evolutivas dos autómatos celulares quando sujeitos a operações genéticas, como mutação, replicação e recombinação. Para este estudo foi necessário implementar em *Python* um algoritmo que permitisse gerar e comparar fenótipos de diferentes autómatos celulares (a partir dos genótipos) para perceber de que forma as operações genéticas influenciam a estabilidade do fenótipo, ou seja, de que forma os padrões e características visuais são afetadas quando sujeitas a estas operações genéticas.

Foi também implementado um algoritmo para estudar a evolução de populações de autómatos celulares que utiliza a operação genética de recombinação para evoluir uma determinada população e que permite perceber de que forma os genótipos e fenótipos evoluem em determinadas populações com diferentes características. Estas diferentes características de populações estão associadas à proximidade inicial dos genótipos, ou seja, podem existir populações em que os genótipos vão ser muito próximos inicialmente e populações em que os genótipos vão ser muito variados, sendo posteriormente analisado de que forma as características das populações subsistem e evoluem. Nesta dissertação vão ser estudados estes dois casos de populações e também um caso em que os genótipos na população são muito próximos inicialmente mas que contém um invasor, ou seja, foi introduzido um genótipo muito diferente dos existentes na população e é estudado como os genótipos e fenótipos dos descendentes são afetados.

Palavras chave: Autómatos celulares, Mutação, Algoritmos de recombinação, Dimensão 1, Vida artificial

Abstract

Genetic algorithms for cellular automata families in dimension 1

Cellular automata are dynamic, discrete systems that can be interpreted as simple programs that perform computational functions or as digital organisms that exhibit characteristic behaviors. From a biological point of view, each cellular automaton is characterized by a code similar to the biological genetic code - the genotype - code that will define the rules for the realization of the cellular automaton, as with biological organisms.

The genotype of the cellular automaton will define the transition rules of the cellular automaton, and the realization of these rules will result in the realization of the cellular automaton, which is represented in a space-time diagram. This realization will visually present unique patterns and characteristics depending on the genotype and the initial conditions, and this realization can also be, analogously, related to biology, being defined as the phenotype of the cellular automaton.

The main objective of this work is to study the evolutionary dynamics of cellular automata when subjected to genetic operations, such as mutation, replication and recombination. For this study it was necessary to implement in Python an algorithm that would allow to generate and compare phenotypes of different cellular automata (from the genotypes) in order to understand how the genetic operations influence the stability of the phenotype, that is, how the patterns and visual characteristics are affected when subjected to these genetic operations.

An algorithm was also implemented to study the evolution of populations of cellular automata that uses the genetic operation of recombination to evolve a given population and that allows us to understand how genotypes and phenotypes evolve in certain populations with different characteristics. These different characteristics of populations are associated with the initial proximity of the genotypes, that is, there may be populations in which the genotypes will be very close initially and populations in which the genotypes will be very different from each other, being later analyzed how the characteristics of the populations subsist and evolve. In this dissertation will be studied these two cases of populations and also a case in which the genotypes in the population are very close initially, but that contains an invader, that is, a genotype very

different from those existing in the population was introduced and it is studied how the genotypes and phenotypes of the descendants are affected.

Keywords: Cellular automata, Mutation, Recombination algorithms, Dimension 1, Artificial life

1

Introdução

Os autómatos celulares são modelos matemáticos simples que consistem numa matriz de células (onde cada uma das células apresenta um estado específico e individual), de dimensão arbitrária e que apresentam uma evolução temporal de acordo com a aplicação de regras simples e determinísticas. Cada uma destas células está num estado específico e o estado da mesma é atualizado em cada iteração, dependendo do estado das células vizinhas e das regras de transição existentes.

Os autómatos celulares são normalmente definidos como sistemas dinâmicos e discretos que permitem descrever e modelar sistemas complexos [RBTF21] e que permitem entender como determinados padrões e estruturas surgem em sistemas naturais e artificiais. Existe uma ampla variedade de aplicações práticas dos autómatos celulares, destacando-se as seguintes áreas:

- Física - Os autómatos celulares são frequentemente utilizados para simular fenómenos físicos, como por exemplo a dinâmica de fluidos [RCC22], [CM99], fluidos em meios porosos heterogéneos, [KP20] e até mesmo a dinâmica de dunas, com a modelação e quantificação de padrões complexos em dunas [NZRC09], entre outros.
- Biologia - Na área da biologia os autómatos celulares podem ser utilizados, por exemplo, para modelar o crescimento de células [LZ19], a dinâmica de populações e a evolução [RR09].

- Criptografia - Os autómatos celulares também podem ser utilizados na área da criptografia, surgindo por exemplo como uma solução para gerar sequências aleatórias [Wol86b], na implementação de cifras de fluxo baseadas em autómatos celulares [Wol86a] e na geração de número aleatórios [HCC+12], [BD19].

Nesta dissertação o foco será direcionado aos autómatos celulares de uma dimensão (autómatos celulares unidimensionais), que são normalmente representados visualmente num diagrama espaço-tempo. No topo do diagrama vamos ter a configuração inicial do autómato celular e cada uma das configurações resultantes da evolução temporal do autómato são representadas uma após a outra (de cima para baixo), originando uma imagem da evolução temporal do autómato.

Os autómatos celulares mais simples unidimensionais podem ser chamados de autómatos celulares binários e apenas têm dois estados possíveis, 0 ou 1. Na representação de autómatos celulares unidimensionais binários num diagrama espaço-tempo normalmente as células com o valor 0 são representadas com a cor branca e o estado 1 é representado com a cor preta.

As regras de transição locais utilizadas para atualizar o estado de cada célula são definidas por uma tabela de regras, onde para cada configuração possível da vizinhança é definido o próximo estado da célula, sendo que na figura 1.1 podemos encontrar um exemplo concreto desta tabela de regras, neste caso para o autómato celular definido pela sequência 01111000. Neste exemplo é especificado todas as combinações possíveis de vizinhança com o seu valor correspondente, por exemplo, a vizinhança de células 000 vai originar uma célula de estado 0, enquanto a sequência 001 vai originar uma célula de estado 1.

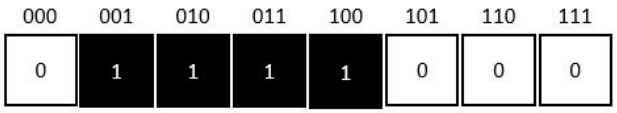


Figura 1.1: Regras de transição locais do autómato celular definido pela sequência 01111000

Na figura 1.2 encontra-se de forma mais detalhada a forma como funciona a aplicação das regras de transição locais. Para uma determinada célula é considerado o valor dessa própria célula e das células em seu redor (vizinhança) para que, de acordo com a correspondência na tabela de regras de transição locais, seja atribuído o valor da célula na iteração temporal seguinte. Neste exemplo, a sequência de células 011 vai resultar numa célula de valor 1 e a sequência 111 vai resultar numa célula de valor 0 , pela tabela de regras de transição locais definida na figura 1.1.

Figura 1.2: Regras de transição locais detalhadas

Na figura 1.3 temos o exemplo de um diagrama espaço-tempo proveniente da realização de um autómato celular binário, definido pela sequência 01111000. Como referido anteriormente, no topo do diagrama espaço-tempo temos a condição inicial do autómato celular (00000000000000001000000000000000), com a cor preta a representar as células com o estado 1 e a cor branca a representar as células com o estado 0.

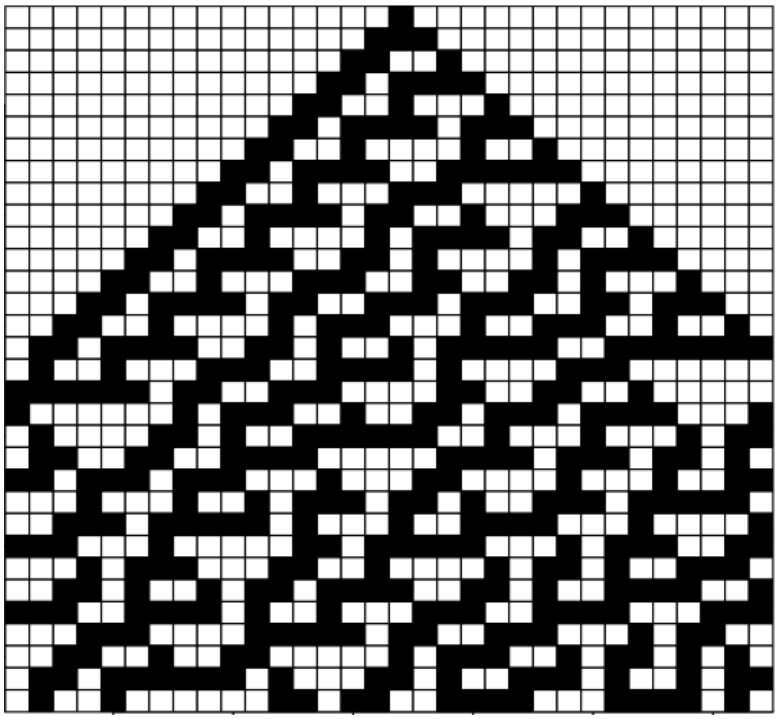


Figura 1.3: Diagrama espaço-tempo de um autómato celular definido pela sequência 01111000

1.1 Definição de autómato celular

Um autómato celular possui três elementos que são fundamentais para a sua caracterização - a dimensão, os estados possíveis para cada célula e a vizinhança das células.

- A dimensão é representada pela letra d e especifica a dimensionalidade das configurações do autómato e a disposição das células, por exemplo, um autómato unidimensional é representado por uma linha, com cada célula a apresentar vizinhos à esquerda e direita enquanto um autómato celular bidimensional é representado por um plano, com cada célula a apresentar vizinhos à esquerda, direita, cima, baixo e/ou nas diagonais [Kun03].
- O número de estados locais possíveis para cada célula, n, é o número de estados que uma célula pode apresentar, sendo que o número mínimo de estados possíveis para representar uma célula é dois, ou seja, num autómato celular é necessariamente $n \geq 2$.
- Para definir o tamanho da vizinhança (representada pela letra m e que define o número de células em redor de uma determinada célula, incluindo neste número a célula central em observação) é necessário saber o número de células vizinhas que existem em cada direção (*radius*), sendo este atributo representado pela letra r e vai definir o número de células em cada direção que vai afetar o estado futuro da célula. Num autómato celular com $d = 1$ (unidimensional) com um raio de células vizinhas r, vamos ter uma vizinhança de tamanho $m = 2r + 1$, por exemplo, se tivermos $r = 2$, vamos ter uma vizinhança de tamanho $m = 2 \times 2 + 1 \Leftrightarrow m = 5$.

É também possível encontrar autómatos celular cuja vizinhança é assimétrica em relação à posição da célula que está a ser atualizada, por exemplo um autómato celular com uma vizinhança $m = 4$. Nestes casos não existe um número de células vizinhas simétrica para ambas as direções, ou seja, o atributo r não vai ser um número inteiro.

Para resolver estes casos o que vai ser escolhido para ser utilizado para todos os estudos desta dissertação, é uma estrutura de vizinhança "*balanced pair*", que vai ser definida na subsecção 1.1.1.

As regras de transição locais são uma condicionante determinante num autómato celular, uma vez que são estas regras de transição que vão especificar os estados das células num determinado instante temporal t , baseando-se para isso nos estados das células da vizinhança no instante $t-1$ [VR09].

A aplicação destas regras de transição locais, vai definir o estado das células em cada instante e é efetuada em momentos discretos por iterações temporais, $(t=0,1,2...,t\geq 0)$, sendo o tempo $t=0$ o momento inicial, ou seja, a configuração inicial do autómato celular.

1.1.1 Definição formal

O conjunto $\mathbb{Z}_n=\{0,1,2,...,n-1\}$, $n>1$, é o conjunto de estados locais. Um autómato celular específico é definido à custa de uma aplicação local $\phi:\mathbb{Z}_n^m\to\mathbb{Z}_n$, para certo $m\geq 1$, que determina a evolução de uma vizinhança ou configuração local, por iteração. O número m é o número de células que constitui a vizinhança para o autómato celular e os elementos de $\mathbb{Z}_n^m$ são chamados de configurações locais ou vizinhanças. Para definir a dinâmica global do autómato é necessário especificar como as vizinhaças são colocadas no novo estado global. Esta especificação é dada através do par $(m_-,m_+)\in\mathbb{N}_0^2$, chamado par de vizinhança, tal que $m_-+1+m_+=m$, com m_- a ser o número de células escolhidas à esquerda da célula central em observação e m_+ o número de células escolhidas à direita da célula central em atualização. Uma escolha particular de (m_-,m_+), é chamada de par balanceado, dado por:

$$m_-=\begin{cases}\frac{m}{2}, & \text{se } m \text{ é par}\\ \frac{m-1}{2}, & \text{se } m \text{ é ímpar}\end{cases}$$

$$m_+=\begin{cases}\frac{m}{2}-1, & \text{se } m \text{ é par}\\ \frac{m-1}{2}, & \text{se } m \text{ é ímpar}\end{cases}$$

Outro tipo de par de vizinhança utilizado para certas aplicações é $(m_-,m_+)=(m-1,1)$.

A aplicação que determina a dinâmica e é chamado de autómato celular é definida, [RBTF23], [RCC22], por

$$\Phi_\Gamma:\mathbb{Z}_n^{\mathbb{I}}\to\mathbb{Z}_n^{\mathbb{I}}$$

$$\Phi_\Gamma(s):=\left(\phi\left(s_{[i-m_-,i+m_+]}\right)\right)_{i\in\mathbb{I}},$$

onde $\mathbb{I}$ pode ser $\mathbb{Z}$, $\mathbb{N}$ ou um conjunto finito$\{1,2,...,L\}$. Neste trabalho será esta última utilizada, um conjunto finito. É necessário especificar condições fronteira para que o autómato celular fique bem definido, $\Gamma=\Gamma_{(m_-,m_+)}$, estas dependem da escolha do par (m_-,m_+). As condições fronteira são

$$\Gamma_{(m_-,m_+)}=(\Gamma_-(t),\Gamma_+(t))\in\mathbb{Z}_n^{m_-}\oplus\mathbb{Z}_n^{m_+}$$

com

$$\Gamma_-(t)=\left(\gamma_{-m_-+1},...,\gamma_{-1},\gamma_0\right)\in\mathbb{Z}_n^{m_-},\Gamma_+(t)=\left(\gamma_1,...,\gamma_{m_+}\right)\in\mathbb{Z}_n^{m_+}$$

O estado global do sistema é uma sequência em $\mathbb{Z}_n^L$ e a evolução temporal do estado do sistema, através do automato celular é dado pela iteração da aplicação global Φ_Γ,

$$s(t)=\Phi_\Gamma(s(t-1)),t\geq 0$$
$$s(0)=(s_i(0))_{i=1}^N\in\mathbb{Z}_n^L.$$

Cada sequência que codifica um autómato é escrita em n símbolos. A ordem na sequência tem significado funcional, pois o primeiro símbolo na sequência corresponde à imagem pela aplicação local ϕ do autómato da vizinhança $000..0$ (tamanho m), que em base n corresponde à posição 0 da sequência, primeira posição. Deste modo o símbolo na posição $i \in \{0, ..., n^m - 1\}$ da sequência que codifica o autómato será imagem da vizinhança $i_0 i_1 ... i_{m-1}$ pela aplicação local, ou seja, $\phi(i_0 i_1 ... i_m)$. Além disso, a relação de i e a vizinhança através da expansão em base n,

$$i = i_0 n^{m-1} + i_1 n^{m-2} + \cdots + i_{m-2} n + i_{m-1}$$

A tabela de regras locais vai especificar qual o estado que uma determinada célula vai apresentar para cada uma das configurações de vizinhança possíveis, sendo o número de configurações de vizinhança possíveis igual a n^m, onde n representa o número de estados possíveis e m o número de células da vizinhança. Por exemplo, num autómato celular unidimensional ($d = 1$), com $n = 2$, e $r = 2 \Rightarrow m = 5$, vamos ter $2^5 = 32$ configurações de vizinhança possíveis, sendo que vamos ter n^{n^m} tabelas de regras de transição possíveis, neste exemplo teríamos $2^{32} = 1024$ tabelas possíveis.

De uma forma mais resumida, a sequência que vai definir a tabela de regras de transição locais vai ser uma sequência $\alpha = \alpha_0 \alpha_1 ... \alpha_{n^m - 1} \in \mathbb{Z}_n^{n^m}$, com $n \in \mathbb{N}$, ou seja, o tamanho da sequência vai ser $N = n^m$. O espaço das regras de transição locais do autómato celular é definido por $\mathcal{G}$ e o espaço das regras que possuem n estados possíveis para cada célula e com um tamanho de vizinhança m, é definido por $\mathcal{G}_{n,m}$ [RCC22].

1.1.2 Numeração de Wolfram

Para uma representação mais compacta da tabela de regras de transição pode ser utilizada a numeração de Wolfram, que consiste na representação das regras de transição na forma de um número inteiro que quando convertido para base n, dá-nos a sequência em n símbolos que codifica as regras de transição do autómato celular, podendo ser compactadas do seguinte modo:

Pode ser interpretada como um numeral expresso em base n, reordenando a sequência de modo a que o dígito mais significativo do numeral seja o último símbolo da sequência. Em seguida o numeral pode ser expresso numa base numérica de valor mais elevado do que n, deste modo a sequência reduz o seu comprimento. A chamada numeração de Wolfram corresponde ao numeral em base 10. Por exemplo, partindo do exemplo da figura 1.1, onde a sequência 01111000 vai codificar o autómato celular, o numeral em binário será 00011110 e o número de Wolfram será o numeral em base decimal

$$30 = 0 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0.$$

Em base hexadecimal ter-se ia uma maior redução, no caso acima,

$$00011110_2 = 0001\ 1110 = 1\ E = 1E_{16}$$

1.2 Condições fronteira

Como já explicado, para aplicar as regras de transição locais é necessário inicialmente conhecer qual é a vizinhança da célula a que se pretende aplicar a transição, por exemplo, se tivermos um $r = 1$ implica que vamos estar perante uma vizinhança $m = 3$, ou seja, centrando na célula com índice i, a vizinhança vai ser a célula com índice $i-1$ (célula à esquerda), i (própria célula) e $i+1$ (célula à direita), existindo uma regra

que vai especificar qual o estado a atribuir a esta vizinhança. Esta forma de definir a vizinhança apresenta um problema quando estamos perante as células que se encontram nos extremos do autómato, voltando ao exemplo anterior com $r = 1$, se quisermos aplicar a regra de transição na célula com índice $i = 0$, a célula à sua esquerda não existe, tal como na última célula do autómato não vai existir uma célula à sua direita.

Para resolver este problema é necessário definir a condição fronteira, ou seja, o que vai acontecer quando não existem células suficientes numa determinada direção para criar uma vizinhança, existindo três abordagens que podem ser utilizadas para definir a condição fronteira:

- Valor fixo: Com uma condição fronteira de valor fixo vamos estar perante uma situação onde é especificado o estado das células nos extremos da matriz do autómato, que se mantém inalterado em toda a evolução temporal. Desta forma sempre que tivermos, por exemplo, a aplicar as regras de transição para a célula de índice $i = 0$, e estivermos perante um $r = 2$, vamos ter uma célula de índice -1 e uma célula de índice -2 cujo estado é definido à priori em toda a evolução temporal do autómato, sendo que a mesma lógica é aplicada ao outro extremo do autómato como se pode verificar na figura 1.4.

- Reflexão: Na condição fronteira reflexiva, os extremos do autómato são refletidos até atingir o valor do raio de células r. De uma forma prática, com um $r = 1$ é necessário efetuar a reflexão apenas de uma célula, mas com um $r = 2$ já são necessárias duas células e por aí em diante. Um exemplo com um $r = 2$ e se quisermos definir o extremo inicial do autómato, a célula de índice $i = -1$ vai apresentar o estado da célula de índice $i = 0$ e a célula de índice $i = -2$ vai apresentar o estado da célula de índice $i = 1$, sendo que a mesma lógica se aplica ao outro extremo do autómato, como se pode verificar na figura 1.5.

- Periodicidade: Nas condições fronteira periódicas os dois extremos na matriz do autómato são considerados vizinhos, ou seja, o autómato é estendido e a última célula do autómato vai ter como vizinho à sua direita a célula de índice $i = 0$ e a célula de índice $i = 0$ vai ter como vizinho à esquerda a última célula do autómato, como se pode verificar na figura 1.6.

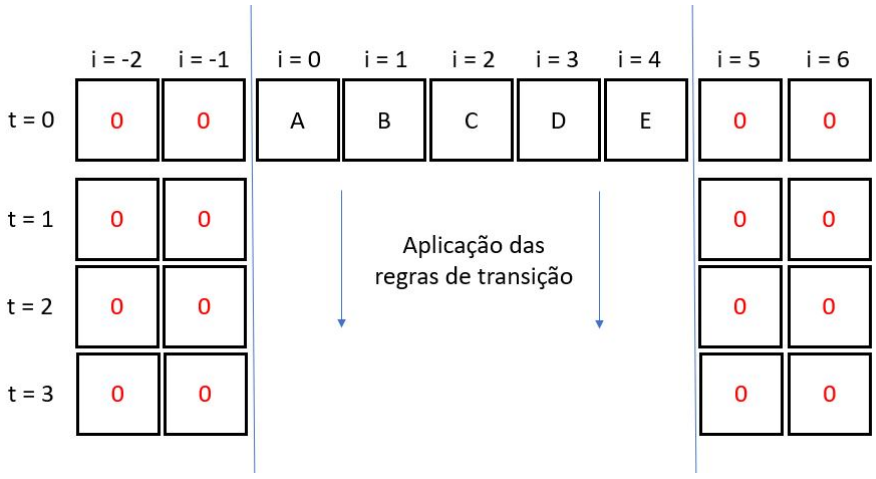


Figura 1.4: Condição fronteira - Valor fixo com $r = 2$

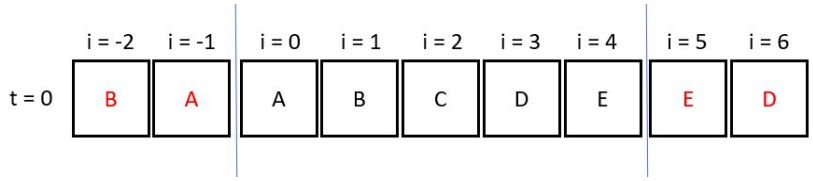


Figura 1.5: Condição fronteira - Reflexão com $r = 2$

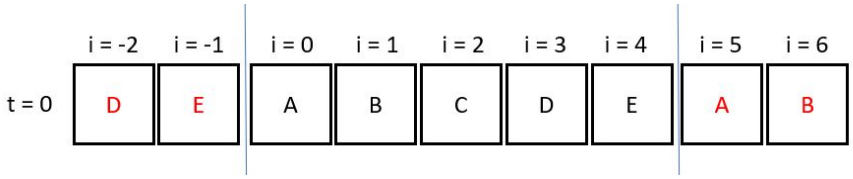


Figura 1.6: Condição fronteira - Periodicidade com $r = 2$

1.3 Perspetiva biológica

Como já foi referido, os autómatos celulares unidimensionais vão apresentar uma evolução temporal do estado global, partindo de uma condição inicial dada. A evolução dos estados globais são representados uns após outros num diagrama espaço-tempo, sendo que esta característica em conjunto com todas as características dos autómatos celulares possibilita uma analogia com a biologia.

"A evolução é um traço característico dos seres vivos, porém não se restringe aos organismos vivos" [RR09], podendo, como já referido, existir uma analogia entre as características dos autómato celulares e conceitos biológicos, sendo a mais evidente e percetível a evolução. A iteração que vai fazer o autómato evoluir temporalmente, pode ser vista como uma analogia para a evolução temporal de um ser vivo [RR09].

A analogia não fica por aqui, existindo dois conceitos biológicos que podem ser identificados no autómato celular:

- Genótipo - O genótipo refere-se à constituição genética de um ser e é o que vai determinar as características observáveis (fenótipo) desse mesmo ser, ou seja, o genótipo vai definir as características do organismo. Por esta lógica, o genótipo do autómato celular vai ser definido e representado pelas regras de transição locais, pois são elas que vão determinar as características do autómato, sendo o genótipo representado pela sequência obtida diretamente das regras de transição do autómato.
- Fenótipo - O fenótipo, como referido anteriormente, são as características observáveis de um ser, ou seja, é a expressão observável de um genótipo. O fenótipo é representado no autómato celular pelo diagrama espaço-tempo, ou seja, pelas características que o diagrama gerado por um determinado autómato celular vai apresentar, por exemplo os padrões que são gerados no diagrama.

Para um determinado genótipo (regras de transição do autómato celular) podemos ter diferentes seres/organismos dependendo das condições ambientais escolhidas, sendo que estes organismos são vistos como clones, pois possuem o mesmo código genético. As condições ambientais, por analogia, referem-se a outras condições que não a sequência das regras (genótipo) e neste caso são a condição inicial, condição fronteira ou a vizinhança do autómato celular.

Se as condições ambientais forem as mesmas os clones vão apresentar o mesmo fenótipo, contudo, se forem escolhidas condições diferentes (por exemplo, uma configuração inicial diferente para o autómato), o fenótipo pode ser diferente em ambos os clones. Contudo, os fenótipos vão ser, por norma, visualmente

muito próximos num certo sentido. Ainda assim, em casos extremos pode ocorrer clones com fenótipos visualmente muito diferentes, sendo que este fenómeno também ocorre na biologia, quando um clone cresce num ambiente muito extremo [RR09].

1.4 Objetivos

Nesta dissertação, o principal objetivo é estudar as dinâmicas evolutivas dos autómatos celulares quando sujeitos a operações genéticas, como mutação, replicação e recombinação. De forma a estudar estas dinâmicas evolutivas é necessário implementar algoritmos que permitam efetuar as operações genéticas referidas sobre autómatos celulares pré-definidos e algoritmos que permitam efetuar a comparação entre diferentes autómatos celulares e que permitam estudar de que forma a estabilidade do genótipo e do fenótipo são afetadas quando o autómato celular é sujeito a estas operações.

Os algoritmos utilizados, ao incorporar a técnica de recombinação, podem ser definidos para estabelecer um protocolo de troca de informação entre genótipos de autómatos celulares, de forma a produzir autómatos celulares com características partilhadas com os originários. Nesta dissertação também vão ser desenvolvidos estes algoritmos genéticos que permitem implementar a dinâmica de recombinação e troca de informação entre autómatos celulares.

O estudo da evolução de populações de autómatos celulares, que utiliza a dinâmica de recombinação genética para a evolução, é também um dos objetivos principais desta dissertação. São alvo deste estudo populações de autómatos celulares cujos genótipos pertençam a uma mesma família, por exemplo, uma população em que todos os elementos possuam genótipos idênticos, e de populações onde é adicionado um "invasor", ou seja, um elemento externo que não pertence a essa família de autómatos celulares. Um exemplo desta última população é uma população que apresente inicialmente todos os elementos com um genótipo muito idêntico mas que é adicionado um invasor, com um genótipo totalmente distinto dos existentes na população.

O principal objetivo no estudo de populações é perceber de que forma os genótipos e fenótipos dos elementos vão ser afetados de acordo com as características da população inicial.

1.5 Estrutura da dissertação

A dissertação encontra-se organizada em 7 capítulos. O primeiro capítulo é a introdução, o segundo capítulo apresenta o estado da arte e o terceiro e quarto capítulo complementam o estado da arte, ao apresentar diversas formas de classificar autómatos celulares (capítulo 3) e ao introduzir as dinâmicas evolutivas em autómatos celulares, com a introdução aos conceitos genéticos e operações genéticas (capítulo 4).

No capítulo 5 inicia-se o primeiro estudo da dissertação, que tem como principal objetivo estudar a estabilidade do fenótipo quando os autómatos celulares são sujeitos a operações genéticas, nomeadamente operações genéticas de mutação e replicação. O capítulo 6 apresenta o último estudo efetuado na dissertação, com o estudo da evolução de populações de autómatos celulares com a utilização do algoritmo de recombinação. Finalmente, o capítulo 7 discute as conclusões e apresenta o trabalho futuro.

2

Estado da arte

Os autómatos celulares são sistemas dinâmicos e discretos que foram introduzidos pelos matemáticos John von Neumann e Stanislaw Ulam no final dos anos 40 e são sistemas que permitem descrever e modelar sistemas complexos [RR09][RCC22]. Desde então foram desenvolvidas múltiplas aplicações nas áreas das ciências e da matemática.

2.1 História e evolução dos autómatos celulares

Desde a introdução por John von Neumann e Ulam, os autómatos celulares aparecem em diversos trabalhos científicos, sendo que em [Sar00], do autor Sarkar, é apresentada de forma mais pormenorizada a história por detrás dos autómatos celulares, sendo que o autor divide os autómatos em três categorias principais [Sar00]:

- Clássica - Na categoria clássica entram os temas que foram influenciados pelos trabalhos iniciais de John von Neumann, por exemplo, as máquinas auto reprodutoras de John von Neumann.

- Jogos - A categoria dos jogos é caracterizada pela utilização de autómatos celulares em jogos, sendo o "Jogo da Vida" de John Conway o melhor exemplo para a descrever.
- Moderna - Nesta categoria enquadram-se os trabalhos que foram influenciados pelo trabalho de Stephan Wolfram e por desenvolvimentos na área da engenharia informática.

A investigação clássica é sobretudo baseada em John von Neumann, que utilizou autómatos celulares como uma ferramenta para modelar processos biológicos de autorreprodução. Este processo foi desenvolvido através da "máquina auto reprodutora de John von Neumann", que era um autómato celular bidimensional com 29 estados possíveis para cada célula ($n = 29$) e uma vizinhança de cinco células ($m = 5$).

Este autómato celular extremamente complexo desenvolvido por John von Neumann era na verdade um computador universal e um construtor universal que, quando dada a descrição de qualquer máquina, esta máquina poderia construir essa mesma máquina [Kun03].

No ano de 1970 John Conway desenvolveu um autómato celular bidimensional que teve muito sucesso na comunidade informática, o Jogo da Vida [Cio22], sendo este trabalho um exemplo da categoria de jogos de Skar [Sar00]. O Jogo da Vida de John Conway ganhou popularidade na comunidade da informática porque apesar de ser bastante simples de programar, demonstra um comportamento emergente e auto-organizado, sendo o autómato celular mais conhecido em geral [ICS15].

O Jogo da Vida de John Conway consiste num autómato celular bidimensional com dois estados possíveis para cada célula ($n = 2$, mais concretamente a célula pode estar "viva" ou "morta") e com regras de transição aplicadas tendo em consideração as oito células vizinhas em seu redor:

- Se a célula estiver morta (estado 0) com exatamente três vizinhos vivos (estado 1) torna-se viva, ou seja, a célula nasce.
- Se a célula estiver viva com menos de dois vizinhos vivos morre por isolamento.
- Se a célula estiver viva com mais de três vizinhos vivos morre por superpopulação.
- Se a célula estiver viva com dois ou três vizinhos vivos permanece viva.

Os autómatos celulares são amplamente utilizados como heurísticas em biologia para explorar implicações e consequências de certas teorias. O Jogo da Vida de Conway permite modular um processo básico em biologia - a evolução das comunidades - permitindo simular a vida e a morte utilizando regras básicas de sobrevivência e é amplamente utilizado com este propósito [CHH16] [Cio22].

2.1.1 Wolfram

Ao aplicar sucessivamente regras de transição sobre um autómato celular o mesmo vai sofrer uma evolução temporal e vai alcançar um ponto onde é possível de forma intuitiva classificar o autómato de acordo com o seu nível de comportamento dinâmico. Em 1983, Stephan Wolfram [Wol83] fez um estudo detalhado que envolveu um grande conjunto de autómatos celulares unidimensionais que resultou na sua classificação nas quatro classes seguintes:

- Classe I - A primeira classe especifica que a evolução temporal do autómato vai dar origem a um estado homogéneo em que todas as células vão apresentar o mesmo estado independentemente das iterações temporais que se apliquem ao autómato, por exemplo na figura 2.1 podemos observar um

autómato cuja evolução deu origem a um estado homogéneo com todas as células a apresentarem o estado 0.

- Classe II - Os autómatos celulares incluídos na segunda classe de Wolfram são autómatos que formam padrões/estruturas periódicas, ou seja, em que o comportamento do autómato é cíclico ou periódico, sendo que a evolução temporal do autómato vai dar origem a blocos de células que são repetidos periodicamente ao longo da evolução. Na figura 2.2 encontra-se um exemplo concreto desta estabilização e periodicidade.

- Classe III - A terceira classe de Wolfram especifica que o autómato vai possuir um comportamento caótico e pseudo-aleatório [Cio22] onde a evolução é dominada por conjuntos de células que formam padrões caóticos e que nunca se repetem, ou seja, formam estados aperiódico ou pseudo-aleatório. A evolução temporal de um autómato de classe III vai originar um diagrama espaço-tempo imprevisível e pseudo-aleatório, como no exemplo da figura 2.3 [MSTMZ12].

- Classe IV - A quarta e última classe refere que a evolução temporal do autómato celular vai gerar estruturas complexas com evolução imprevisível [Cio22] que se podem "mover" ao longo da evolução, espacial e temporalmente, podendo ainda existir regiões uniformes, periódicas ou caóticas a coexistir com estas estruturas. Esta classe é frequentemente associada a um comportamento complexo e é comum afirmar que esta classe é a "fronteira do caos" [MSTMZ12] [Cio22]. Na figura 2.2 encontra-se um exemplo de um autómato celular de classe IV, com padrões complexos de evolução imprevisível.

A classificação dos autómatos celulares numa das quatro classes de Wolfram parece uma tarefa simples, contudo, a sua natureza qualitativa resulta em classes com fronteiras que podem ser difusas. Alguns autómatos celulares, principalmente os mais complexos e com grandes vizinhanças, vão apresentar propriedades que pertencem a mais do que uma classe, sendo que as classes III e IV são particularmente difíceis de distinguir. Na figura 2.5 encontram-se exemplos concretos para as quatro classes de classificação de autómatos celulares de Wolfram, sendo estes exemplos gerados por autómatos celulares com uma configuração inicial aleatória, com 358 células e 344 iterações temporais [MSTMZ12].

Este estudo de Stephan Wolfram, que classifica os autómatos celulares unidimensionais em quatro classes distintas, enquadra-se na categoria de investigação moderna e marca uma mudança no estudo de autómatos celulares, onde se deixou de estudar autómatos específicos e complicados para estudar grandes conjuntos de autómatos mais simples [Kun03].

Além dos trabalhos na classificação de autómatos celulares unidimensionais, também são encontrados trabalhos científicos de Stephan Wolfram sobre a utilização de autómatos celulares na área da criptografia, nomeadamente em [Wol86a] podemos encontrar o estudo da utilização de uma cifra de fluxo baseada em autómatos celulares de dimensão 1 e em [Wol86b] a utilização de autómatos celulares de dimensão 1 na geração de sequências aleatórias.

Os autómatos celulares apesar de possuírem estruturas bastante simples, as abordagens exploradas em [Wol86b] suportam a ideia de que o comportamento dos autómatos pode ser tão imprevisível que na prática o seu comportamento parece aleatório, com as sequências produzidas a apresentar um alto grau de aleatoriedade.

Figura 2.1: Regra 32 - Classe I

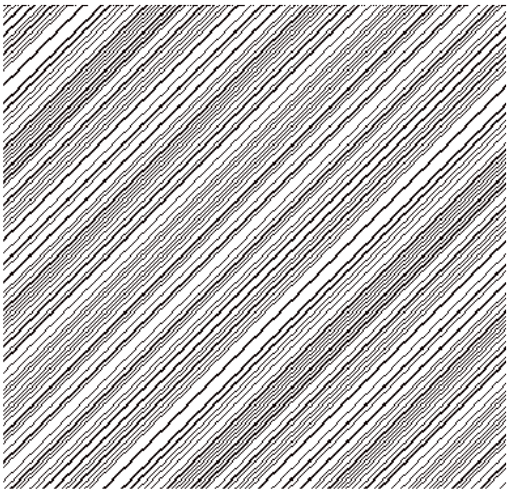


Figura 2.2: Regra 10 - Classe II

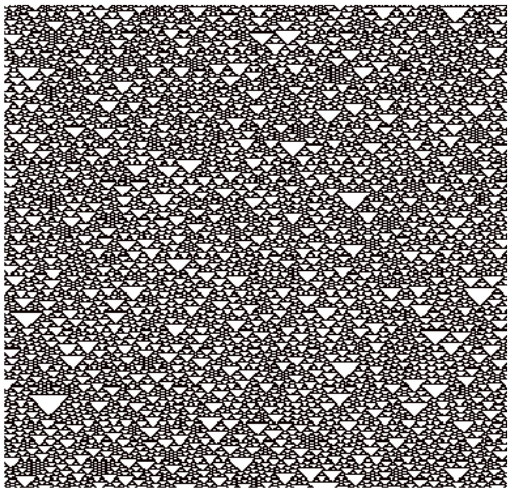


Figura 2.3: Regra 126 - Classe III

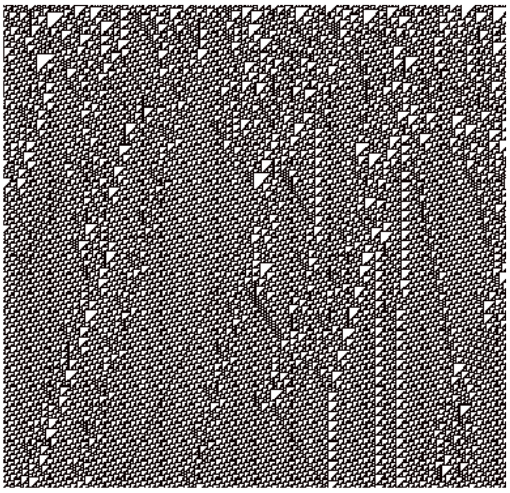


Figura 2.4: Regra 110 - Classe IV

Figura 2.5: Exemplos das quatro classes de Wolfram

2.2 Algoritmos genéticos e evolutivos em autómatos celulares

Os algoritmos genéticos são algoritmos de otimização meta-heurística inspirados pela teoria da evolução que permitem realizar operações de, por exemplo, mutação, recombinação e seleção numa população. A robustez destes algoritmos é justificada com o facto de que com a utilização de um processo evolutivo é possível alcançar soluções que à priori eram muito difíceis de prever [JLH+19].

Os conceitos de autómatos celulares e de algoritmos genéticos são, em certa medida, comparáveis e deste modo podem ser combinados num instrumento que incorpora as vantagens de ambos e que pode ser usado para encontrar novos casos de estudo interessantes. Em [CDM20] são mencionadas algumas aplicações da evolução de autómatos celulares através de algoritmos genéticos, tais como aplicações no jogo do dilema do prisioneiro.

Na referência [RR09] é possível encontrar a aplicação de conceitos genéticos no desenvolvimento de di-

nâmicas evolutivas em autómatos celulares. Neste artigo é feita uma analogia com a biologia, onde por exemplo, a aplicação sucessiva das regras de transição sobre o autómato celular pode ser visto como a evolução temporal de um ser, a regra de evolução do autómato pode ser visto como o genótipo e as diversas características resultantes da evolução temporal do autómato celular podem ser chamadas de fenótipo [RCC22].

A aplicação de algoritmos genéticos em autómatos celulares vai resultar em processos de mutação, replicação e recombinação de autómatos, onde pode ser estudado, por exemplo, a estabilidade do genótipo perante estas simples operações [RR09].

A principal vantagem da aplicação de métodos evolutivos é a obtenção eficiente de regras de autómatos celulares com características específicas, que são obtidas através da aplicação das técnicas de mutação, recombinação e *assembly* entre autómatos celulares [RCC22].

3

Classificação de autómatos celulares

A evolução temporal dos autómatos celulares unidimensionais e a sua representação em diagramas espaço--tempo vai resultar em diagramas que, quando sujeitos a análise, vão permitir estudar o comportamentos e dinâmicas da evolução temporal dos autómatos, resultando em diversos esquemas possíveis para classificar os autómatos celulares.

3.1 Classificação de Wolfram

Um dos esquemas mais conhecido para a classificação de autómatos celulares é o sistema proposto por Wolfram, que inclui quatro classes qualitativas cuja classificação é baseada na análise visual do diagrama espaço-tempo resultante da evolução temporal do autómato celular, sendo que este esquema de classificação está detalhado na secção 2.1.1.

3.1.1 Estabilidade dos autómatos celulares

Para estudar a estabilidade do autómato celular, neste caso do fenótipo do autómato, pode-se utilizar uma técnica que consiste na comparação de dois diagramas espaço-tempo, com o primeiro diagrama a corresponder ao autómato celular original e o segundo diagrama a corresponder a um clone do autómato, mas com uma diferença, no segundo autómato foi introduzida uma perturbação numa determinada posição (ou várias) da configuração inicial do autómato no instante inicial $t = 0$. Por exemplo uma alteração no estado da célula da posição $i = 0$ do estado 0 para o estado 1. A partir da comparação dos dois diagramas espaço-tempo, pode-se comparar a trajetória da evolução do autómato inicial e do autómato alterado, calculando a distância de Hamming entre cada um dos autómatos [Mel09].

Esta métrica baseia-se no facto de que os autómatos celulares são sistemas sensíveis a pequenas variações nas condições iniciais (condição para constatar que o sistema dinâmico é caótico) e são comparadas realizações de autómatos celulares em que as regras de transição locais vão ser as mesmas, mas aplicadas a autómatos celulares com uma condição ambiental diferente (configuração inicial).

A distância de Hamming é definida pelo número total de posições nas quais os estados das células do autómato original e do autómato com uma diferente condição inicial diferem entre si dividido pelo número total de posições (definido pela letra L). Para efetuar o cálculo interpretam-se os estados locais como elementos de $\mathbb{Z}_n$, utilizando a aritmética modular para o efeito:

$$DH = \frac{1}{L}[\sum_{i=1}^{L} |\sigma_i(t) - \sigma'_i(t)|]$$

Na fórmula da distância de Hamming, $\sigma_i(t)$ representa o estado na posição i no instante t do autómato celular original, $\sigma'_i(t)$ representa o estado na posição i no instante t do clone, i representa a posição da célula dentro do autómato, L é o tamanho do autómato e t representa o número da iteração temporal [Mel09].

A distância de Hamming além de possibilitar o estudo da estabilidade do fenótipo também possibilita, a partir do valor obtido, distinguir as quatro classes de Wolfram [Mel09]:

- Classe I - Na classe I a distância de Hamming torna-se nula ao longo da evolução temporal do autómato se o estado final absorvente for o mesmo nas duas realizações, caso contrário a DH vai tornar-se num valor constante ao longo das iterações.
- Classe II - Na classe II de Wolfram, a distância de Hamming vai apresentar um valor periódico ao longo das iterações temporais efetuadas sobre o autómato.
- Classe III - Como referido anteriormente, os autómatos celulares da classe III apresentam um comportamento caótico, pelo que a distância de Hamming vai apresentar um valor irregular (tendencialmente aleatório) a cada iteração temporal.
- Classe IV - Os autómatos celulares classificados como classe IV vão gerar estruturas complexas com evolução imprevisível, com a distância de Hamming a revelar padrões não regulares e não aleatórios.

3.1.2 Distâncias de Hamming ao nível do genótipo

Além do estudo da estabilidade do fenótipo e da possível distinção entre as quatro classes de Wolfram, a distância de Hamming (DH) também pode ser aplicada ao nível do genótipo e vai ser utilizada no estudo

de autómatos celulares específicos para quantificar as diferenças ao nível do genótipo entre o autómato celular original e um autómato celular que resulta da aplicação da operação genética de mutação sobre o autómato original.

Se dois genótipos forem muito similares, por exemplo, o autómato original e um autómato que resulte da mutação do autómato original numa única posição (mutação singular), os genótipos vão ser muito idênticos, com a distância de Hamming a ser a menor possível diferente de zero, ou seja 1. Contudo, se a mutação for em múltiplas posições do genótipo, os genótipos vão ser muito diferentes e a distância de Hamming vai apresentar um valor maior que 1.

Esta distância de Hamming aplicada à comparação de dois genótipos fornece o número de posições em que os dois genótipos diferem entre si, ou seja, fornece o número de bits diferentes entre os dois genótipos. O resultado da operação da distância de Hamming também pode ser definida pelo menor número de substituições necessárias para transformar o genótipo x no genótipo x^1 [Mac93].

No caso de genótipos com um número de estados $n > 2$, se a distância de Hamming for calculada como a diferença dos estados locais com a aritmética modular, mede mais do que o número de estados diferentes, mede a diferença entre eles. Por exemplo, considerando o genótipo original com a célula da posição $i = 1$ a apresentar o estado 0, se tivermos perante dois genótipos, cada um a apresentar uma mutação singular nessa mesma posição, com um deles a apresentar o estado 1 e outro o estado 2, a DH será maior no genótipo que tem a mutação singular com o estado 2. Contudo, neste trabalho a DH é calculada tendo por base o número de bits diferentes entre os dois genótipos, ou seja, no exemplo anterior ambas as mutação vão apresentar a mesma DH, pois apresentam apenas um bit diferente relativamente ao genótipo original.

Por exemplo, se considerarmos o genótipo $x^1 = 1110110$ e o genótipo $x^2 = 0110110$, a distância de Hamming d entre estes dois genótipos vai ser $d = 1$, pois apenas existe uma única diferença entre eles, na primeira posição.

3.2 Classificação de Li-Packard

Baseando-se na estrutura de classificação proposta por Wolfram, os autores Li e Packard desenvolveram um sistema de classificação para a evolução temporal dos autómatos celulares com seis classes [LP90].

A configuração limite de um autómato celular é o estado final estável (ou seja, é o conjunto de estados inseridos num ciclo em que ao longo das iterações não se sai deste estado ou ciclo) ou o ciclo de estados após um determinado número de passos.

O tamanho das configurações limite é o tempo que o autómato celular demora a atingir a configuração limite e é um fator primário determinante para classificar os autómatos celulares. Esta ideia está presente no sistema proposto por Wolfram, contudo, esta ideia é mais implícita na classificação de Li-Packard [Kun03].

Seguindo a classificação de Li-Packard um autómato celular pode ser classificado numa das seis classes seguintes [Kun03]:

- Classe Nula - Esta classe é idêntica à classe I proposta por Wolfram, onde a configuração limite é um estado homogéneo em que todas as células possuem o mesmo estado.
- Classe Ponto Fixo - Esta classe aplica-se a autómatos celulares cuja configuração limite é invariante após **uma** aplicação das regras de transição. Esta classe inclui regras que deslocam espacialmente o padrão e exclui regras que conduzam a estados homogéneos.
- Classe Dois Ciclos - Esta classe é idêntica à classe Ponto Fixo, com a variante de que se aplica a

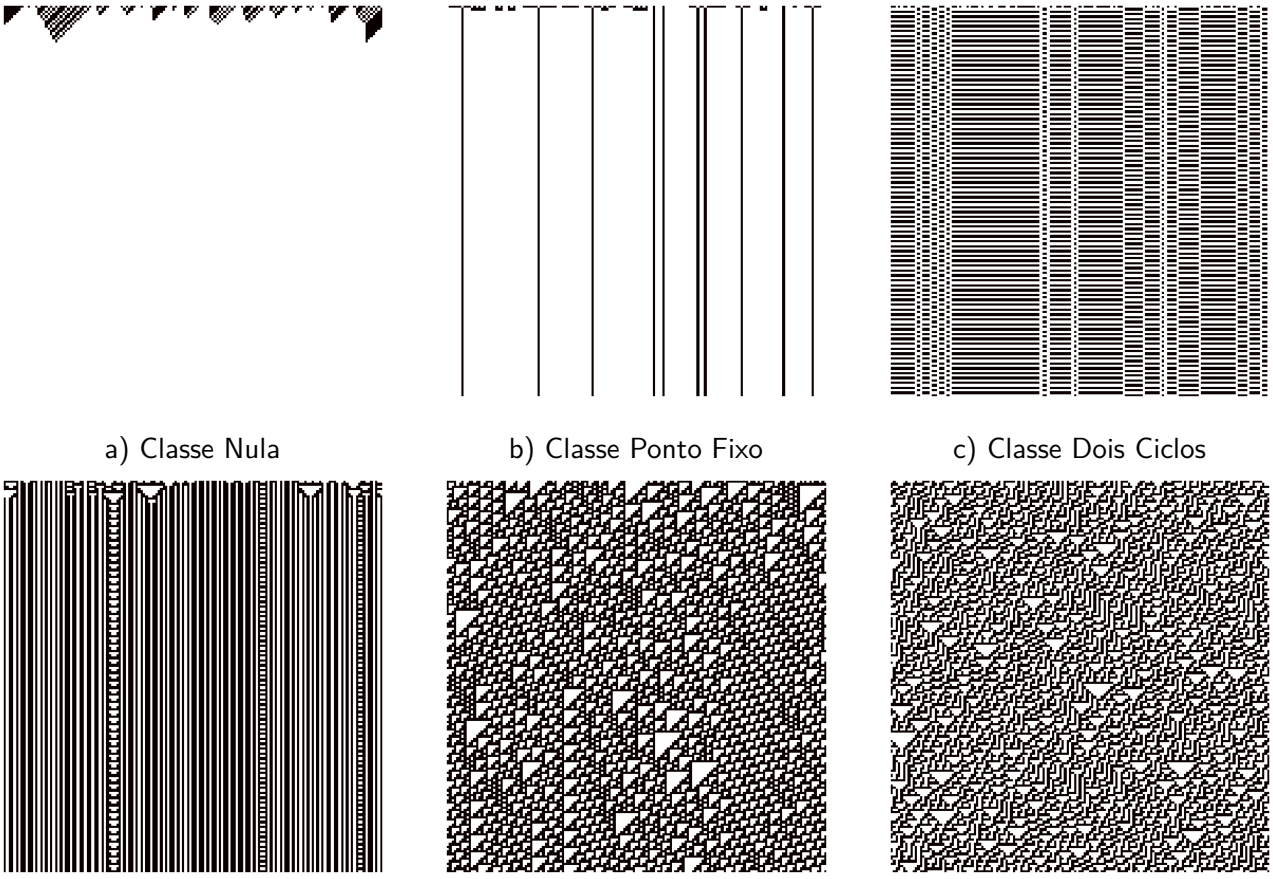
autómatos celulares em que a configuração limite é invariante após **duas** aplicações das regras de transição, incluindo regras que deslocam espacialmente o padrão.

- Classe Periódica - Nesta classe, a configuração limite do autómato celular é invariante após a aplicação de regras de transição T vezes, com o tamanho do ciclo T independente ou fracamente dependente do número de células.

- Classe Complexa - Os autómatos classificados como classe complexa têm uma configuração limite periódica, contudo o tempo necessário para atingir a condição limite pode ser extremamente longa, sendo que normalmente, quanto maior o número de células L na condição inicial, maior é o tempo necessário para atingir a configuração limite.

- Classe Caótica - Esta classe é caracterizada por autómatos que possuem um comportamento dinâmico não periódico, que possuem uma divergência exponencial da duração do ciclo de acordo com o número de células e que possuem uma instabilidade com respeito pelas perturbações das condições iniciais.

Resumidamente, a classificação de Li-Packard vai transformar a classe II da classificação de Wolfram em três classes distintas: Classe Ponto Fixo, Classe Dois Ciclos e Classe Periódica, sendo que na figura 3.1 encontram-se exemplos concretos deste sistema de classificação.



a) Classe Nula b) Classe Ponto Fixo c) Classe Dois Ciclos

d) Classe Periódica e) Classe Complexa f) Classe Caótica

Figura 3.1: Exemplos das seis classes de Li-Packard

3.3 O parâmetro λ de Langton

A classificação em quatro classes proposta por Wolfram apenas considerava os 256 autómatos elementares [VR09], algo limitador caso se pretenda trabalhar com autómatos celulares mais complexos, com um maior número de estados por célula ($n > 2$) e com vizinhanças maiores ($m > 3$) . Tendo isto em consideração, em 1986, Christopher Langton sugeriu classificar os autómatos celulares segundo um parâmetro geral (λ).

A proposta de Wolfram consiste na classificação através de um método visual que em determinadas situações pode conduzir a análises extremamente difíceis e com baixa precisão, pelo que o método proposto por Langton vem resolver este problema, trazendo simplicidade e rigor, sendo a classificação efetuada através de um método quantitativo.

O parâmetro geral (λ) é resumidamente a probabilidade, entre todas as configurações de vizinhança possíveis (n^m), que uma determinada configuração de vizinhança conduza a uma célula ativa. Este parâmetro, vai apresentar um valor entre 0 e 1, sendo que o valor 0 vai representar os autómatos celulares com comportamento fixo e o valor 1 vai representar um comportamento caótico ($\lambda \in [0, 1]$) [VR09].

É preciso ter ainda em consideração que para o cálculo de λ um dos n estados possíveis para as células é definido como um estado *quiescent*, ou seja, as células que apresentarem o estado que é definido como *quiescent* vão ser definidas como estando "mortas".

Nos autómatos celulares o estado 0 vai sempre ser definido como um estado *quiescent*, independentemente do número de estados possíveis n. Por exemplo, para os autómatos celulares binários, as células com o estado 0 são definidas como tendo um estado *quiescent* e as células com o estado 1 são definidas como não *quiescent* ou como células ativas [Kun03]. Se aumentarmos número de estados possíveis para $n = 4$, é igualmente definida a célula com o estado 0 como célula "morta" e as células com o estado 1, 2 e 3 como células ativas.

A fórmula normalizada do parâmetro λ, como já referido, é definido pelo rácio entre transições de células ativas e o número total de transições possíveis, levando a que $\lambda \in [0, 1]$. O parâmetro λ de Langton pode ser então calculado pela seguinte fórmula [VR09]:

$$\lambda = \frac{N - n_q}{N}$$

Na fórmula apresentada para o cálculo de λ, o parâmetro n_q representa o número de estados locais que apresentam o estado *quiescent* e N representa o número de configurações possíveis (n^m).

A partir do cálculo do parâmetro λ, Langton observou que o parâmetro tem uma correlação com o comportamento qualitativo dos autómatos celulares, sendo que à medida que λ aumenta de 0 para 1, os autómatos celulares transitam dum comportamento ponto fixo (classe I de Wolfram) para um comportamento caótico (classe III). Na figura 3.2 é possível observar em detalhe o comportamento dos autómatos celulares de acordo com o valor de λ e a sua correlação com as classes qualitativas de Wolfram, sendo importante frisar que para um valor específico de λ pode ser possível associar mais do que uma classe de Wolfram.

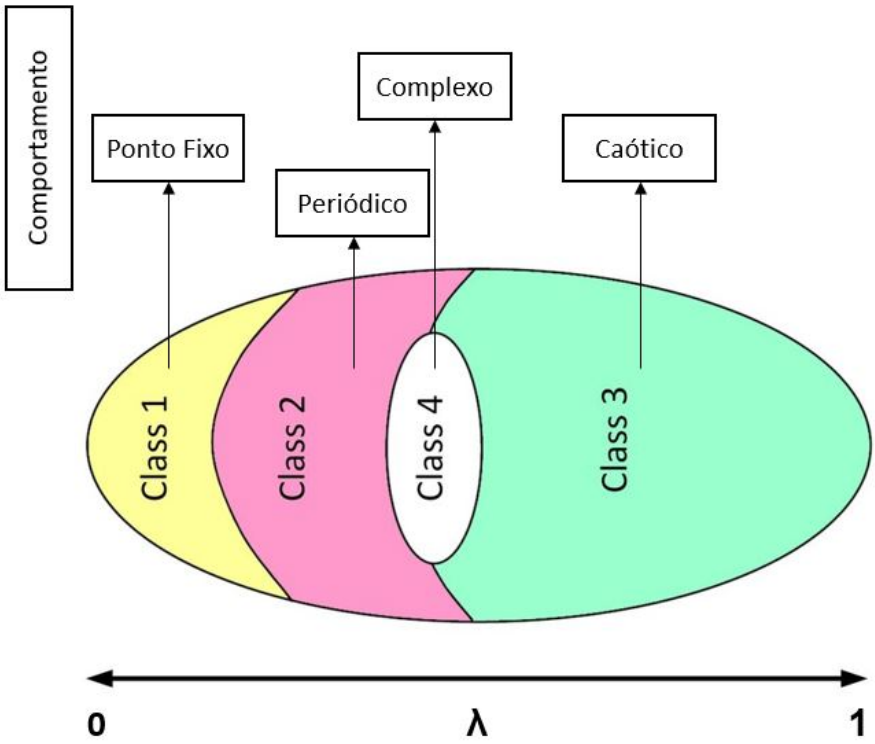


Figura 3.2: Correlação dos valores de λ com as classes de Wolfram

4

Dinâmicas evolutivas dos autómatos celulares

A evolução pode ser definida, de forma geral, como qualquer processo de desenvolvimento que envolva desde células individuais, organismos, até mesmo populações e sociedades. Apesar de a evolução ser uma característica dos seres vivos não está limitada aos organismos vivos. Em biologia, os processos de desenvolvimento são caracterizados em ontogenéticos ou filogenéticos [RR09]:

- Ontogenéticos - Os processos de desenvolvimento ontogenéticos referem-se a processos em que a evolução conduz a mudanças continuas e observáveis sobre um único organismo, ou seja, o desenvolvimento desde um estado simples até uma forma complexa. Um exemplo de um processo ontogenético é o desenvolvimento embrionário.

- Filogenéticos - Pelo contrário, os processos de desenvolvimento filogenéticos são processos de evolução em que, ao invés de um único organismo, a evolução ocorre sobre uma determinada população, que interage com um ambiente, os organismos da população interagem entre si e a evolução é notada tanto nos organismos individuais como nas características globais da população.

4.1 Genótipo e fenótipo

Como referido anteriormente, para certas características dos autómatos celulares podem ser estabelecidos paralelismos com noções biológicas.

Uma destas características é o **genótipo**, que na biologia é o termo utilizado para fazer referência à constituição genética de um indivíduo, ou seja, o genótipo é composto por todos os genes de um determinado organismo. É o conjunto de todos os genes, o genótipo, que vai determinar as características observáveis, ou seja, as informações presentes no genótipo vão definir as diversas características, por exemplo, físicas e comportamentais de um determinado organismo.

Voltando aos autómatos celulares e tendo em consideração a definição de genótipo apresentada, existe uma característica dos autómatos celulares que vai definir as características do autómato celular, que é a regra com a lista das transições locais, como descrito na secção 1.3. A regra das transições locais do autómato celular será identificado com o genótipo do autómato celular.

Por exemplo, se tivermos perante um autómato celular unidimensional que tenha como regra de transição local a regra 30 de Wolfram (ver secção 1.1.2), pela numeração de Wolfram 30 em binário será 00011110 e deste modo reordenando os dígitos do mais significativo para o menos significativo a sequência 01111000 caracteriza as transições locais para o autómato. Esta sequência - 01111000 - vai ser definido como o genótipo do autómato celular.

As características para cada realização do autómato, escolhidas condições iniciais e fronteira, são definidas como o **fenótipo**.

No autómato celular, e tendo em consideração a definição de fenótipo apresentada, um dos elementos observáveis mais diretos e significativos é o diagrama espaço-tempo observável gerado pela evolução temporal do autómato celular, ou seja, o fenótipo dos autómatos celulares é o diagrama espaço-tempo. O genótipo do autómato celular vai determinar as diversas características que o fenótipo (diagrama espaço-tempo) vai apresentar, por exemplo, os diversos padrões e estruturas observáveis que são gerados ao longo da evolução temporal do autómato celular.

Resumindo, o genótipo vai ser definido pela sequência $\alpha = \alpha_0\alpha_1...\alpha_{n^m-1} \in \mathbb{Z}_n^{n^m}$, com $n \in \mathbb{N}$

Para uma determinada regra de transição do autómato celular (genótipo) podemos obter autómatos (organismos) com diferentes características (fenótipo), dependendo das condições ambientais escolhidas. As condições ambientais neste contexto, serão as condições iniciais, tamanho da vizinhança e condições fronteira. Estes organismos apesar de possuírem diferentes fenótipos são definidos como clones, pois possuem o mesmo código genético, ou seja, o mesmo genótipo.

A condição ambiental que vai determinar o fenótipo é a configuração inicial. Se as condições ambientais forem as mesmas os clones vão apresentar fenótipos idênticos, contudo, se forem escolhidas condições ambientais diferentes, o fenótipo pode ser diferente em ambos os clones.

Apesar de os autómatos celulares com o mesmo genótipo poderem gerar autómatos com fenótipos diferentes, estes fenótipos vão ser, por norma, visualmente muito idênticos. Em casos extremos pode ocorrer existirem clones com fenótipos visualmente muito diferentes, sendo que este fenómeno também ocorre na biologia, quando um clone cresce num ambiente muito extremo [RR09].

Os conceitos biológicos que foram apresentados anteriormente, analogamente relacionados com os autómatos celulares, podem ser utilizados no desenvolvimento e implementação de algoritmos genéticos aplicados a autómatos celulares. Os algoritmos genéticos são algoritmos de otimização meta-heurística inspirados pela teoria da evolução que permitem realizar operações de, por exemplo, mutação, recombinação e seleção

numa população [JLH+19].

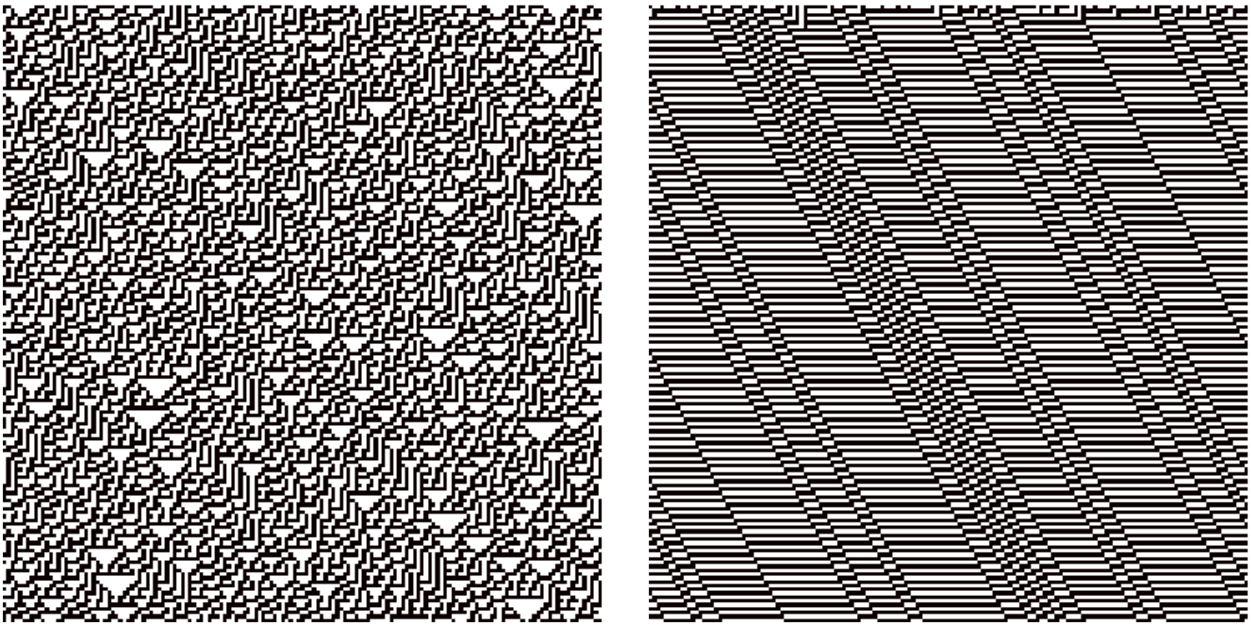
4.2 Mutação

Tal como na biologia, os organismos ao longo da sua evolução podem sofrer mutações, algo que é muito positivo uma vez que as mutações vão permitir obter uma grande diversidade de organismos, que assim podem estar adaptados a uma variedade de condições externas.

Nos autómatos celulares, a operação de **mutação** é a transformação genética mais simples possível de ser efetuada e pode ser definida por uma mudança no genótipo do autómato celular, uma alteração nalgum símbolo da sequência associada ao genótipo, podendo ter uma origem aleatória ou determinística.

Na operação de mutação, o termo "perturbação singular" refere-se à transformação dum único valor do genótipo e é a transformação mais simples de ser efetuada utilizando a operação de mutação sobre o genótipo.

A operação de mutação é especificada pelas posições do genótipo do autómato nas quais surgiu uma mutação. Por exemplo, considerando o autómato celular inicial $x^{(0)}$ com o seu genótipo definido por $x^{(0)} = 01111000$ sobre o qual foram efetuadas mutações e resultou o autómato celular x_1 definido pelo genótipo $x^{(1)} = 11111000$, chegamos a conclusão que existiu uma mutação sobre o autómato $x^{(0)}$ na posição $i = 0$, sendo que na figura 4.1 podemos comparar a realização do autómato celular original com a sua mutação e podemos concluir que, neste caso, uma simples alteração proporcionou uma alteração muito considerável no diagrama espaço-tempo.



a) Autómato inicial - regra 30 b) Autómato com mutação na posição $i = 0$

Figura 4.1: Comparação do autómato original com o autómato com uma mutação na posição $i = 0$

4.3 Replicação

A replicação é uma operação genética, que altera a vizinhança, onde o genótipo do organismo vai ser copiado, sendo que quando aplicado ao contexto dos autómatos celulares, significa que o genótipo x do autómato celular vai ser copiado **n** vezes de forma a obter um genótipo maior $\tilde{x}$, sendo que o fenótipo para ambos os genótipos $x, \tilde{x}$, é idêntico se as condições ambientais forem as mesmas. Considerando o autómato celular com o genótipo definido por $x = x_0...x_{N-1}$, com $N = n^m$, o genótipo resultante da operação de replicação do genótipo $x = x_0...x_{N-1}$ é definido por:

$$\tilde{x} = x_0...x_{N-1}x_0...x_{N-1}x_0...x_{N-1} = xx...x \in \mathbb{Z}_n^{n^m+1}$$

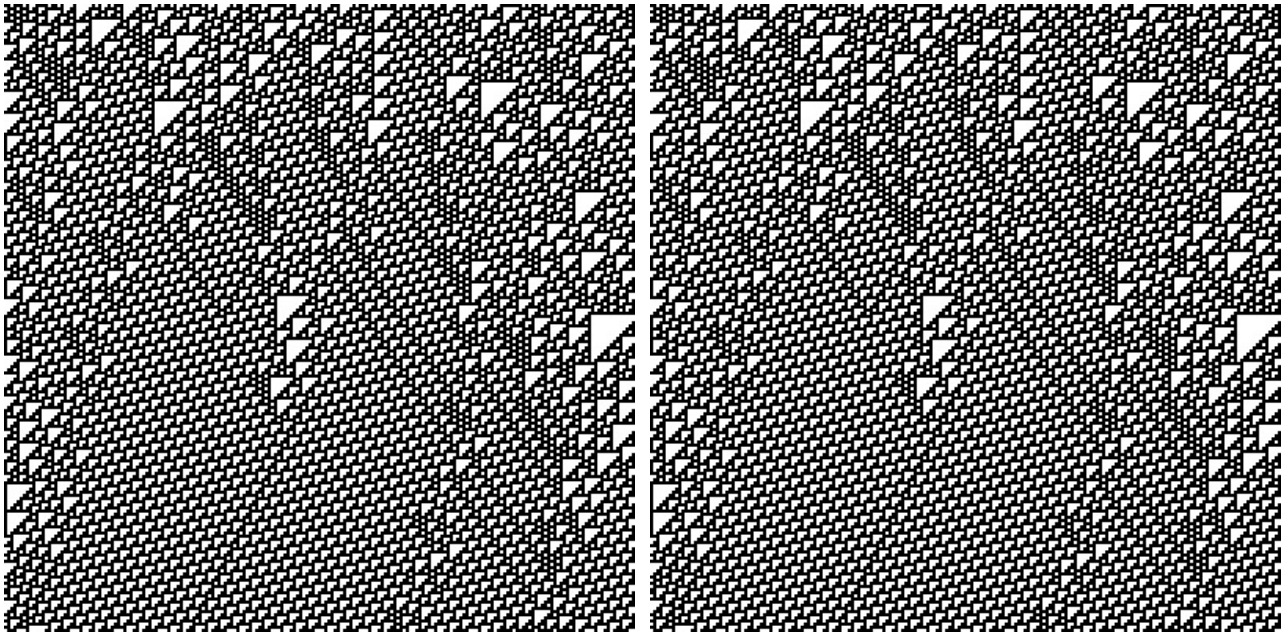
Por outras palavras, o genótipo replicado $\tilde{x}$ é composto por n cópias exatas do genótipo x original, com o tamanho do genótipo replicado $\tilde{x}$ a ser definido por n^{m+1} e com o tamanho da vizinhança a ser $m+1$, ou seja, o genótipo original x vai ser copiado n vezes e o seu tamanho vai ser $N = n^{m+1}$

Resumindo, o genótipo replicado $\tilde{x}$ é composto por n cópias exatas do genótipo original x, sendo que a consequência desta ordenação lexicográfica do genótipo, que vai determinar as regras de transição locais, vai ser que o fenótipo de ambos (com as mesmas condições ambientais) vai ser exatamente o mesmo. Ou seja, se se mantiverem as condições iniciais e as condições fronteira, ambos os autómatos celulares com os genótipos x e $\tilde{x}$ vão originar um diagrama espaço-tempo igual, contudo isto não significa que os genótipos são equivalentes do ponto de vista da evolução dinâmica, sendo a estabilidade a principal diferença entre estes genótipos, algo que vai ser mais detalhado na secção 4.4.1.

É ainda importante realçar que em cada operação de replicação, o valor da vizinhança $m = m + 1$, por exemplo, se o autómato celular x, com $n = 2$ e $m = 3$, o autómato celular replicado $\tilde{x}$ vai apresentar $n = 2$ e $m = 3 + 1 = 4$. A estrutura de vizinhança é alterada do seguinte modo, no autómato original, com genótipo x, tem se a vizinhança (m_-, m_+), com $m = m_- + 1 + m_+$, como explicado na secção 1.1.1. No replicado $x = xx$ a vizinhança passa a $(m_-, m_+ + 1)$, com $m = m + 1$ [RBTF23].

Considerando agora o autómato celular com o genótipo $x = 01110110$ (regra 110 pela numeração de Wolfram) em $\mathcal{G}_{2,3}$ ($n = 2$ e $m = 3$) e efetuando a operação de replicação uma única vez, obtém-se o genótipo $\tilde{x} = 01110110\mathbf{01110110}$ (regra 28270 pela numeração de Wolfram), com $n = 2$ e $m = 4$, sendo que como se pode observar na figura 4.2, apresentam ambas o mesmo fenótipo. O genótipo replicado $\tilde{x}$ é então constituído por $n = 2$ cópias exatas do genótipo original x, com o seu tamanho a ser $N = n^m + 1 = 2^3 + 1 = 16$.

Na figura 4.3 percebe-se o porquê da vizinhança no autómato replicado $\tilde{x}$ aumentar e o fenótipo, para as mesmas condições ambientais, ser o mesmo que no autómato original x. No autómato celular x, com $m = 3$ e tendo em consideração a definição de par balanceado definido em 1.1.1, o par de vizinhança $(m_-, m_+) = (\frac{3-1}{2}, \frac{3-1}{2}) = (1, 1)$ e no autómato celular $\tilde{x}$ com $m = 3$ o par de vizinhança $(m_-, m_+) = (\frac{4}{2}, \frac{4}{2} - 1) = (2, 1)$. Tendo em consideração o valor dos pares de vizinhança, nas configurações possíveis para ambos os autómatos, na figura 4.3, a célula central em análise encontra-se destacada (maior e a negrito) e percebe-se que para o autómato celular replicado $\tilde{x}$, a célula mais à esquerda (destacada a vermelho) em todas as configurações é redundante. Qualquer que seja o estado (0 ou 1) na célula mais à esquerda do autómato replicado, vai resultar na mesma configuração. Se retirarmos esta célula redundante das configurações, o autómato replicado $\tilde{x}$ apresenta exatamente as mesmas regras de transição do autómato original x, ou seja $\tilde{\phi}(*i_0i_1i_2) = \phi(i_0i_1i_2)$,



a) Autómato inicial x - Regra 110 b) Autómato replicado $\tilde{x}$ - Regra 28270

Figura 4.2: Comparação do autómato original com o autómato replicado

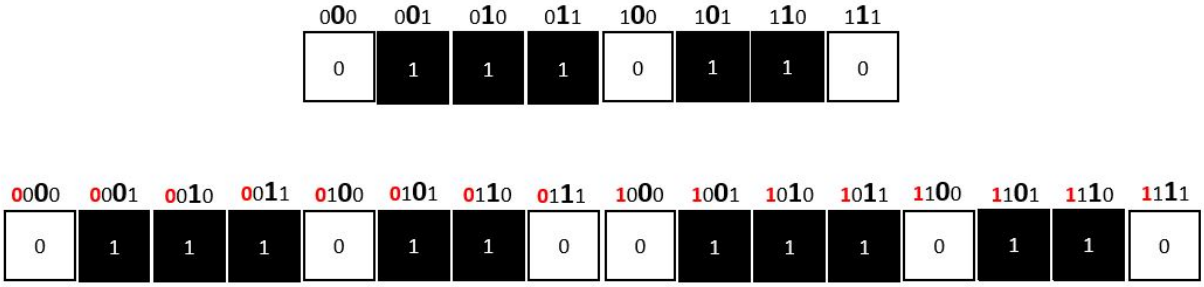


Figura 4.3: Regras de transição para o autómato original x (em cima) e o replicado $\tilde{x}$ (em baixo)

O processo de replicação pode ser repetido indeterminadamente, por exemplo, considerando o genótipo já replicado $\tilde{x} = 0111011001110110$ e aplicando o processo genético de replicação novamente a este genótipo obtemos $n = 2$ cópias exatas do genótipo replicado, com o novo genótipo a ser $\tilde{x^{(1)}} = 0111011001110110\mathbf{0111011001110110}$, com $n = 2$ e $m = 4 + 1 = 5$.

4.4 Estabilidade do fenótipo

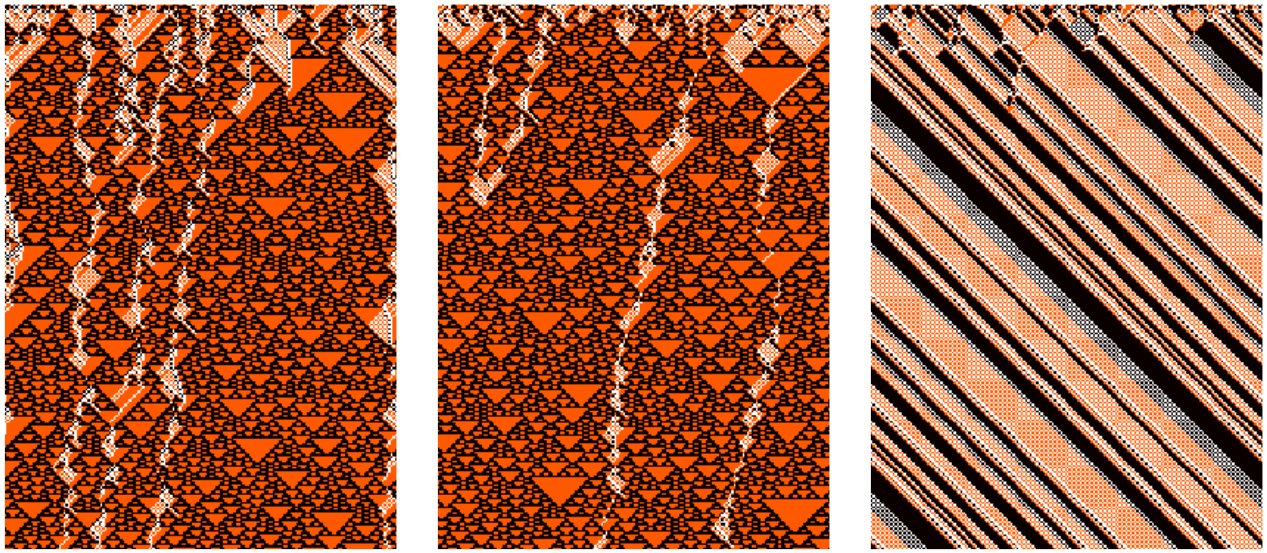
A estabilidade do fenótipo vai ser alvo de um estudo mais aprofundado no capítulo 5, contudo é importante perceber que a aplicação da operação genética de mutação sobre um autómato celular vai resultar numa alteração visual do fenótipo. Contudo existem autómatos celulares mais sensíveis e outros mais robustos quando sujeitos a estas operações de mutação. Por exemplo, considerando o autómato celular inicial $x^{(0)}$ em $\mathcal{G}_{3,3}$, uma condição fronteira periódica e com o genótipo definido pela sequência $x^{(0)} = 201012020211012022102220121$, aplicando-se uma operação de mutação a este autómato inicial

$x^{(0)}$ resultaram os seguintes autómatos celulares, definidos pelos genótipos correspondentes:

$$x^{(1)} = 2010120202110120221\mathbf{1}220121 \text{ (mutação na 21ª posição)}$$

$$x^{(2)} = 201012020211012022102220122 \text{ (mutação na 27ª posição)}$$

Na figura 4.4 encontram-se as representações em diagrama espaço-tempo destes três autómatos definidos anteriormente e conseguimos perceber diferentes comportamentos na estabilidade do fenótipo perante a operação de mutação. O autómato $x^{(1)}$ resulta de uma mutação na vigésima primeira posição do autómato $x^{(0)}$ e é um autómato que apesar de gerar um fenótipo diferente, é muito idêntico ao original, preservando praticamente todas as suas características. Já o autómato $x^{(2)}$ resulta de uma mutação na vigésima sétima posição do autómato $x^{(0)}$ e ao contrário do autómato $x^{(1)}$, o fenótipo obtido é completamente diferente do fenótipo original do autómato $x^{(0)}$.



a) Autómato $x^{(0)}$ - Autómato original

b) Autómato $x^{(1)}$ - Mutação na posição 24

c) Autómato $x^{(2)}$ - Mutação na posição 27

Figura 4.4: Comparação entre o autómato original $x^{(0)}$ e os autómatos $x^{(1)}$ e $x^{(2)}$ duas mutações diferentes

No autómato $x^{(2)}$ o fenótipo é completamente diferente para as mesmas condições ambientais, sendo possível associar uma correlação entre este exemplo e a hipótese genética nos autómatos celulares. O gene (um conjunto de posições do genótipo) pode ser associado a uma característica visual do fenótipo, sendo que quando este gene é alterado implica o desaparecimento dessa característica visual do fenótipo.

4.4.1 Estabilidade do fenótipo e a operação de replicação

Como referido anteriormente na secção 4.3, se as condições iniciais e as condições fronteira se mantiverem, os autómatos celulares com os genótipos x e $\tilde{x}$ (sendo x o autómato celular original e $\tilde{x}$ o autómato celular replicado) vão originar um diagrama espaço-tempo igual, contudo isto não significa que os genótipos destes dois autómatos são equivalentes do ponto de vista da evolução dinâmica. Do ponto de vista da evolução dinâmica um ponto essencial para analisar é a **estabilidade do genótipo**, sendo que o genótipo do autómato replicado (autómato $\tilde{x}$) é um genótipo mais robusto, sendo que no caso de alguma operação

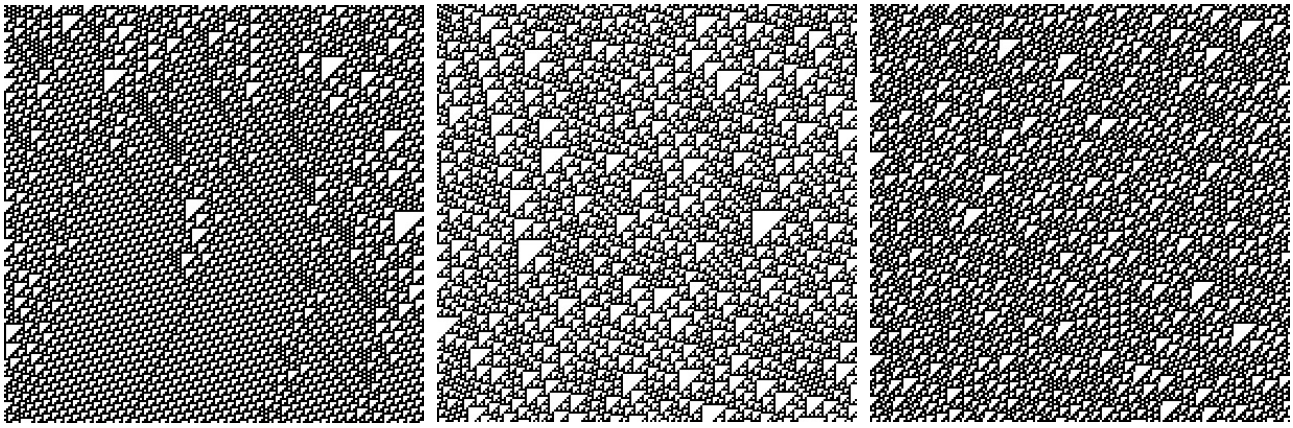
de mutação, o fenótipo deste autómato comparativamente com o fenótipo do autómato original x vai apresentar menos alterações.

Por exemplo, considerando o autómato com o genótipo $y = 01110110$ (regra 110 pela numeração de Wolfram), em $\mathcal{G}_{2,3}$, uma condição fronteira periódica e efetuando a operação de replicação uma única vez, obtém-se o genótipo $\tilde{y} = 01110110\mathbf{01110110}$ (regra 28270 pela numeração de Wolfram), com $n = 2$ e $m = 4$ e aplicando a operação de mutação ao autómato replicado $\tilde{x}$ obtiveram-se os seguintes genótipos:

$$\tilde{y}^{(1)} = 011\mathbf{0}011001110110 \text{ (mutação na quarta posição)}$$

$$\tilde{y}^{(2)} = 01110110010\mathbf{0}10110 \text{ (mutação na décima primeira posição)}$$

Estes dois genótipos correspondem aos autómatos celulares, segundo a numeração de Wolfram, 27246 e 28262, sendo a regra 27246 correspondente o genótipo $\tilde{y}^{(2)}$ e a regra 28262 correspondente ao genótipo $\tilde{y}^{(1)}$. Na figura 4.5 tem-se a comparação entre a realização destes três autómatos e é possível observar que uma simples operação de mutação sobre o genótipo original replicado $\tilde{y}$, apesar de alterar visualmente o fenótipo, não o altera de uma forma muito acentuada, resultado em fenótipos visualmente parecidos.



a) Autómato replicado $\tilde{y}$ b) $\tilde{y}^{(1)}$ - Mutação na posição 11 c) $\tilde{y}^{(2)}$ - Mutação na posição 7

Figura 4.5: Comparação entre o autómato original replicado $\tilde{y}$ e os autómatos $\tilde{y}^{(1)}$ e $\tilde{y}^{(2)}$

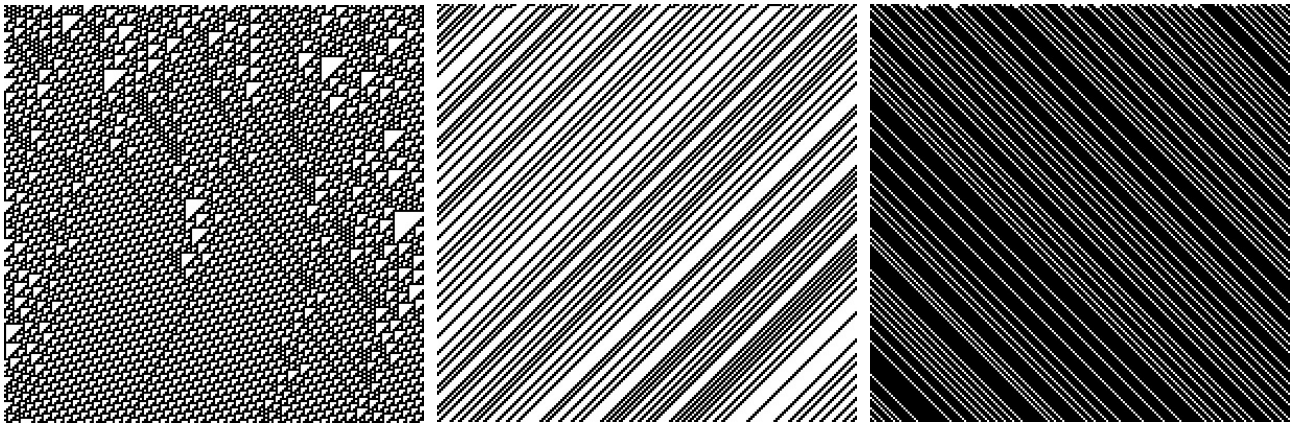
Considerando agora o autómato celular original não replicado com o genótipo $y = 01110110$, correspondente à regra 110 pela numeração de Wolfram e aplicando agora a operação genética de mutação sobre este genótipo obtemos os seguintes genótipos:

$$y^{(1)} = \mathbf{1}1110110 \text{ (mutação na primeira posição)}$$

$$y^{(2)} = 011101\mathbf{0}0 \text{ (mutação na sétima posição)}$$

Na figura 4.6 podemos comparar a realização destes dois genótipos com o genótipo original não replicado $y = 01110110$ e conseguimos perceber que, ambas as mutações destroem completamente o fenótipo original, ao contrário do que aconteceu com o autómato replicado $\tilde{y}$. Apesar de não estarem todos os exemplos, quase todas as mutações possíveis vão destruir o fenótipo original y.

Com isto é possível perceber que para os autómatos replicados x^k, com $n = 2$ e $m = 3 + k$, uma mutação



a) Autómato original y b) $y^{(1)}$ - Mutação na posição 1 c) $y^{(2)}$ - Mutação na posição 1

Figura 4.6: Comparação entre o autómato original y e os autómatos $y^{(1)}$ e $y^{(2)}$

individual não vai alterar significativamente o fenótipo, sendo que quanto maior for o valor de k, menos alterações vão ser notadas no fenótipo do autómato celular [RR09].

4.5 Recombinação

De forma resumida, a operação mais simples de recombinação consiste na conjugação de dois genótipos distintos, de tamanho N, onde é mantido o tamanho original da sequência N e a posição i e a ordem de cada célula no genótipo original. Os dois genótipos são misturados e numa determinada posição i do genótipo resultante da operação, a célula na posição i vai ter o estado de um dos dois genótipos nessa posição i, implicando a necessidade de escolher para cada posição i que elemento vai ser escolhido, se o elemento da posição i do primeiro ou segundo genótipo.

A forma como estes N elementos vão ser escolhidos, no contexto de **algoritmos genéticos**, é normalmente efetuado por processos aleatórios mas também pode ser parametrizado pela iteração de uma aplicação de forma a ganhar controlo de todo o processo de recombinação, podendo esta aplicação produzir, por iteração, uma sequência binária de escolha $a = (a_i)^N{}_{i=1}$ que pode ser vista como uma expansão binária de um número real em $[0, 1]$ [RR09].

Este processo simultâneo de escolha entre as **duas** sequências, para cada posição i do genótipo, vai resultar numa escolha **binária**, ou seja, podemos parametrizar a operação de recombinação através de uma sequência binária. Considerando $a = (a_i)^N{}_{i=1}$ como a sequência binária e os genótipos x, y, com $x = x_1...x_N$ e $y = y_1...y_N$, podemos definir o autómato celular resultante da operação de recombinação $z = x \circ_a y$, com $z = z_1...z_N$ por:

$$z_i = \begin{cases} x_i \text{ se } a_i = 0 \\ y_i \text{ se } a_i = 1 \end{cases}$$

Como exemplo concreto e considerando os autómatos celulares x, y definidos anteriormente e considerando $N = 8$ e $a = 11010000$, se tivermos $z = x \circ_a y$, vamos obter o genótipo $z = y_1y_2x_3y_4x_5x_6x_7x_8$. Como regra geral, $x \circ_a y \neq y \circ_a x$, neste exemplo concreto se considerássemos $y \circ_a x$ iríamos obter o genótipo $z = x_1x_2y_3x_4y_5y_6y_7y_8$.

Esta forma de recombinação através da sequência binária apresenta um problema que é a perda do potencial de diversidade gerado pela recombinação, sendo que, como referido anteriormente, a operação de recombinação no contexto de algoritmos genéticos é normalmente executada através de processos aleatórios, garantindo assim uma maior diversidade nos autómatos gerados.

4.6 Assembly de genótipos

A operação de *assembly* [RR09], consiste na junção de dois genótipos diferentes de forma a obter um novo genótipo, sendo que nos autómatos celulares, é possível realizar esta operação e obter um novo genótipo cujo fenótipo herda características visuais de ambos os genótipos originais. Ao contrário da operação de recombinação, que o novo genótipo mantém o tamanho original N dos genótipos que foram recombinados, nesta operação de *assembly* os genótipos são "juntos", ou seja, irá resultar num genótipo de tamanho maior relativamente aos genótipos originais, tal como na operação de replicação.

Esta operação de *assembly* entre genótipos, que permite a obtenção de um novo genótipo cujo fenótipo resultante preserva características dos dois genótipos originais, vai resultar num autómato celular com um maior número de estados n. Esta operação consiste em partir de um genótipo $x = x_0...x_{N-1}$, com o número de estados locais n e vizinhança m e dum genótipo y definido por $y = y_0...y_{M-1}$ com o número de estados locais n' e vizinhança m', sendo que para ambos os genótipos $N = (n)^m$ e $M = (n')^{m'}$, vamos encontrar uma classe de genótipos em que $b = n + n' + 1$, correspondendo ao autómato celular que tem como sub-autómato celular o autómato gerado por x e y. Os primeiros n símbolos pertencentes a $\mathbb{Z}_b$ são reservados para codificar o genótipo x e os últimos n' são utilizados para codificar o genótipo y, utilizando a correspondência seguinte:

$$\begin{array}{ccccccc} \mathbb{Z}_n & 0 & ... & n & & \mathbb{Z}_{n'} & 0 & ... & n' \\ \downarrow & \downarrow & & \downarrow & \text{e} & \downarrow & \downarrow & & \downarrow \\ \mathbb{Z}_b & 0 & ... & n & & \mathbb{Z}_b & n+1 & ... & n+n'+1 \end{array}$$

Após a codificação com a correspondência anterior, para a configuração $i_0...i_{m-1} \in \mathbb{Z}_n^m$, associa-se a mesma configuração, com os mesmos símbolos em $\mathbb{Z}_b$. Já para a configuração $j_0...j_{m-1} \in \mathbb{Z}_{n'}^{m'}$ associa-se a configuração $(j_0+n+1)...(j_{m'}+n+1)$ em $\mathbb{Z}_b$. Isto fornece um grau enorme de liberdade para determinar o novo genótipo, relacionado com a escolha da imagem da aplicação local associada a configurações que misturam símbolos desde $\{0, ..., n\}$ e $\{n+1, ..., n+n'+1\}$ [RR09] [RCC22].

Como um exemplo, podemos considerar o autómato celular α (regra 18) e β (regra 110), pela numeração de Wolfram, com $n = 2$ e $m = 3$ com os genótipos correspondentes:

$$\alpha = 01001000 \quad \text{and} \quad \beta = 01110110.$$

O segundo genótipo, β, é então transformado por $0 \to \widehat{0} = 2$ e $1 \to \widehat{1} = 3$ em:

$$\widehat{\beta} = 23332332.$$

É necessário ter em consideração que o autómato celular β e $\widehat{\beta}$ são equivalentes embora os símbolos sejam distintos, portanto, os dois autómatos são identificados $\beta \longleftrightarrow \widehat{\beta}$.

Considerando uma regra de transição local γ com $n = 4$ e $m = 3$, obtida pela operação de *assembly* entre os autómatos celulares α e β, a regra γ quando restringida às condições iniciais (e condições de fronteira) com os estados $0, 1$, vai corresponder a um autómato celular que reproduz os mesmos padrões de α e

quando restringida aos estados $3, 4$ vai reproduzir os meus padrões de β (com a transformação $0 \to 3$, $1 \to 4$), sendo que o autómato celular com a regra γ que cumpra esta propriedade é chamada de *assembly* entre α e β.

Como referido anteriormente, existe um grau enorme de liberdade para determinar o novo genótipo, existindo muitas regras diferentes provenientes da operação de *assembly*. As regras das configurações locais em $\mathbb{Z}_4 = \{0, 1, 2, 3\}$, para a regra γ com $n = 4$ e $m = 3$, resultante da operação de *assembly* entre α e β vai ter a seguinte estrutura:

000	001	002	003	010	011	012	013
↓	↓	↓	↓	↓	↓	↓	↓
$\gamma_1 = \alpha_1$	$\gamma_2 = \alpha_2$	γ_3	γ_4	$\gamma_5 = \alpha_3$	$\gamma_6 = \alpha_4$	γ_7	γ_8

020	021	022	023	030	031	032	033
↓	↓	↓	↓	↓	↓	↓	↓
γ_9	γ_{10}	γ_{11}	γ_{12}	γ_{13}	γ_{14}	γ_{15}	γ_{16}

100	101	102	103	110	111	112	113
↓	↓	↓	↓	↓	↓	↓	↓
$\gamma_{17} = \alpha_5$	$\gamma_{18} = \alpha_6$	γ_{19}	γ_{20}	$\gamma_{21} = \alpha_7$	$\gamma_{22} = \alpha_8$	γ_{23}	γ_{24}

120	121	122	123	130	131	132	133
↓	↓	↓	↓	↓	↓	↓	↓
γ_{25}	γ_{26}	γ_{27}	γ_{28}	γ_{29}	γ_{30}	γ_{31}	γ_{32}

200	201	202	203	210	211	212	213
↓	↓	↓	↓	↓	↓	↓	↓
γ_{33}	γ_{34}	γ_{35}	γ_{36}	γ_{371}	γ_{38}	γ_{39}	γ_{40}

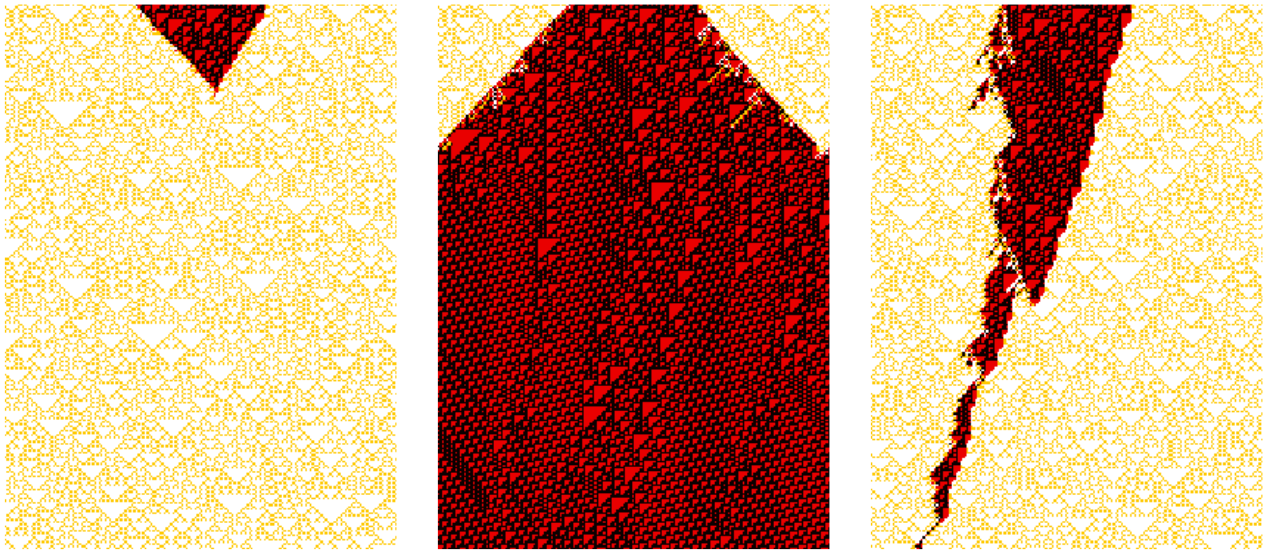
220	221	222	223	230	231	232	233
↓	↓	↓	↓	↓	↓	↓	↓
γ_{41}	γ_{42}	$\gamma_{43} = \widehat{\beta}_1$	$\gamma_{44} = \widehat{\beta}_2$	γ_{45}	γ_{46}	$\gamma_{47} = \widehat{\beta}_3$	$\gamma_{48} = \widehat{\beta}_4$

300	301	302	303	310	311	312	313
↓	↓	↓	↓	↓	↓	↓	↓
γ_{49}	γ_{50}	γ_{51}	γ_{52}	γ_{53}	γ_{54}	γ_{55}	γ_{56}

320	321	322	323	330	331	332	333
↓	↓	↓	↓	↓	↓	↓	↓
γ_{57}	γ_{58}	$\gamma_{59} = \widehat{\beta}_5$	$\gamma_{60} = \widehat{\beta}_6$	γ_{61}	γ_{62}	$\gamma_{63} = \widehat{\beta}_7$	$\gamma_{64} = \widehat{\beta}_8$

Como exemplo concreto, na figura 4.7 encontram-se três realizações distintas γ , provenientes da operação de *assembly* entre os genótipos α e β, As condições iniciais destas realizações são compostas por dois segmentos com condições iniciais aleatórias entre $\{0, 1\}$, que representam a parte inicial e final, sendo o segmento do meio também gerado de forma aleatória mas com os símbolos $\{2, 3\}$, sendo que a principal diferença entre as regras de transição locais $\gamma^{(1)}$, $\gamma^{(2)}$ e $\gamma^{(3)}$ é que para $\gamma^{(1)}$ o valor o valor das regras das configurações locais é escolhido aleatoriamente entre $\{0, 1\}$, o que significa que os padrões de α vão ser dominantes. Para $\gamma^{(2)}$, os valores também são escolhidos aleatoriamente, mas apenas com os valores $\{2, 3\}$,

pelo que os padrões de β vão ser os dominantes. Para $\gamma^{(3)}$, o valor para as regras das configurações locais são escolhidos também aleatoriamente, mas entre todos os valores $\{0, 1, 2, 3\}$, com igual probabilidade, o que significa que os padrões de α e β misturam-se e interagem ao longo das iterações temporais.



a) b) c)

Figura 4.7: Operações de *assembly* distintas da regra 18 com a regra 110: (a) Realização de $\gamma^{(1)}$, (b) Realização de $\gamma^{(2)}$, (c) Realização de $\gamma^{(3)}$.

Os exemplos mostrados na Figura 4.7, correspondentes ao resultado da operação de *assembly* entre $\alpha = 01001000$ e $\beta = 01110110$ correspondem aos genótipos:

$$\gamma^{(1)} = 011200112011010110110020011101200010100002231133100110120223213 2$$

$$\gamma^{(2)} = 012200132212330110320033223231223232120302233133120332123223213 2$$

$$\gamma^{(3)} = 013200313012012310110023011331200020100002231333102310323223213 2$$

5

Estudo da estabilidade do fenótipo em autómatos celulares

A aplicação de operações genéticas e evolutivas sobre autómatos celulares, como referido no capítulo 4, vai resultar numa alteração do fenótipo e apesar de já ter sido feita referência a alguns aspetos no capítulo mencionado, neste capítulo vai ser realizado um estudo mais intensivo de forma a perceber de que forma a aplicação das operações genéticas vai influenciar o fenótipo. Um dos principais fatores a ser estudado vai ser a manutenção dos padrões existentes no diagrama espaço-tempo do autómato original sobre os autómatos resultantes da aplicação das operações genéticas sobre o genótipo original.

Parte fundamental deste estudo é a identificação de autómatos celulares que sejam muito próximos do ponto de vista do genótipo, ou seja, com uma distância de Hamming pequena (ver 5.1.1) mas muito diferentes do ponto de vista do fenótipo, ou seja, dois genótipos muito idênticos vão originar fenótipos muito diferentes. Também o inverso vai ser estudado, ou seja, a identificação de autómatos celulares cujo genótipo é consideravelmente diferente (distância de Hamming consideravelmente grande) mas que resulta na produção de fenótipos muito idênticos do ponto de vista visual ou geométrico.

Ao longo de todo o estudo que vai ser realizado neste capítulo vão sempre ser consideradas condições inicias aleatórias de forma a evitar a situação de uma certa condição inicial produzir um fenótipo atípico e próximo

do produzido por outro autómato que se queira comparar, ou seja, todas as realizações dos autómatos celulares vão ter uma condição inicial aleatória. Para além da condição inicial aleatória, todos os autómatos celulares vão ter uma condição fronteira periódica, o tamanho individual de todos os autómatos celulares vai ser de 180, ou seja, vão existir 180 células em cada iteração temporal t, sendo que cada autómato celular vai ser iterado temporalmente 250 vezes, logo, as iterações vão-se iniciar no instante $t = 0$ e terminar no $t = 249$.

O software desenvolvido para o estudo da estabilidade do fenótipo em autómatos celulares encontra-se no anexo A, com as funções auxiliares a serem definidas no anexo A.1 e o *script* principal do estudo no anexo A.4.

5.1 Medidas para a quantificação dos autómatos celulares

Os diagramas espaço-tempo resultantes da realização dos autómatos celulares são facilmente comparáveis de forma visual, ou seja, visualmente é possível comparar estes diferentes fenótipos e ter uma ideia de que padrões foram preservados aquando da aplicação das operações genéticas e se estes fenótipos são de alguma forma idênticos ou se são muito diferentes. Contudo, de forma a ser possível comparar os autómatos celulares de forma quantitativa foram escolhidas e implementadas algumas medidas para quantificar estas diferenças e consolidar a análise visual do fenótipo.

5.1.1 Distância de Hamming - Quantificação do genótipo

Para a quantificação do genótipo foi escolhida a distância de Hamming (DH), que já foi explicada na secção 3.1.2, e que resumidamente quando aplicada para a comparação de genótipos vai-nos fornecer o número de posições em que os dois genótipos diferem entre si. Com esta medida é possível saber de que forma dois genótipos diferem, sendo que quanto maior for esta distância maior vai ser a diferença entre os dois genótipos.

5.1.2 Erro quadrático médio - Quantificação do fenótipo

Para uma quantificação genérica vai ser utilizado o erro quadrático médio (*Mean Square Error - MSE*) dos valores dos pixels de dois fenótipos. O erro quadrado médio (EQM) vai ter como principal objetivo identificar diferenças entre os dois fenótipos ao nível dos pixels , sendo que quanto menor for o erro, mais idênticas são duas imagens.

A aplicação desta medida é possível porque os fenótipos vão ter todos o mesmo tamanho, ou seja, as imagens resultantes do diagrama espaço-tempo em todos os autómatos celulares vão ter as mesmas dimensões, em pixels, sendo o erro quadrático médio entre duas imagens I_1 e I_2 definido por:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I_1(i,j) - I_2(i,j)]^2$$

Esta medida a nível computacional é bastante rápida e vai ser utilizada para estudar a operação genética de mutação em autómatos celulares que já foram alterados com a operação de replicação.

Inicialmente esta medida foi ponderada para o estudo individual de dois fenótipos específicos, ou seja, para o estudo entre um autómato celular original e um autómato com alguma operação genética aplicada,

contudo, o facto desta medida simplesmente comparar as diferenças ao nível dos pixels nas mesmas posições em ambos os fenótipos não é muito apropriado. Se dois fenótipos forem visualmente muito idênticos, mas se os padrões não estiverem nos mesmos pixels, a medida, por norma, vai dizer que os fenótipos são muito diferentes.

Contudo esta medida apresentou alguns resultados interessantes no estudo da operação genética de mutação em autómatos celulares que já foram alterados com a operação de replicação, resultados que vão ser apresentados neste capítulo.

5.1.3 SIFT - Quantificação do fenótipo

O algoritmo SIFT (*Scale-Invariant Feature Transform*) foi apresentado em 2004 por David Lowe, [Tya20], e é um algoritmo utilizado para a deteção e descrição de características distintivas em imagens. O principal objetivo deste algoritmo é identificar pontos-chave (*keypoints*) que sejam invariantes a mudanças de escala, rotação, iluminação e outros tipos de transformações geométricas, sendo estes pontos-chave na imagem facilmente detetáveis e podem ser usados para corresponder e comparar diferentes imagens.

Este algoritmo é amplamente utilizado no reconhecimento de objetos e é a capacidade de encontrar características invariantes a transformações geométricas que o torna robusto e eficaz em uma variedade de cenários e condições de imagem. Ao contrário da medida anterior, que comparava pixel a pixel, este algoritmo vai identificar os padrões e procurá-los em toda a imagem, fornecendo resultados mais concretos para a comparação de dois fenótipos.

Apesar deste algoritmo não ser diretamente projetado para comparar autómatos celulares, é possível utilizar este algoritmo para, de alguma forma, quantificar as diferenças que existem entre os fenótipos. Os dados de entrada do algoritmo vão ser as imagens resultantes da realização do autómato celular, ou seja, o diagrama espaço-tempo.

A partir dos dados de entrada o algoritmo vai identificar os *keypoints*, que este caso vão ser os padrões existentes em ambos os fenótipos, e para isto o algoritmo analisa a imagem em várias escalas e identifica regiões com características distintivas, como mudanças de contraste e textura. Depois de todos os *keypoints* identificados em ambos os fenótipos, a última etapa deste algoritmo vai ser a correspondência dos *keypoints* identificados entre as duas imagens (fenótipos), fornecendo os *keypoints* que são comuns aos dois fenótipos, ou seja, vai identificar que padrões estão presentes nos dois fenótipos em estudo.

A correspondência dos *keypoints* entre os dois fenótipos é o que vai possibilitar a quantificação dos fenótipos, ou seja, a partir do número de *keypoints* (padrões) que são comuns a ambos os fenótipos é possível ter uma ideia do quão semelhantes são dois fenótipos. Em teoria, se dois fenótipos tiverem um grande número de *keypoints* em comum, vão ser visualmente mais idênticos do que dois fenótipos com um número consideravelmente menor de *keypoints* em comum.

É desta forma que vai ser considerada a escala para a quantificação dos dados recolhidos por este algoritmo, ou seja, a partir da comparação entre um fenótipo original e vários fenótipos resultantes da aplicação de operações genéticas no fenótipo original vai ser possível quantificar quais são os fenótipos mais idênticos e os mais díspares relativamente ao original. Como já foi referido, os fenótipos mais idênticos vão apresentar um maior número de *keypoints* enquanto os com maiores diferenças vão apresentar um número menor de *keypoints*.

5.1.4 Análise direta do fenótipo

A utilização do algoritmo SIFT para a comparação de fenótipos é bastante robusta, contudo, a combinação deste algoritmo com uma análise direta e visual do fenótipo fornece uma avaliação mais completa e robusta da comparação dos fenótipos. Esta abordagem permite uma comparação mais abrangente entre os diferentes fenótipos dos autómatos celulares.

A análise direta dos fenótipos em autómatos celulares envolve a identificação e a interpretação das características e padrões observáveis que surgem da dinâmica do sistema. Esta análise visual que resulta numa classificação descritiva, permite observar os estados do autómato celular ao longo do tempo e identificar padrões, estruturas ou comportamentos específicos e também identificar configurações estáveis, oscilações periódicas, estruturas em movimento ou regiões de interesse.

A comparação visual de dois ou mais fenótipos vai implicar a análise das características e padrões observáveis em cada fenótipo e posteriormente a procura por semelhanças ou diferenças entre os mesmos, ou seja, os padrões, estruturas e comportamentos observados em cada fenótipo vão ser identificados e visualmente vão ser procurados nos outros fenótipos.

5.2 A mutação e a estabilidade do fenótipo

A operação genética de mutação, como referido em 4.2, refere-se à transformação genética mais simples de ser efetuada e refere-se à alteração num ou em múltiplos valores do genótipo, sendo que esta perturbação pode ser gerada aleatoriamente ou gerada por um processo determinístico [RCC22].

Nos autómatos celulares a operação genética de mutação pode ter impactos distintos na estabilidade do fenótipo, por exemplo, considerando a operação genética de mutação na sua forma mais simples, uma perturbação singular, existem autómatos celulares que vão manter as suas características observáveis (padrões, estruturas e comportamentos observados) do fenótipo quando sujeitos a esta operação e outros que vão ficar irreconhecíveis, ou seja, existem autómatos que são mais sensíveis a esta operação do que outros. Também é possível que alguns autómatos celulares quando sujeitos a uma perturbação singular numa determinada posição i sejam muito robustos, mas quando sujeitos a uma perturbação singular na posição j já sejam sensíveis, ou seja, existem autómatos celulares que são robustos para perturbações singulares apenas em determinadas posições, sendo que noutras posições são fortemente sensíveis [RCC22].

Tendo como objetivo o estudo da estabilidade do fenótipo, considera-se como caso de estudo uma família de autómatos com características fenotípicas muito marcadas, com o autómato celular original definido pelo genótipo x, com uma vizinhança $m = 3$, um número de estados $n = 3$ (definido em $\mathcal{G}_{3,3}$) e considerando a condição fronteira periódica $\Gamma = (i_{N-1}, i_0) = (i_{26}, i_0)$, cuja realização encontra-se na figura 5.1. Vão ser efetuadas operações genéticas de mutação em diferentes posições do genótipo e para cada caso um diagrama espaço-tempo vai ser obtido, considerando sempre condições iniciais aleatórias. O autómato em causa é determinado pelo genótipo:

$$x = 20200021101101022222210111 1$$

Para um estudo mais abrangente de diferentes autómatos celulares também vai ser estudado um autómato celular mais complexo que o anterior com genótipo y, que é um autómato celular com um número de estados $n = 5$ e uma vizinhança $m = 3$, ou seja, é definido em $\mathcal{G}_{5,3}$ e tem $\Gamma = (i_{N-1}, i_0) = (i_{124}, i_0)$, cuja realização se encontra na figura 5.2 e é determinado pelo genótipo:

$$y = 10112012010203330010000132114301221022003232323032102022204212144234330430433121211104303324034420334444121100040211034304333$$

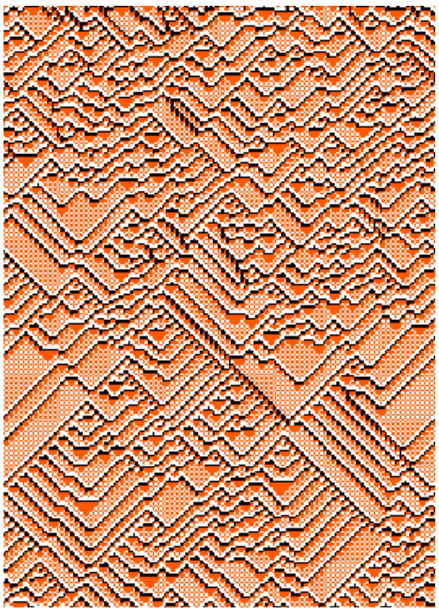


Figura 5.1: Realização do genótipo original x

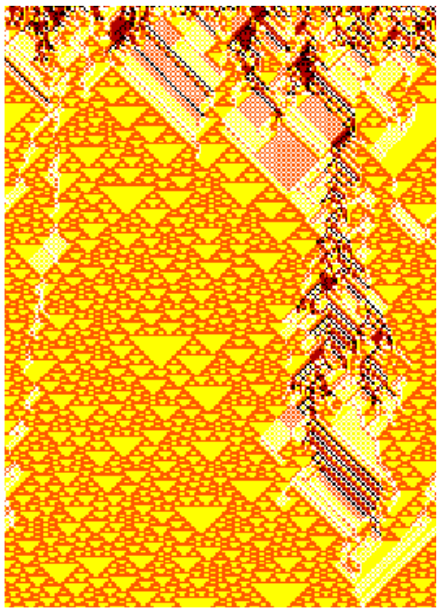


Figura 5.2: Realização do genótipo original y

Para cada realização vai ser efetuada a comparação com o genótipo original através do algoritmo SIFT (número de *keypoints* em comum entre os dois fenótipos) e da análise direta e descritiva do fenótipo, conseguido assim perceber o nível de similaridade entre os fenótipos.

5.2.1 Perturbação singular e a estabilidade do fenótipo

Como referido, existem autómatos celulares que são mais robustos ou mais sensíveis a alterações no fenótipo de acordo com a posição do genótipo em que se realiza a mutação, como tal, nesta secção vão ser estudadas algumas operações singulares de mutação que afetam de diferentes formas a estabilidade do fenótipo.

Considerando inicialmente o genótipo original x, apresentado anteriormente, cuja realização se encontra na figura 5.1, foram realizadas três operações de mutação singular que resultaram em três genótipos, cuja distância de Hamming (DH) tem o valor 1, resultando nos seguintes genótipos:

$$x = 20200021101101022222210111 \text{ - Genótipo Original}$$

$$x^{(1)} = 20200021101101022222211\mathbf{1}1111 \text{ (mutação na posição 23)}$$

$$x^{(2)} = 20200021101101022222210\mathbf{2}11 \text{ (mutação na posição 25)}$$

$$x^{(3)} = \mathbf{1}02000211011010222222101111 \text{ (mutação na posição 1)}$$

Antes de iniciar o estudo dos genótipos apresentados anteriormente, para uma melhor compreensão e

análise do número de *keypoints* obtidos pelo algoritmo SIFT para este autómato celular, se compararmos a realização do fenótipo original com ela própria, para distintas condições iniciais, obtemos 1699 *keypoints* em comum. Como este valor pode ser de alguma forma díspar para comparação com realizações de mutações, foi desenvolvido na linguagem de programação Python um algoritmo que permite gerar todas as mutações singulares possíveis de serem efetuadas sobre um determinado genótipo e que permite comparar cada uma delas com o fenótipo original, utilizando o algoritmo SIFT. É importante ainda realçar que os autómatos celulares obtidos da execução do algoritmo têm uma condição inicial aleatória diferente da do autómato celular original, de forma a evitar que uma certa condição inicial produza um fenótipo atípico e próximo do produzido por outro autómato que se queira comparar.

O algoritmo mencionado retorna todas as mutações possíveis ordenadas por ordem crescente de similaridade (segundo o critério do algoritmo SIFT) e a execução deste algoritmo sobre o autómato celular x indicou que o fenótipo da mutação com maiores diferenças perante o original apresenta 52 *keypoints* em comum, pelo que é possível concluir que a presença de *keypoints* não implica que os fenótipos sejam de alguma forma semelhantes, apenas que este número possibilita a sua comparação. Por outro lado, excluindo ainda qualquer análise direta do fenótipo, o algoritmo indicou que o autómato celular mais idêntico com o original apresenta 653 *keypoints* em comum.

Tendo em consideração os valores obtidos pela aplicação do algoritmo, vai ser possível ter uma maior perspetiva dos valores que o algoritmo SIFT retorna para a comparação dos fenótipos.

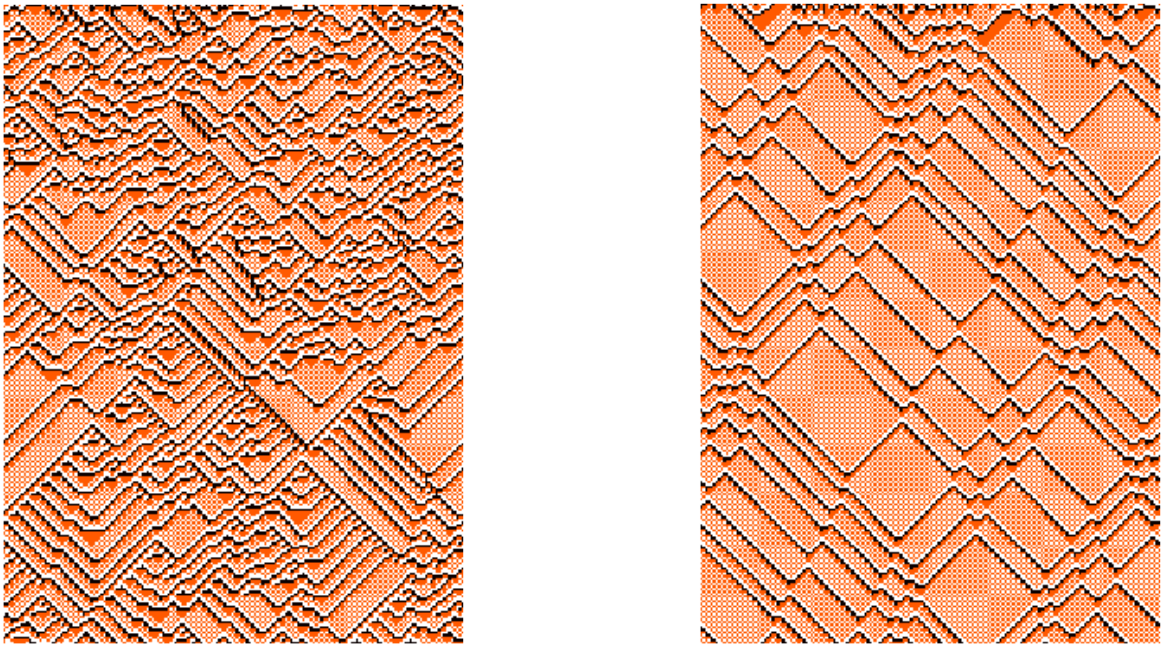
Iniciando agora o estudo no autómato celular $x^{(1)}$, cuja realização se encontra lado a lado com a realização do autómato original na figura 5.3, em termos quantitativos existem 160 *keypoints* em comum entre estes dois fenótipos, segundo o algoritmo SIFT. O número 160 obtido da comparação do fenótipo da realização do autómato celular original x com o autómato celular $x^{(1)}$ pode ser considerado um valor que apesar de considerável, indica que existem muitas diferenças entre estes dois fenótipos, tendo em consideração que a realização menos idêntica apresenta 52 *keypoints* e a mais idêntica 653.

Para uma análise mais abrangente, vai ser efetuada também uma análise direta e visual dos fenótipos para confirmar a informação e análises provenientes do algoritmo SIFT. Partindo para a análise direta é possível perceber que a mutação na posição 23 de $x^{(1)}$, apesar de não destruir totalmente o fenótipo, muitos dos padrões, estruturas e comportamentos existentes no fenótipo original desaparecem, mantendo apenas o padrão de algumas das linhas "principais", pelo que a análise direta vai em linha com a análise do algoritmo SIFT, que sugeriu que este fenótipo fosse consideravelmente diferente do original.

Em relação ao autómato celular $x^{(2)}$, cuja realização se encontra lado a lado com a realização do autómato original na figura 5.4, o algoritmo SIFT indica que este fenótipo é mais idêntico com o original do que o autómato celular $x^{(1)}$, existindo 398 *keypoints* em comum entre estes dois fenótipos, sendo que este valor aproxima-se mais dos fenótipos mais idênticos.

Também a análise direta dos fenótipos confirma a indicação do algoritmo, com este fenótipo a ser visualmente mais idêntico com o original comparativamente com o anterior, mantendo mais padrões e estruturas, mas mesmo assim a ser, visualmente, significativamente diferente quando comparado com o original, contudo, é possível concluir que a indicação do algoritmo SIFT é correta.

Passando para o autómato celular $x^{(3)}$, cuja realização se encontra na figura 5.5, para este diagrama espaço-tempo o algoritmo SIFT indica que existem 303 *keypoints* em comum entre o genótipo original x e o genótipo $x^{(3)}$. Inicialmente este valor pode parecer absurdo, pois está a indicar que este fenótipo é mais idêntico com o original do que a realização do genótipo $x^{(1)}$, mas se olharmos detalhadamente e passando agora para uma análise direta, é possível perceber que apesar de todas as linhas de cor "preta" desaparecerem ao longo da realização do fenótipo mas que os triângulos pequenos laranja são mantidos, pelo que o algoritmo indica que existe um nível de semelhança considerável entre estes dois fenótipos.



a) Autómato inicial x b) Autómato $x^{(1)}$ - Mutação na posição 23

Figura 5.3: Comparação do autómato original com o autómato $x^{(1)}$

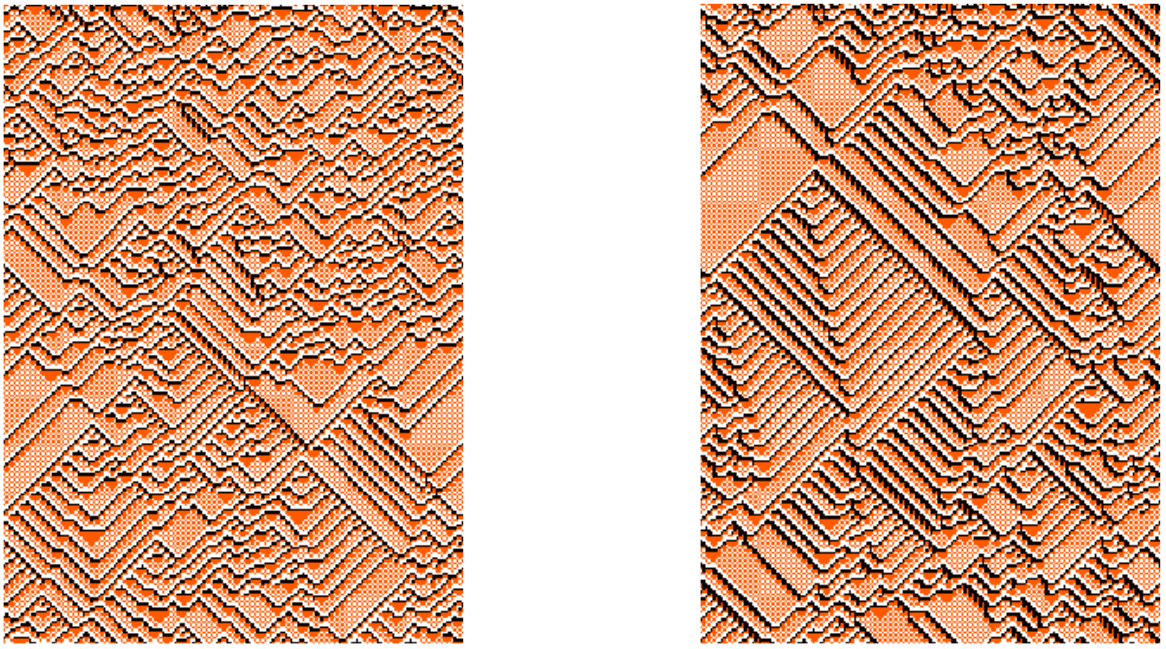
O número de 303 *keypoints* obtido pelo algoritmo SIFT indica-nos que que apesar de algumas diferenças, os fenótipos até são consideravelmente idênticos, mas talvez numa análise direta comparativamente entre o autómato original e os autómatos $x^{(1)}$ e $x^{(3)}$ fosse correto afirmar que o autómato celular $x^{(1)}$, que apenas apresenta 160 *keypoints* em comum é visualmente mais idêntico que a realização do autómato $x^{(3)}$

Esta realização e comparação de fenótipos também é importante para perceber a necessidade de utilização de múltiplas ferramentas para comparar os fenótipos, pois com uma análise superficial direta do fenótipo talvez não fosse possível perceber o nível de semelhança, mas como o algoritmo indicou um valor não esperado, foi efetuada uma análise mais ao detalhe e foi possível perceber a manutenção dos triângulos laranja em ambas as realizações.

Dos três genótipos apresentados, segundo o algoritmo SIFT, a realização destes genótipos enquadra-se no quadrante mediano de semelhanças entre o fenótipo original e todos os fenótipos singularmente perturbados, e apesar de ainda ser necessário ir aos extremos já é possível observar como uma operação singular de mutação pode perturbar de formas completamente diferentes o fenótipo.

Da execução do algoritmo anteriormente mencionado, que utiliza internamente o algoritmo SIFT para comparar todas as operações singulares de mutação possíveis é possível extrair quais as realizações mais idênticas e com maiores diferenças comparativamente com o fenótipo original, pelo que de seguida vão ser apresentados os três genótipos, que mantém a distância de Hamming em 1, mas que vão alterar de forma mais significativa o fenótipo e os três genótipos cuja realização são muito idênticos ao original, segundo as indicações do algoritmo SIFT.

Na figura 5.6 é possível observar as três realizações com uma mutação singular que, segundo o algoritmo SIFT, são as mais idênticas comparativamente com a realização do autómato celular original x, cujos genótipos dos autómatos celulares referidos são os seguintes:



a) Autómato inicial x b) Autómato $x^{(2)}$ - Mutação na posição 25

Figura 5.4: Comparação do autómato original com o autómato $x^{(2)}$

$$x = 20200021101101022222210111 \text{ - Genótipo Original}$$

$$x^{(4)} = 2020002110110102222\mathbf{0}101111 \text{ (mutação na posição 21 | 653 } \textit{keypoints} \text{ em comum)}$$

$$x^{(5)} = 2020002110110102222\mathbf{1}101111 \text{ (mutação na posição 21 | 649 } \textit{keypoints} \text{ em comum)}$$

$$x^{(6)} = 202000211011010222222\mathbf{0}111 \text{ (mutação na posição 24 | 641 } \textit{keypoints} \text{ em comum)}$$

Com as realizações dos autómatos celulares com mutação singular mais idênticos à realização do autómato celular original é possível observar que estas mutações pouco influenciaram a estabilidade do fenótipo, resultando em realizações muito idênticas, visualmente, à realização original. Também é possível observar que ambas as mutações possíveis de serem efetuadas na posição 21 do autómato celular pouco influenciaram na sua estabilidade, pelo que é possível concluir que esta posição do genótipo pouco influencia na estabilidade do fenótipo.

Na figura 5.7 é possível observar as três realizações com uma mutação singular que, segundo o algoritmo SIFT, são as realizações que apresentam maiores diferenças comparativamente com a realização do autómato celular original x, cujos genótipos são os seguintes:

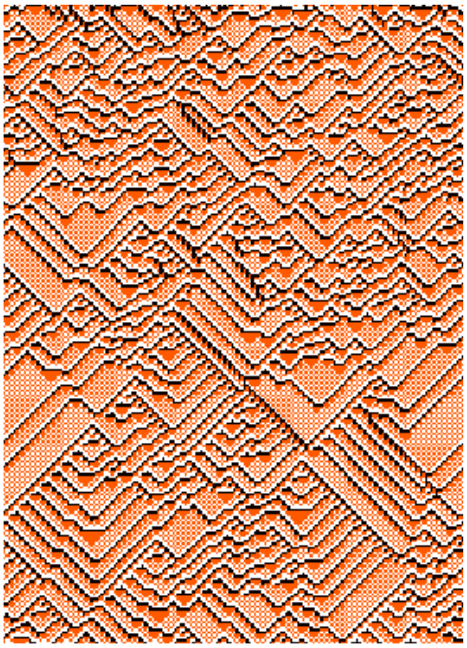
$$x = 20200021101101022222210111 \text{ - Genótipo Original}$$

$$x^{(7)} = 2\mathbf{1}2000211011010222222101111 \text{ (mutação na posição 2 | 52 } \textit{keypoints} \text{ em comum)}$$

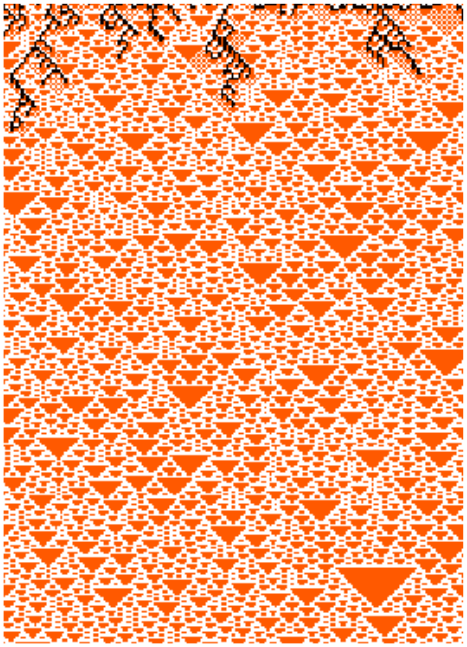
$$x^{(8)} = \mathbf{0}0200021101101022222101111 \text{ (mutação na posição 1 | 57 } \textit{keypoints} \text{ em comum)}$$

$$x^{(9)} = 20200021101100222222101111 \text{ (mutação na posição 14 | 77 } \textit{keypoints} \text{ em comum)}$$

A partir destas três realizações que apresentam maiores diferenças e das três realizações mais semelhantes



a) Autómato inicial x



b) Autómato $x^{(3)}$ - Mutação na posição 1

Figura 5.5: Comparação do autómato original com o autómato $x^{(3)}$

é possível concluir o que foi referido inicialmente, que a operação singular de mutação, apesar de alterar apenas uma posição do genótipo, vai influenciar de formas totalmente diferentes o fenótipo, existindo mutações que pouco vão influenciar a sua estabilidade, mutações que vão influenciar consideravelmente e outras que vão eliminar traços específicos do fenótipo.

Passando agora para o autómato celular y, que foi apresentado no início da secção 5.2 e de forma a confirmar o que foi referido anteriormente, que a posição onde é efetuada a operação genética de mutação vai influenciar de formas completamente distintas o fenótipo, vai ser efetuado o mesmo estudo para este autómato celular, que é um autómato com um genótipo maior, mais complexo e com um maior número de estados comparativamente com o autómato celular x. De forma a analisar os fenótipos mais idênticos e mais díspares com o original, foi novamente utilizado o algoritmo que gera e compara todas as mutações individuais possíveis de serem efetuadas sobre um determinado genótipo e que retorna todas as mutações possíveis ordenadas por ordem crescente de similaridade (segundo o critério do algoritmo SIFT).

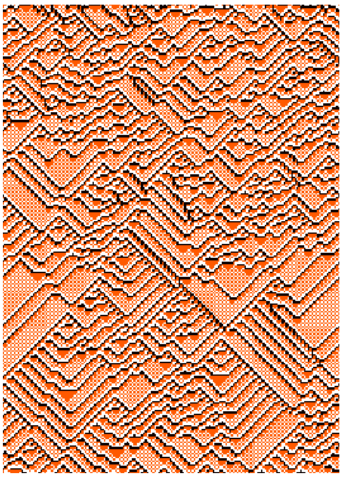
Iniciando o estudo do autómato celular y pelos genótipos que originam os fenótipos mais idênticos com o fenótipo original y, obtiveram-se os seguintes genótipos:

$$y = 10112012010203330010000132114301221022003232323032102022204212144234330430433121211104303324034420334444121100040211034304333$$

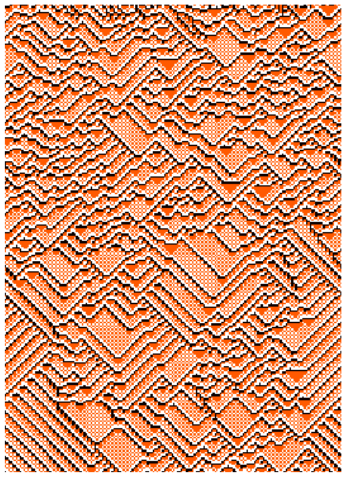
Genótipo Original

$$y^{(1)} = 10112012010203330010000132114301221022003232323032102022204212144234330430433121211104303324034420334444121104040211034304333$$

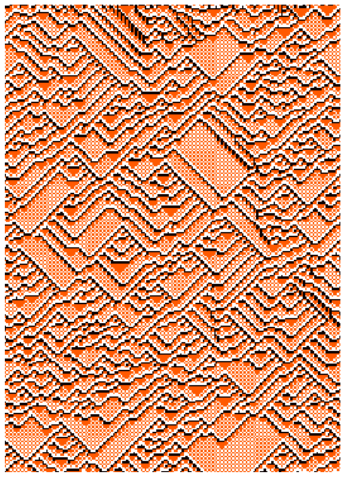
(mutação na posição 110 | 1262 *keypoints* em comum)



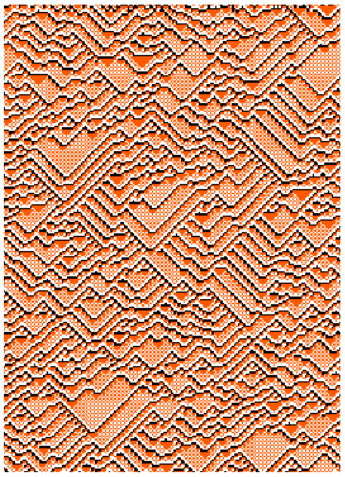
a) Autómato original x



b) Realização de $x^{(4)}$ - Mutação na posição 21



c) Realização de $x^{(5)}$ - Mutação na posição 21



d) Realização de $x^{(6)}$ - Mutação na posição 24

Figura 5.6: Realização dos 3 autómatos celulares com mutação singular mais idênticos ao original

$$y^{(2)} = 10112012010203330010000132114301221022\mathbf{2}3232323032102022204212144234330430433121211104303324034420334444121100040211034304333$$

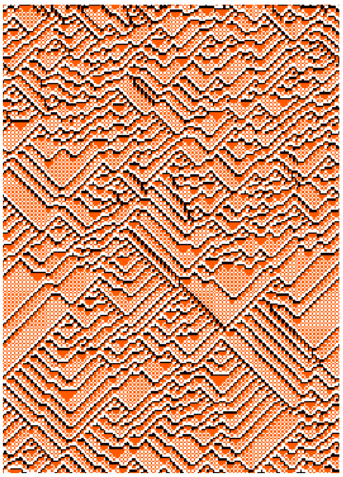
(mutação na posição 40 | 1258 *keypoints* em comum)

$$y^{(3)} = 10112012010203330010000132114301221022003232323032102022204212144234330430433121211104303324034420334444121100040213034304333$$

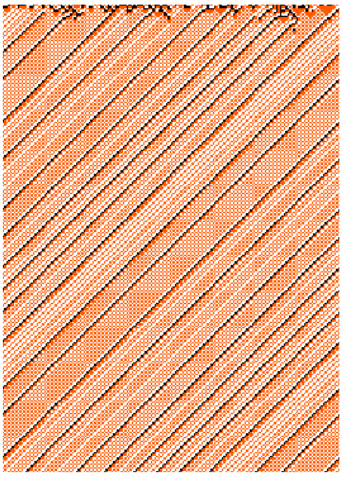
(mutação na posição 116 | 1256 *keypoints* em comum)

As realizações dos genótipos mais idênticos com o original encontram-se na figura 5.8 e como é possível observar as realizações destes genótipos são muito idênticos com o fenótipo original y.

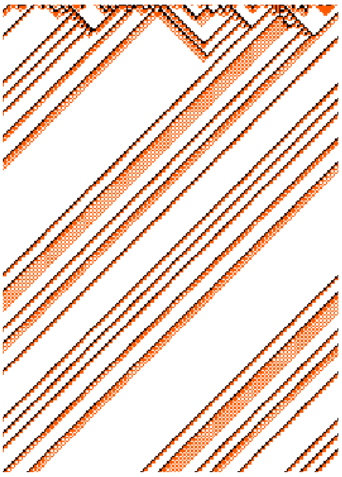
Partindo agora para os genótipos que tornam o fenótipo irreconhecível com apenas uma mutação singular, obtiveram-se os seguintes genótipos:



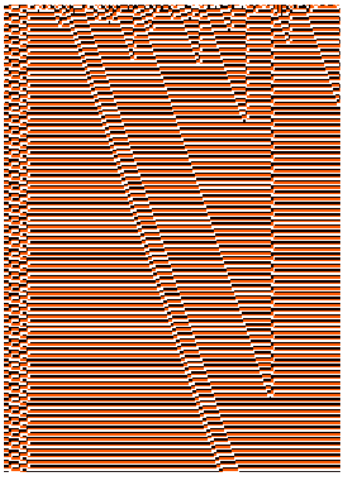
a) Autómato original x



b) Realização de $x^{(7)}$ - Mutação na posição 2



c) Realização de $x^{(8)}$ - Mutação na posição 1



d) Realização de $x^{(9)}$ - Mutação na posição 14

Figura 5.7: Realização dos 3 autómatos celulares com mutação singular menos idênticos ao original

$$y = 10112012010203330010000132114301221022003232323032102022204212144234330430433121211104303324034420334444121100040211034304333$$

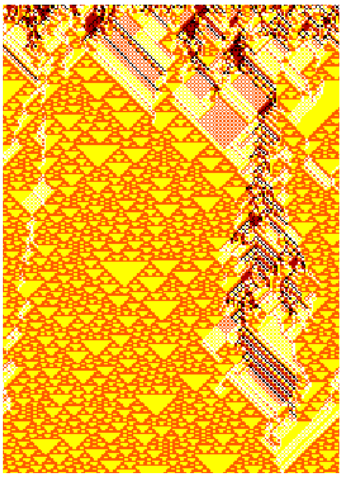
Genótipo Original

$$y^{(4)} = 10112012010203330010000132114301221022003232323032102022\mathbf{1}42121442343304304331212111043033240344203344441211000402110343043333$$

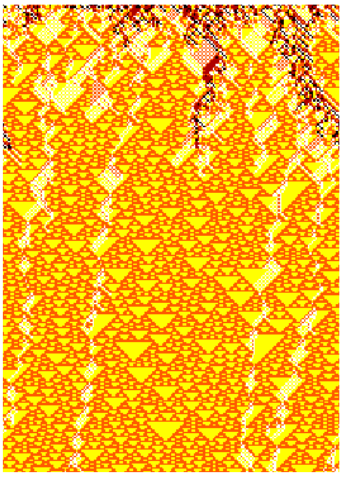
(mutação na posição 58 | 190 *keypoints* em comum)

$$y^{(5)} = 10112012010203330010000132114301221022003232323032102020\mathbf{0}2042121442343304304331212111043033240344203344441211000402110343043333$$

(mutação na posição 56 | 231 *keypoints* em comum)

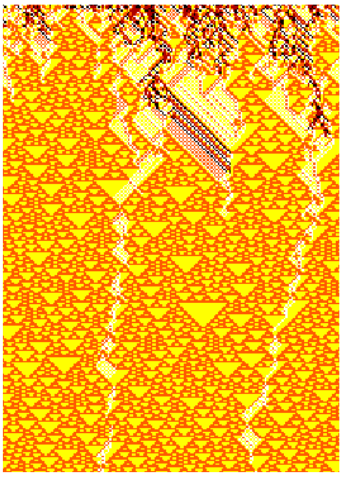


a) Autómato original y



b) Realização de $y^{(1)}$ - Mutação na posição 110

c) Realização de $y^{(2)}$ - Mutação na posição 40



d) Realização de $y^{(3)}$ - Mutação na posição 116

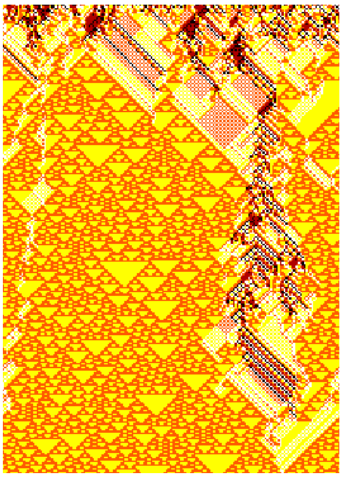
Figura 5.8: Realização dos 3 autómatos celulares y com mutação singular mais idênticos ao original

$$y^{(6)} = 10112012010203330010000132114301221022003232323032102022\mathbf{0}042121$$
$$44234330430433121211104303324034420334444121100040211034304333$$

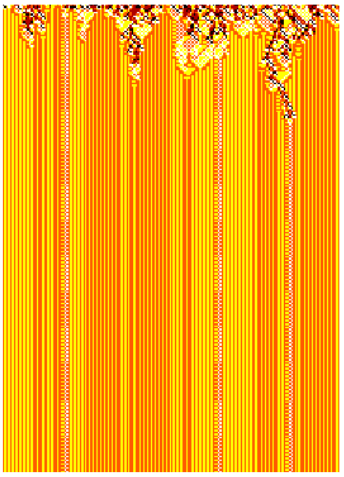
(mutação na posição 57 | 353 *keypoints* em comum)

As realizações dos genótipos com maiores diferenças perante original encontram-se na figura 5.9 e como é possível observar as realizações destes genótipos são muito diferentes da do fenótipo original. Também é relevante observar que as posições do genótipo 56, 57 e 58 (sequência) as que mais tornam o fenótipo irreconhecível, sendo estas posições do genótipo muito importantes para a estabilidade do fenótipo.

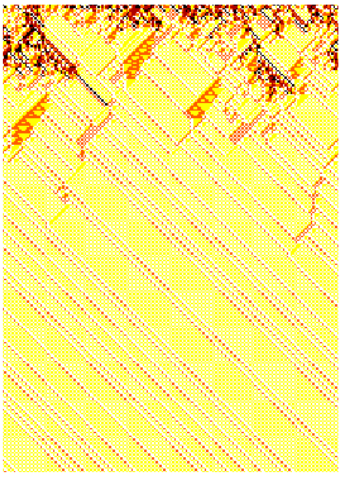
Apesar de ter sido efetuada uma análise mais simples sobre este autómato celular y, é o suficiente para chegar à conclusão anterior, de que com apenas uma operação de mutação num único dígito é possível destruir totalmente o fenótipo, tornando-o completamente irreconhecível independentemente do número de estados e tamanho do genótipo. Por outro lado, também existem mutações singulares que quando aplicadas sobre um determinado genótipo, conservam muitas das características do fenótipo original, como



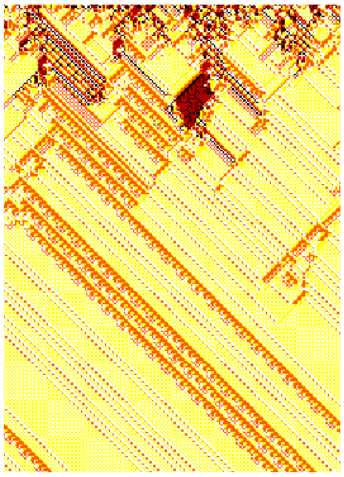
a) Autómato original y



a) Realização de $y^{(4)}$ - Mutação na posição 58



b) Realização de $y^{(5)}$ - Mutação na posição 56



c) Realização de $y^{(6)}$ - Mutação na posição 57

Figura 5.9: Realização dos 3 autómatos celulares com mutação singular menos idênticos ao original

foi visto em ambos os autómatos celulares x e y.

5.2.2 Múltipla perturbação e a estabilidade do fenótipo

Da mesma forma que existem operações de mutação singulares que destroem completamente todos os padrões e comportamentos visualmente identificáveis do fenótipo, existem operações de mutação complexas que ao alterar o genótipo original em três ou mais posições (ou seja, os genótipos alterados vão apresentar uma distância de Hamming de valor igual ou superior a três) o fenótipo vai preservar globalmente os padrões e aspetos identificáveis do fenótipo original.

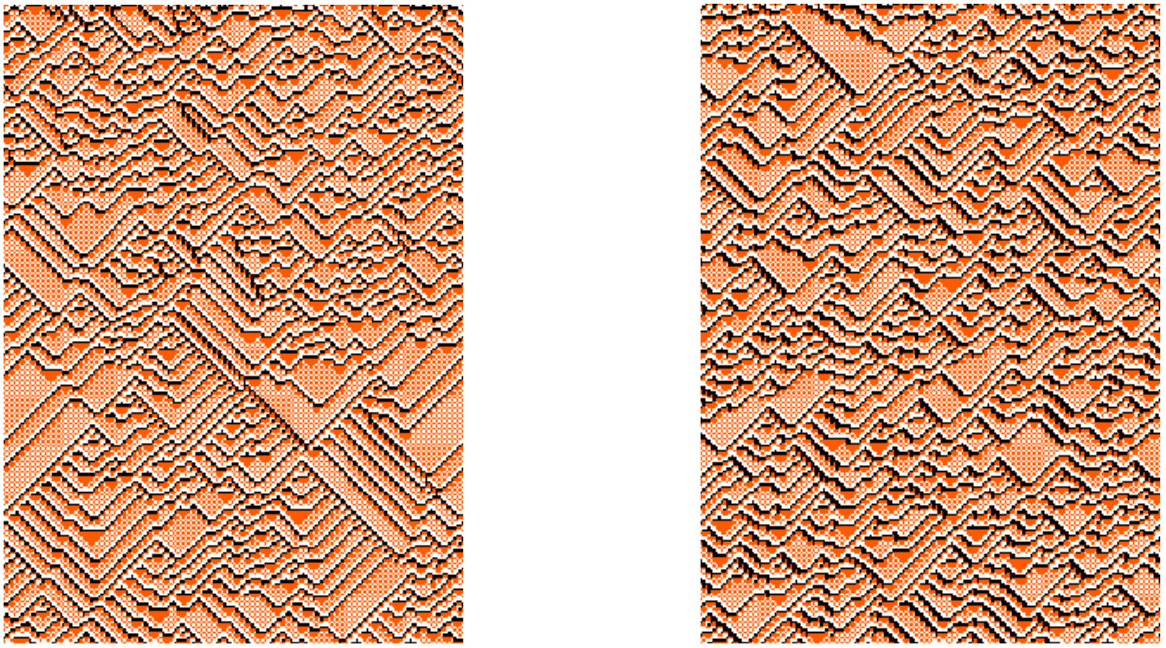
Como exemplo para iniciar o estudo e considerando novamente o genótipo original x, foi efetuada uma operação de mutação que resultou num genótipo cuja distância de Hamming, comparando com o genótipo original, tem o valor 3, que é o genótipo:

$$x = 202000211011010222222101111 \text{ - Genótipo Original}$$

$$x^{(10)} = 20200021101101021\mathbf{1}22\mathbf{0}2101\mathbf{2}11 \text{ (Mutação nas posições 17, 20 e 25)}$$

A realização do autómato celular $x^{(10)}$, encontra-se lado a lado com a realização do autómato original na figura 5.10 e segundo o algoritmo SIFT existem 453 *keypoints* em comum entre estes dois fenótipos, algo que, considerando o estudo das mutações singulares anteriores, indica que existem muitas semelhanças entre estes dois fenótipos. A análise direta dos fenótipos também confirma o que é sugerido pelo algoritmo SIFT, com a maioria dos padrões e estruturas a serem mantidos em ambos os fenótipos, sendo que a maior diferença que consegue ser visualmente identificável é uma maior concentração de "células" de cor preta, ou seja, as linhas pretas aparentam ser mais volumosas, sobressaindo mais na realização do autómato celular $x^{(10)}$

É necessário ter em consideração que o genótipo original tem um comprimento de 27 "genes" e que no autómato celular $x^{(10)}$ foram alterados 3 destes "genes", ou seja, é uma alteração em 11% do genótipo original e mesmo assim obtém-se um nível de semelhança muito superior à maior parte de todas as mutações singulares possíveis.



a) Autómato original x b) $x^{(10)}$- Mutação na posição 17, 20 e 25

Figura 5.10: Comparação do autómato original x com o autómato $x^{(10)}$

De forma a estudar a existência de fenótipos idênticos mesmo quando o genótipo é sujeito a várias operações de mutação foi desenvolvido em Python um algoritmo que de acordo com o número de mutações pretendidas, gera um número pré-definido de mutações possíveis (aleatoriamente) e que vai comparar todas elas com o fenótipo original utilizando o algoritmo SIFT. Esta tarefa é computacionalmente intensiva, pelo que de dentro das milhares de mutações possíveis apenas são selecionadas algumas, neste caso 300, sendo que o algoritmo vai retornar a imagem dos fenótipos mais idênticos segundo o critério do algoritmo.

Da execução deste algoritmo foi possível obter algumas amostras interessantes e muito idênticas ao fenótipo original, sendo que de seguida vão ser apresentados os genótipos dos fenótipos idênticos ao original mesmo quando sujeitos a múltiplas operações de mutação:

$x = 2020002110110102222222101111$ - Genótipo Original

$x^{(11)} = 20200\mathbf{1}2110110102\mathbf{0}22\mathbf{0}10\mathbf{2}111$

(mutação na posição 6, 18, 21 e 24 | 470 *keypoints* em comum)

Distância de Hamming = 4

$x^{(12)} = 20200\mathbf{1}2110110102\mathbf{0}22\mathbf{00}101\mathbf{2}11$

(mutação na posição 6, 17, 20, 21 e 25 | 442 *keypoints* em comum)

Distância de Hamming = 5

$x^{(13)} = 20200\mathbf{1}2110110102\mathbf{00}22\mathbf{00}10\mathbf{2}111$

(mutação na posição 6, 16, 17, 20, 21 e 24 | 429 *keypoints* em comum)

Distância de Hamming = 6

$x^{(14)} = 20\mathbf{0}000\mathbf{0}1\mathbf{2}011010\mathbf{00}22\mathbf{0}2101\mathbf{2}11$

(mutação na posição 3, 7, 9, 16, 17, 20 e 25 | 434 *keypoints* em comum)

Distância de Hamming = 7

Na figura 5.11 é possível observar as realizações dos autómatos celulares referidos anteriormente, onde é possível observar que apesar do elevado número de mutações (4, 5, 6 e 7 mutações respetivamente) o fenótipo mantém-se estável e visualmente mantém-se muito idêntico ao fenótipo do autómato celular original x, pelo menos nas três primeiras mutações. Em termos de análise visual e direta do fenótipo, concluímos que os fenótipos dos autómatos celulares com 4, 5 e 6 mutações, apesar de algumas diferenças, são muito idênticos com o fenótipo do autómato celular original, apesar do elevado número de mutações.

Em relação ao fenótipo do autómato celular $x^{(14)}$, com 7 mutações e uma distância de Hamming de valor 7, já começam a existir algumas diferenças na análise visual e direta, apesar do número de *keypoints* em comum ainda ser relativamente próximo ao das mutações anteriores (cerca de 400). A partir das 7 mutações já não foi possível encontrar fenótipos com um grande nível de semelhança, para o genótipo x, sendo todos os fenótipos visualmente muito diferentes do fenótipo original x e com um número de *keypoints* inferior a 350.

Passando agora para o autómato celular y, foi executado exatamente o mesmo algoritmo de forma a encontrar mutações com uma distância de Hamming superior a 3 que preservem os padrões do fenótipo original, tendo sido encontrados os seguintes genótipos relevantes:

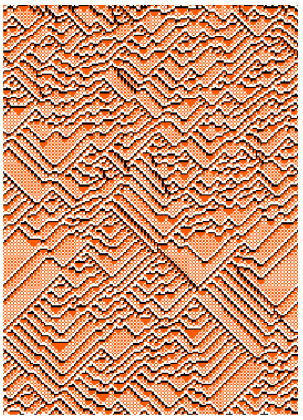
$y = 10112012010203330010000132114301221022003232323032102022204212144234330430433121211104303324034420334444121100040211034304333$

Genótipo Original

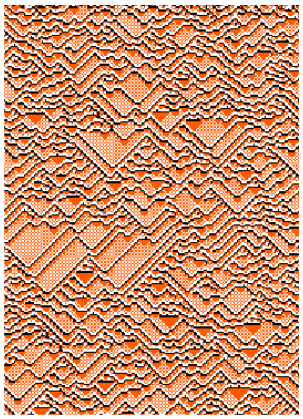
$y^{(7)} = \mathbf{21}1120120102033300100\mathbf{4}1321143012210220032323230321020222042121442343304304331212111043033240344203344\mathbf{2}4412110004021103430433$

(mutação na posição 1, 2, 23 e 102 | 1208 *keypoints* em comum)

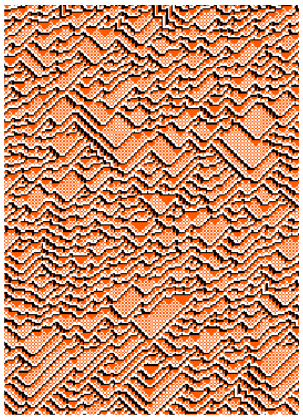
Distância de Hamming = 4



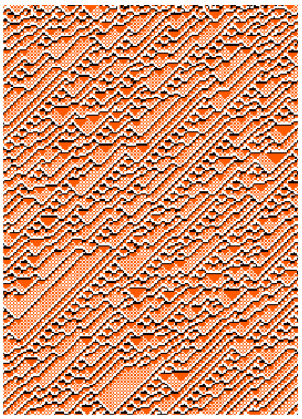
a) Autómato original x



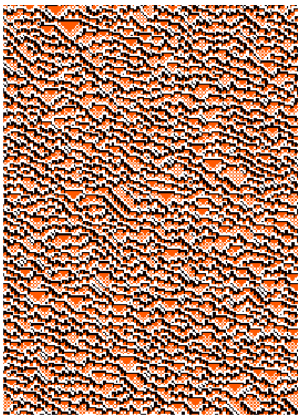
b) Realização de $x^{(11)}$ - Mutação nas posições 6, 18, 21 e 24



c) Realização de $x^{(12)}$ - Mutação nas posições 6, 17, 20, 21 e 25



d) Realização de $x^{(13)}$ - Mutação nas posições 6, 16, 17, 20, 21 e 24



e)Realização de $x^{(14)}$ - Mutação nas posições 3, 7, 9, 16, 17, 20 e 25

Figura 5.11: Realização dos 4 autómatos celulares com mutações múltiplas idênticas ao original

$$y^{(8)} = \mathbf{0104}2012010203330010000132114301221022003232323032102022204212144234330430433121211104303324034420334\mathbf{2}4121100040211034304333$$

(mutação na posição 1, 2, 3, 4 e 103 | 1192 *keypoints* em comum)

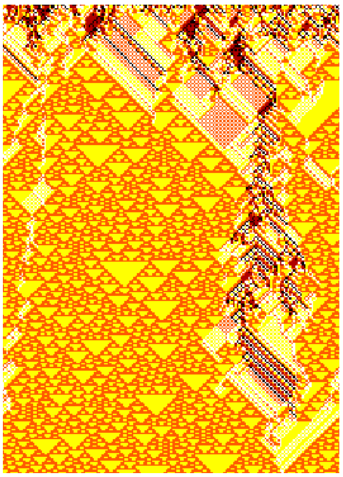
Distância de Hamming = 5

$$y^{(9)} = \mathbf{4100}2\mathbf{1}1201020333001000013211430122102200323232303210202220421214423433043043312121110430332403442033\mathbf{2}444121100040211034304333$$

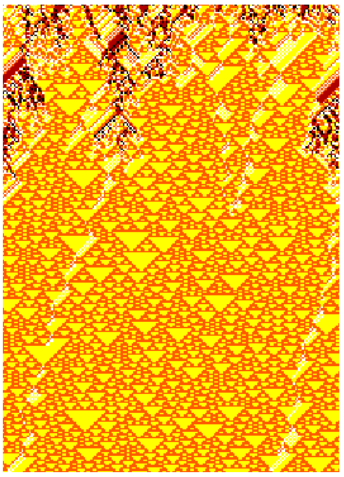
(mutação na posição 1, 2, 3, 4, 6, 101 | 1206 *keypoints* em comum)

Distância de Hamming = 6

Na figura 5.12 é possível encontrar a realização dos três autómatos celulares apresentados anteriormente, com uma distância de Hamming de valor 4, 5 e 6, o que corresponde a 4, 5 e 6 mutações em diferentes dígitos, respetivamente. Com isto é possível concluir o que já tinha sido referido para o autómato celular x, de que apesar de um número elevado de mutações, é possível encontrar genótipos cuja realização preserva e mantém a maioria dos padrões visuais do fenótipo.

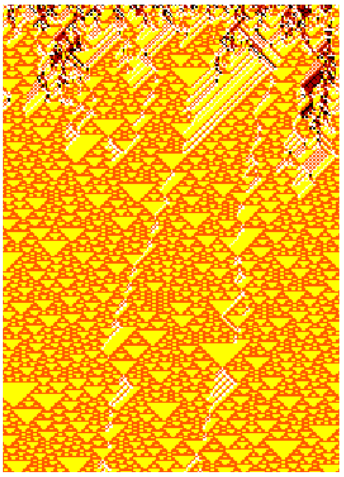


a) Autómato original y



a) Realização de $y^{(7)}$ - Mutação nas posições 1, 2, 23 e 102

b) Realização de $y^{(8)}$ - Mutação nas posições 1, 2, 3, 4 e 103



c) Realização de $y^{(9)}$ - Mutação nas posições 1, 2, 3, 4, 6 e 101

Figura 5.12: Realização dos 3 autómatos celulares com mutação em mais de 3 dígitos idênticos ao original

Com isto é possível concluir que enquanto existem mutações singulares numa única posição que destroem totalmente o fenótipo, existem mutações complexas, cujos genótipos perante o original apresentam uma distância de Hamming igual ou superior a 3, ou seja, com mais de três mutações, que preservam os padrões, estruturas e comportamentos do fenótipo original.

5.3 A replicação e a estabilidade do fenótipo perante mutações

A operação genética de replicação, conforme explicado em 4.3, quando aplicada aos autómatos celulares é definida como uma operação que vai copiar o genótipo de um determinado autómato celular n vezes de forma a obter um genótipo maior, com a particularidade de que os fenótipos de ambos os genótipos (original e replicado) vão ser exatamente iguais. É necessário considerar que para o genótipo replicado a vizinhança m vai ser $m = m' + 1$ (com m' a ser a vizinhança do genótipo original) e que o tamanho do

genótipo vai ser definido por $n^{m'+1}$, ou seja, o genótipo do autómato celular original vai ser copiado n vezes, originado um genótipo de tamanho $N = n^{m'+1}$.

O processo de replicação pode ser repetido indeterminadamente, sendo necessário ter em atenção que a repetição do processo de replicação vai originar n cópias do genótipo do autómato celular replicado. Por exemplo, considerando o autómato celular original a, com $N = 27$, $n = 3$ e $m = 3$, a primeira operação de replicação vai resultar em $n = 3$ cópias exatas do autómato celular a, ou seja, vai originar o autómato celular $\tilde{a}$, com $N = 3^{3+1} = 81$, $n = 3$ e $m = 3 + 1 = 4$. Se for novamente aplicada a operação genética de replicação ao autómato celular $\tilde{a}$, vamos obter $n = 3$ cópias exatas deste genótipo replicado, resultando em $N = 3^{4+1} = 243$, $n = 3$ e $m = 4 + 1 = 5$.

Na secção 4.4.1 e como é mencionado em [RR09], é referido que o genótipo do autómato celular replicado vai ser mais robusto comparativamente com o original, ou seja, quando os genótipos são sujeitos a alguma operação genética de mutação o genótipo replicado vai apresentar uma maior estabilidade do fenótipo, mantendo mais padrões e estruturas existentes na sua realização comparativamente com o genótipo original.

Para estudar e provar que o genótipo replicado vai apresentar uma maior estabilidade do fenótipo, o que é referido em [RR09], vai ser efetuado um estudo sistemático com vários autómatos celulares replicados. Para este estudo e para provar que os fenótipos são mais idênticos, vai ser utilizado a medida do erro quadrático médio e os *keypoints* fornecidos pela análise do algoritmo SIFT, sendo este último o mais apropriado para comparar os padrões existentes nos fenótipos.

Para este estudo vão ser considerados os seguintes genótipos originais:

$$\mathbf{a} = 01110110\,(\mathcal{G}_{2,3} \in \Gamma = (i_{N-1}, i_0) = (i_7, i_0))$$

$$\mathbf{b} = 20200021101101022222210111 1\,(\mathcal{G}_{3,3} \in \Gamma = (i_{N-1}, i_0) = (i_{26}, i_0))$$

$$\mathbf{c} = 01111000\,(\mathcal{G}_{2,3} \in \Gamma = (i_{N-1}, i_0) = (i_7, i_0))$$

Para cada genótipo anteriormente apresentado foi aplicada a operação genética de replicação uma vez, com a exceção do genótipo a, que para uma maior diversidade de genótipos em estudo, foi replicado uma vez e posteriormente novamente replicado, resultando nos genótipos seguintes:

$$\tilde{\mathbf{a}} = 01110110\,\mathbf{01110110\,0111011001110110}$$

$$\mathcal{G}_{2,5} \in \Gamma = ((i_{N-2}, i_{N-1}), (i_0, i_1)) = ((i_{30}, i_{31}), (i_0, i_1))$$

$$\tilde{\mathbf{b}} = 20200021101101022222210111 1\,\mathbf{20200021101101022222210111 1}$$
$$\mathbf{20200021101101022222210111 1}$$

$$\mathcal{G}_{3,4} \in \Gamma = ((i_{N-2}, i_{N-1}), i_0) = ((i_{79}, i_{80}), i_0)$$

$$\tilde{\mathbf{c}} = 01111000\,\mathbf{01111000}$$

$$\mathcal{G}_{2,4} \in \Gamma = ((i_{N-2}, i_{N-1}), i_0) == ((i_{14}, i_{15}), i_0)$$

Para cada um destes genótipos o estudo vai consistir no seguinte, vão ser efetuadas todas as mutações singulares e duplas (2 mutações) possíveis de serem efetuadas em cada um dos genótipos e cada uma das mutações vai ser comparada com o fenótipo original utilizando o algoritmo SIFT, retornando o número de *keypoints* em comum e pela medida do erro quadrático médio, sendo que depois de todas as mutações possíveis serem comparadas, vai ser retornado a média de ambas as medidas referidas, sendo possível concluir se existe ou não uma maior estabilidade do fenótipo nos genótipos replicados.

A execução do algoritmo que possibilitou este estudo tem cinco etapas distintas apresentadas de seguida e o pseudocódigo correspondente encontra-se no Algoritmo 1:

1. O algoritmo recebe como dados de entrada o genótipo, número de estados n , tamanho do autómato, número de iterações, tamanho da vizinhança m e número de mutações.
2. É gerado o fenótipo do autómato celular original (caso não exista) e guardado em formato .PNG
3. De acordo com os dados de entrada (número de estados e número de mutações), são geradas todas as combinações de mutação possíveis de serem efetuadas e são guardadas numa lista.
4. A lista com todas as combinações de mutação é percorrida e para cada uma das mutações é construído o fenótipo correspondente e é guardado o fenótipo numa imagem temporária em formato .PNG, sendo esta imagem temporária comparada com a imagem do fenótipo original utilizando o algoritmo SIFT e com o algoritmo dos erros quadráticos médios, sendo guardado o valor retornado por cada um dos algoritmos em listas distintas.
5. Depois de todas as mutações construídas e comparadas, é retornado o número de mutações que foram comparadas, a média de *keypoints* do algoritmo SIFT e a média dos erros quadráticos médios.

Algoritmo 1: Estudar_Mutacoes

Entrada: genotipo, n, tamanho_automato, num_iteracoes, m, num_mutacoes
Saída: num_mutacoes_comparadas, media_keypoints, media_erros_quadraticos

```
1  início
2      fenotipo_original ← GerarFenotipo(genotipo, k, tamanho_automato, num_iteracoes, m);;
3      GuardarImagemPNG(fenotipo_original, "fenotipo_original.png");;
4      comb_mutacoes ← GerarCombinacoesMutacao(num_estados, num_mutacoes);;
5      keypoints_lista ← lista vazia;;
6      erros_quadraticos_lista ← lista vazia;;
7      para cada mutacao em comb_mutacoes faça
8          fenotipo_mutacao ← GerarFenotipo(mutacao, k, tamanho_automato, num_iteracoes, m);
9          GuardarImagemPNG(fenotipo_mutacao, "fenotipo_temporario.png");
10         keypoints ← AlgoritmoSIFT("fenotipo_original.png", "fenotipo_temporario.png");
11         erros_quadraticos ← ErrosQuadraticosMedios("fenotipo_original.png",
             "fenotipo_temporario.png");
12         keypoints_lista.adicionar(keypoints);
13         erros_quadraticos_lista.adicionar(erros_quadraticos);
14     fim
15     num_mutacoes_comparadas ← tamanho(comb_mutacoes);
16     media_keypoints ← Media(keypoints_lista);
17     media_erros_quadraticos ← Media(erros_quadraticos_lista);
18     retorna num_mutacoes_comparadas, media_keypoints, media_erros_quadraticos;
19 fim
```

A primeira parte do estudo vai incidir sobre a comparação de todas as mutações individuais possíveis de serem efetuadas, cujos resultados se encontram na tabela 5.1.

A partir dos resultados da tabela 5.1 é possível observar que todos os genótipos replicados apresentaram um melhor desempenho comparativamente com os genótipos originais, apesar do número de mutações possíveis ser maior nos genótipos replicados, devido aos mesmos serem maiores.

	Número de Mutações Total	Média de Keypoints	Média Erros Quadráticos
Genótipo a	8	280.37	3.30
Genótipo $\tilde{a}$	32	664.87	3.20
Genótipo b	54	379.26	2.04
Genótipo $\tilde{b}$	162	514.14	1.75
Genótipo c	8	197.13	3.65
Genótipo $\tilde{c}$	16	341.56	3.45

Tabela 5.1: Tabela do estudo de todas as mutações individuais em autómatos celulares replicados

Antes de iniciar a análise dos valores retornados pelos erros quadráticos médios é necessário ter em consideração que quanto menor for este valor, mais idênticas são duas imagens, segundo o critério do algoritmo. Apesar dos valores fornecidos pela média dos erros quadráticos médios não serem as mais indicadas para comparar fenótipos, como referido em 5.1.2, conseguimos observar que existe uma redução superficial, sendo a média dos erros quadráticos de todas as mutações singulares possíveis de serem efetuadas sobre o genótipo menor nos autómatos celulares replicados, o que vai de encontro com o referido em [RR09], de que o genótipo replicado é mais robusto e menos suscetível a mudanças no fenótipo.

Passando agora para a análise dos *keypoints* fornecidos pelo algoritmo SIFT, é possível observar que existe uma diferença muito considerável na maioria dos genótipos, existindo mais *keypoints* em comum entre o fenótipo original e os fenótipos de todas as mutações singulares possíveis de serem efetuadas nos genótipos replicados, o que vai de encontro o que já foi referido, que o autómato celular replicado é mais robusto e menos suscetível a alterações no seu fenótipo.

Para ter mais uma medida de comparação, foi efetuado o mesmo estudo, mas desta vez para todas as combinações de mutação de dois dígitos possíveis de serem efetuadas, para todos os genótipos anteriormente apresentados, cujos resultados se encontram na figura 5.2.

	Número de Mutações Total	Média de Keypoints	Média Erros Quadráticos
Genótipo a	28	135.47	3.37
Genótipo $\tilde{a}$	496	551.34	3.33
Genótipo b	1404	261.22	2.30
Genótipo $\tilde{b}$	1800[1]	396.56	1.88
Genótipo c	28	210.94	3.52
Genótipo $\tilde{c}$	120	309.52	3.50

[1] Existem 12960 mutações possíveis para este genótipo, contudo do ponto de vista computacional envolve um tempo de processamento elevado, pelo que foram selecionadas aleatoriamente 1800 mutações para o estudo.

Tabela 5.2: Tabela do estudo de todas as mutações duplas em autómatos celulares replicados

Os resultados obtidos no estudo de todas as mutações duplas possíveis de serem efetuadas sobre os genótipos, que se encontram na tabela 5.2, vão ao encontro daquilo que foi obtido no estudo de todas as mutações singulares, com o valor médio (média do erro quadrático médio de todas as mutações possíveis) a ser menor nos genótipos replicados, quando comparado com o genótipo original. Também o número de *keypoints* médio vai ao encontro daquilo que foi obtido no estudo anterior, com o número de *keypoints* a ser substancialmente maior para os autómatos celulares que sofreram uma operação genética de replicação.

No autómato celular $\tilde{b}$, apesar de existirem 12960 mutações de dois dígitos possíveis de serem efetuadas e como referido, apenas foram consideradas 1800 mutações, selecionadas aleatoriamente, uma vez que do ponto de vista computacional a execução do algoritmo SIFT para comparar 12960 fenótipos apresenta um

desafio devido ao tempo de processamento envolvido. Apesar do número de comparações ter sido reduzido, é um número suficientemente elevado de mutações comparadas e foi possível retirar as mesmas conclusões que se retiraram para os outros autómatos celulares.

Com isto é possível concluir que existe um refinamento dos autómatos celulares quando sujeitos à operação genética de replicação, conforme referido em [RR09], resultando em genótipos mais robustos do ponto de vista do fenótipo e menos sensíveis a operações de mutação.

6

Estudo da evolução de populações de autómatos celulares

Uma população de autómatos celulares pode ser definida como um conjunto de autómatos celulares que coexistem e evoluem ao longo do tempo, sendo que cada autómato celular é um elemento individual da população, definido pelo seu genótipo, que evolui individualmente segundo as suas regras de transição locais. Por outro lado, a evolução global da população remete a que os autómatos celulares (genótipos) interajam entre si e que troquem informações genéticas para criar novos autómatos celulares, que vão apresentar características partilhadas entre autómatos, sendo a população representada por um conjunto de genótipos dos autómatos celulares.

A evolução global da população ocorre por meio da operação genética de recombinação (que foi apresentada na secção 4.5) e pode ser feita uma analogia com a biologia já que com a reprodução em organismos biológicos assemelha-se ao processo de recombinação de genótipos de autómatos celulares. Na operação de recombinação, dois genótipos são selecionados e é efetuada uma "troca" de informação genética, originando um novo genótipo que combina características de ambos os autómatos celulares, sendo que este genótipo proveniente da operação genética de recombinação vai ser posteriormente adicionado à população, aumentando assim o número de elementos e a diversidade genética da população.

O principal objetivo deste capítulo é estudar de que forma a evolução de uma determinada população vai influenciar a estabilidade dos fenótipos, ou seja, estudar se os padrões e características dos fenótipos vão ser mantidas ao longo da evolução da população e também a estabilidade do genótipo, ou seja, observar se ao longo da evolução os genótipos vão tornar-se muito diferentes comparativamente com a população inicial. Para este estudo vão ser definidas três populações distintas de forma a abranger diversas possibilidades de populações.

As populações em estudo vão no estado inicial $t = 0$ possuir 10 genótipos (com os mesmos parâmetros, $n = 5$, $m = 3$ e $N = 125$) e a primeira população a ser estudada vai ser uma população que inicialmente contém genótipos e fenótipos idênticos, ou seja, a população inicialmente é constituída por 10 genótipos que são muito proximos entre si entre si do ponto de vista do genótipo e do fenótipo, com uma distância de Hamming média entre genótipos perto de 5. A segunda população em estudo vai conter genótipos muito variados, ou seja, genótipos completamente distintos entre si com uma distância de Hamming média próxima de 90.

A última população em estudo vai ser uma cópia da primeira população (com genótipos e fenótipos muito próximos) mas com uma diferença, foi adicionado um invasor à população. Este invasor é um genótipo completamente diferente dos existentes na população, sendo que esta população vai apresentar no instante $t = 0$, 11 genótipos, ou seja, os 10 genótipos da primeira população com a adição do genótipo invasor. Desta forma vai ser possível estudar diversas possibilidades de populações de autómatos celulares e conseguir perceber de que forma a evolução vai influenciar na estabilidade tanto do fenótipo como do genótipo.

De forma a quantificar as diferenças, tanto do fenótipo como do genótipo, vão ser consideradas as medidas definidas no capítulo anterior, ou seja, a distância de Hamming (definida na secção 3.1.2 para quantificar as diferenças ao nível do genótipo e o número de *keypoints* do algoritmo SIFT (definido na secção 5.1.3 para quantificar as diferenças ao nível do fenótipo. Todas estas medidas vão ser consideradas em média, ou seja, tanto para a distância de Hamming como para o número de *keypoints*, todos os genótipos e fenótipos, respetivamente, vão ser **comparados com todos os outros em todas as combinações possíveis**, sendo posteriormente efetuado o cálculo da média das medidas, sendo desta forma possível quantificar as diferenças e evolução da população. De seguida segue o pseudocódigo do algoritmo que permite quantificar uma população e que retorna a média de *keypoints* e a distância de Hamming média para todas as combinações possíveis:

Para ser possível quantificar a evolução da população vão ser considerados os valores médios das medidas referidas, ou seja, para quantificar as diferenças ao nível do genótipo e fenótipo vão ser efetuadas as leituras em $t = 0$ (população inicial), em $t = 30$ (meio da evolução) e $t = 60$ (população final) e posteriormente comparadas para perceber de que forma as semelhanças entre genótipos e entre fenótipos foram afetadas ao longo da evolução da população.

Neste estudo todos os autómatos celulares considerados apresentam as mesmas condições do estudo do capítulo anterior, com uma condição fronteira periódica, 180 células em cada autómato celular individual e 250 iterações temporais individuais para cada autómato celular, sendo que neste estudo todos os autómatos celulares apresentam um número de estados $n = 5$, um tamanho do genótipo de 125 (número de dígitos da sequência do genótipo N) e uma vizinhança $m = 3$, ou seja, são definidos em $\mathcal{G}_{5,3}$. Considerando a condição fronteira periódica, todos os autómatos celulares vão apresentar $\Gamma = (i_{N-1}, i_0) = (i_{124}, i_0)$.

6.1 Algoritmo de recombinação

O processo de recombinação genética entre autómatos celulares permite que a população evolua ao longo do tempo, aumentando o número de elementos da população e a diversidade de autómatos celulares existentes,

Algoritmo 2: Quantificar_População

Entrada: populacao
Saída: mediaDistanciaDeHamming, mediaNumeroDeKeypoints

```
1  início
       // Medidas de diferença
2      DistanciaDeHamming ← calcularDistanciaHamming();
3      NumeroDeKeypoints ← calcularNumeroDeKeypoints();
       // Acumuladores para as médias
4      mediaDistanciaDeHamming ← 0;
5      mediaNumeroDeKeypoints ← 0;
6      numComparacoes ← 0;
7      para cada genotipo1 em populacao faça
8          para cada genotipo2 em populacao faça
               // Não compara um genótipo consigo mesmo
9              se genotipo1 ≠ genotipo2 então
10                 mediaDistanciaDeHamming ← mediaDistanciaDeHamming +
                    DistanciaDeHamming(genotipo1, genotipo2);
11                 mediaNumeroDeKeypoints ← mediaNumeroDeKeypoints +
                    NumeroDeKeypoints(genotipo1, genotipo2);
12                 numComparacoes ← numComparacoes + 1;
13             fim
14         fim
15     fim
       // Divide pelo número de comparações
16     mediaDistanciaDeHamming ← mediaDistanciaDeHamming / numComparacoes;
17     mediaNumeroDeKeypoints ← mediaNumeroDeKeypoints / numComparacoes;
18     retorna mediaDistanciaDeHamming, mediaNumeroDeKeypoints;
19 fim
```

sendo que cada operação individual de recombinação vai ser definida como uma interação temporal t na população. Por exemplo, no instante $t = 0$ vamos estar perante a população inicial, que não sofreu qualquer alteração genética e por outro lado, no instante $t = 10$ já vamos estar perante uma população que sofreu 10 operações de recombinação, existindo já mais **10 elementos novos** na população, já que a operação de recombinação vai criar novos elementos da população (novos genótipos).

O algoritmo de recombinação é o que vai possibilitar a evolução da população e em cada operação genética de recombinação, ou seja, em cada iteração temporal t de evolução da população, são selecionados de forma aleatória dois elementos da população (genótipos) sendo depois estes dois elementos recombinados segundo uma sequência binária, também aleatória. Conforme já foi explicado na secção 4.5, é gerada uma sequência aleatória binária do mesmo tamanho dos genótipos e é posteriormente iterada de forma a gerar o novo genótipo, sendo que como existem dois dígitos nesta sequência binária, dependendo do valor na sequência (0 ou 1) é selecionado o dígito do primeiro genótipo (se for 0) ou o dígito do segundo genótipo (se for 1).

É necessário ainda considerar que ao longo da evolução da população não são removidos elementos, ou seja, em cada iteração temporal são selecionados de forma aleatória dois elementos, recombinados e o genótipo resultante é adicionado à população, ficando em subsistência com os seus "progenitores" e todos os outros elementos da população. Também é necessário considerar que não existem restrições na seleção de genótipos, ou seja, os dois genótipos selecionados podem voltar a ser selecionados novamente e como o processo de recombinação é aleatório, vão gerar (muito provavelmente) um genótipo diferente.

De seguida é apresentado o pseudocódigo do algoritmo de recombinação, que define o que ocorre em cada iteração temporal t. O algoritmo encontra-se divido em dois, sendo o algoritmo 3 referente à recombinação entre dois genótipos e o algoritmo 4 referente à recombinação da população (seleção de dois genótipos e posterior recombinação dos genótipos selecionados).

Algoritmo 3: Recombinacao_Genotipos

Entrada: genotipo1, genotipo2
Saída: genotipo_recombinado

```
1  início
2      sequencia_binaria ← lista vazia;
       // Gerar a sequencia binária aleatória
3      para cada i em genotipo1 faça
4          sequencia_binaria.adicionar(aleatorio(0, 1));
5      fim
6      genotipo_recombinado ← lista vazia;
7      para cada i de 0 até tamanho(sequencia_binaria) faça
8          se sequencia_binaria[i] = 0 então
9              genotipo_recombinado.adicionar(genotipo1[i]);
10         fim
11         senão
12             genotipo_recombinado.adicionar(genotipo2[i]);
13         fim
14     fim
15     retorna genotipo_recombinado;
16 fim
```

A implementação deste algoritmo e todo o código implementado para o estudo da estabilidade do fenótipo em autómatos celulares encontra-se no anexo A, com o algoritmo de recombinação a ser definido nas

Algoritmo 4: Recombinar_Populacoes

Entrada: populacao
Saída: populacao

```
// Retorna a população já com o novo elemento
início
    genotipo1 ← escolherAleatoriamente(populacao);
    genotipo2 ← escolherAleatoriamente(populacao);
    // Garante que os genótipos selecionados não são iguais
    enquanto genotipo1 = genotipo2 faça
        genotipo2 ← escolherAleatoriamente(populacao);
    fim
    população.adicionar(Recombinacao_Genotipos(genotipo1, genotipo2));
    retorna populacao;
fim
```

funções auxiliares, no anexo A.1, as populações a serem declaradas no anexo A.2 e o *script* principal de estudo da evolução de populações no anexo A.4.

6.2 Estudo de uma população com genótipos próximos

A primeira população em estudo vai ser a população a, que vai ser uma população que vai apresentar 10 genótipos muito idênticos entre si (distância de Hamming com média de 5.11) e é constituída inicialmente pelos seguintes genótipos:

$$a^{(1)} = 012140120102033300100001321143012210220032323230321020222042121$$
$$442313304304331212111043033240344203344441211000402110343043 33$$

$$a^{(2)} = \mathbf{3}121\mathbf{1}012010203330010000132114301221022003232323\mathbf{1}321020222042121$$
$$4423\mathbf{4}330430433121211104303324034420334444121100040211034304333$$

$$a^{(3)} = 01\mathbf{0}1\mathbf{2}012\mathbf{3}102033300100001321143012210220032323230321020222042121$$
$$4423\mathbf{4}330430433121211104303324034420334444121100040211\mathbf{0}04304333$$

$$a^{(4)} = \mathbf{2}1\mathbf{0}1\mathbf{2}012012\mathbf{1}20\mathbf{0}330010000132114301221022003232323032102022204212 1$$
$$4423\mathbf{4}330430433121211104303324034420334444121100040211034304333$$

$$a^{(5)} = 01\mathbf{0}1\mathbf{2}012\mathbf{4}102033300100001321143012210220032323230321020222042121$$
$$4423\mathbf{4}33043043\mathbf{1}312121110430332403442033444412110004021103430433 3$$

$$a^{(6)} = \mathbf{101}1\mathbf{2}012010203330010000132114301221022003232323032102022204212 1$$
$$4423\mathbf{4}330430433121211104303324034420334444121100040211034304333$$

$$a^{(7)} = \mathbf{2}1\mathbf{1}1\mathbf{2}0120102033300100\mathbf{4}1321143012210220032323230321020222042121$$
$$4423\mathbf{4}33043043312121110430332403442033\mathbf{2}4412110004021103430433 3$$

$$a^{(8)} = 01\mathbf{0}1\mathbf{3}0120102033\mathbf{1}0100001321143012210220032323230321020222042121$$
$$4423\mathbf{4}33043043312121110430332403442033444412110004021103430433 3$$

$$a^{(9)} = 01\mathbf{0}1\mathbf{2}012\mathbf{1}1020333001000013211430122102200323232303210202220421 21$$
$$4423\mathbf{4}3304\mathbf{2}0433121211104303324034420334444121100040211034304333$$

$$a^{(10)} = \mathbf{4}121\mathbf{2}0120\mathbf{4}0203330010000132114301221022003232323032102022204212 1$$
$$4423\mathbf{4}330430433121\mathbf{1}11104303324034420334444121100040211034304333$$

Após a população a estar definida, a mesma foi submetida a uma evolução com 60 iterações temporais, ou seja um $t = 60$ e os resultados obtidos encontram-se na tabela 6.1. A partir dos resultados obtidos é possível concluir que a evolução de uma população com genótipos idênticos vai criar uma população geneticamente idêntica, em que inclusive, ao longo da evolução temporal da população os genótipos apresentam uma tendência de se aproximar, com a distância de Hamming a apresentar-se menor ao longo da evolução comparativamente com a população inicial.

Relativamente ao fenótipo é importante realçar que nesta população inicial a todos os genótipos vão produzir fenótipos muito próximos, sendo que no estudo da população inicial existe um número elevado de *keypoints* devido a esta elevada semelhança. Ao longo da evolução e como seria expectável foi gerada uma maior diversidade de fenótipos, tendo o número de *keypoints* diminuído mas apenas de forma pontual, com o número de *keypoints* a manter-se estável desde $t = 30$ até $t = 60$, sendo possível concluir que ao longo da evolução da população, a maioria dos padrões e características dos fenótipos originais são mantidas pelos descendentes da população.

Iteração Temporal	Tamanho da população	Distância de Hamming Média	Média de Keypoints
$t = 0$	10	5.11	1027.93
$t = 30$	40	3.92	938.33
$t = 60$	70	4.01	944.06

Tabela 6.1: Tabela do estudo da população com genótipos idênticos

6.3 Estudo de uma população com genótipos variados

Após o estudo da população a com genótipos idênticos passamos para a população b, que é uma população que apresenta genótipos muito variados e distintos entre si. Esta população apresenta inicialmente uma distância de Hamming média entre genótipos de 88.5 e é definida pelos seguintes genótipos:

$$b^{(1)} = 2103400243200130120423112001301004400033424042010400004000 2101$$
$$1340434032322441441332332104142343424440433342320231324343421 33$$

$$b^{(2)} = 2\mathbf{02}3\mathbf{3}00\mathbf{011}2\mathbf{112}3\mathbf{1}12\mathbf{32}2\mathbf{2234}0\mathbf{1}1\mathbf{4301}0\mathbf{00222023}24\mathbf{21013102223}41\mathbf{0}1\mathbf{2}\mathbf{31}1$$
$$1\mathbf{01}0\mathbf{201}0\mathbf{4314}2\mathbf{113}44\mathbf{3410}33\mathbf{324100}234\mathbf{130}44\mathbf{31314011}42\mathbf{43121111342}\mathbf{2}33$$

$$b^{(3)} = 2\mathbf{0202}00\mathbf{012}2\mathbf{113032421310}1\mathbf{0}01\mathbf{14201}0\mathbf{3322214}4\mathbf{3}4\mathbf{32}2\mathbf{33}4\mathbf{222212}1\mathbf{0}\mathbf{1441}1$$
$$1\mathbf{432}4\mathbf{033}3\mathbf{3431}44\mathbf{313311}2\mathbf{44033303}234\mathbf{401}44\mathbf{140}3\mathbf{0230003}4\mathbf{3213}4\mathbf{1342}\mathbf{44}3$$

$$b^{(4)} = 21\mathbf{10}4\mathbf{1}0\mathbf{01}3\mathbf{122242003}4\mathbf{00}1\mathbf{331}0\mathbf{0}3\mathbf{401203112120400203322210222120021}$$
$$\mathbf{02}44\mathbf{23}0\mathbf{4411}2\mathbf{21232}1\mathbf{4010243}0\mathbf{3}11\mathbf{4}34\mathbf{0}4\mathbf{0}444\mathbf{1001404443040}3\mathbf{4203410343}$$

$$b^{(5)} = 21\mathbf{142}0\mathbf{31110}0\mathbf{434200}0\mathbf{331244}0\mathbf{122424}0\mathbf{202}0\mathbf{3013314302410}00\mathbf{1}04\mathbf{4310}\mathbf{14}1$$
$$\mathbf{0223011324444133323201144040301}34\mathbf{1013324}43\mathbf{0003}3\mathbf{04002042}343\mathbf{244}33$$

$$b^{(6)} = \mathbf{10112}0\mathbf{1}2\mathbf{0102}0\mathbf{3}3\mathbf{30010000}1\mathbf{321}1\mathbf{4301221}0\mathbf{22003}2\mathbf{323}2\mathbf{30321}0\mathbf{2}0\mathbf{222}0\mathbf{4212}$$
$$1\mathbf{4}4\mathbf{2343304}3\mathbf{0433}1\mathbf{2121110430332}4\mathbf{0}34\mathbf{42033}4\mathbf{4}4\mathbf{4121100}0\mathbf{402110}343\mathbf{043}33$$

$$b^{(7)} = \mathbf{1011}4\mathbf{1201212}0\mathbf{04}0\mathbf{304}4\mathbf{123031}0\mathbf{200}\mathbf{33023}00\mathbf{400}2\mathbf{1204202210043}00\mathbf{14132}$$
$$1\mathbf{202}4\mathbf{422}32\mathbf{14}2\mathbf{2341032}3\mathbf{3211023230}34\mathbf{113}444\mathbf{21}3\mathbf{0}3\mathbf{13}3\mathbf{12}2\mathbf{0243}4\mathbf{43}4\mathbf{4}1\mathbf{24}3$$

$$b^{(8)} = 2\mathbf{0203}00\mathbf{0}41\mathbf{2}\mathbf{1134133300}3\mathbf{421}0\mathbf{11440}1\mathbf{0}2\mathbf{2222123031231441222221}0\mathbf{1131}1$$
$$1\mathbf{0020}3\mathbf{334}232\mathbf{41}41\mathbf{1140013}\mathbf{2}2\mathbf{3}0\mathbf{0}11\mathbf{2}343\mathbf{04}44\mathbf{0}0\mathbf{04243013401}1\mathbf{0}\mathbf{2}2343\mathbf{01}133$$

$$b^{(9)} = 2\mathbf{02}32\mathbf{0}0\mathbf{0}432\mathbf{11224}1\mathbf{030}2\mathbf{1023}0\mathbf{1}11\mathbf{3}0\mathbf{1}04\mathbf{0222}3\mathbf{2}4\mathbf{0}4\mathbf{42014202221210}\mathbf{1331}1$$
$$13\mathbf{2420}4\mathbf{221}32\mathbf{3233233411412}3\mathbf{1}4\mathbf{311}343\mathbf{01}443\mathbf{0}3\mathbf{1}34\mathbf{111}2\mathbf{241}\mathbf{1}4\mathbf{124}43\mathbf{01}433$$

$$b^{(10)} = 2\mathbf{021}400\mathbf{031}2\mathbf{1104144413}0\mathbf{1}\mathbf{04}0113\mathbf{10}1\mathbf{0}\mathbf{322202}4\mathbf{4223}1010\mathbf{2224010100}\mathbf{1}1$$
$$13\mathbf{123}3\mathbf{232}132\mathbf{432230312}0\mathbf{3}\mathbf{21}1\mathbf{31231}343\mathbf{01}44\mathbf{122101014240}3\mathbf{443}4343\mathbf{210}33$$

Tal como para a população a, a população inicial b foi submetida a uma evolução com 60 iterações temporais e os resultados da evolução encontram-se na tabela 6.2. Nesta população e analisando apenas o estudo da população inicial ($t = 0$), é possível observar que existe uma diferença média de 88.51 dígitos entre todos os genótipos, ou seja, genótipos muito variados e que existem 556.73 *keypoints* em comum entre todos os fenótipos.

Ao longo do processo evolutivo da população e considerando $t = 30$ e $t = 60$ é possível observar uma diminuição considerável na distância de Hamming, pelo que é possível concluir que ao longo da evolução e das operações genéticas de recombinação, os genótipos vão se tornando mais idênticos entre si, algo que é justificado com o facto de que ao longo da evolução são efetuadas trocas de informação genética para os "descendentes", pelo que os genótipos vão se tornando mais idênticos apesar da grande discrepância inicial entre genótipos.

Em relação ao fenótipo é possível observar uma subida significativa na iteração temporal $t = 30$ e depois uma descida em $t = 60$, comparativamente com $t = 30$, contudo, em ambas as iterações o valor da média de *keypoints* é superior à média de *keypoints* da população inicial. Com toda esta informação é possível concluir que apesar de os genótipos e fenótipos serem inicialmente muito díspares entre si (existindo já uma grande diversidade genética), ao longo da evolução e com todas as trocas de informação genética, os genótipos e fenótipos foram ficando mais idênticos entre si.

Iteração Temporal	Tamanho da população	Distância de Hamming Média	Média de Keypoints
$t = 0$	10	88.51	556.73
$t = 30$	40	78.03	657.78
$t = 60$	70	75.94	621.66

Tabela 6.2: Tabela do estudo da população com genótipos variados

Finalizados os estudos das populações a e b é possível observar que no caso da população a, onde existe inicialmente uma proximidade genética muito grande, a população evoluiu e ao nível do fenótipo criou descendentes com uma variedade genética maior (apesar de pouco significativa) e na população b onde existia inicialmente uma grande variedade genética a população evoluiu e originou uma população mais idêntica entre si, tanto ao nível do fenótipo como do genótipo.

6.4 Estudo de uma população com um invasor

Para o estudo da população com um invasor vamos considerar a população inicial a onde vai ser adicionado um genótipo invasor cuja distância de Hamming perante todos os genótipos da população é superior a 90 dígitos, sendo o mesmo definido pelo genótipo seguinte (as diferenças destacadas é em relação ao genótipo $a^{(1)}$) :

$$a^{(11)} = \mathbf{2}1\mathbf{10}4\mathbf{100131}2\mathbf{2242}00\mathbf{34}00\mathbf{13}3\mathbf{10034}012\mathbf{0311}2\mathbf{12040020}3\mathbf{32}2\mathbf{2102}22\mathbf{1}$$
$$\mathbf{200}21\mathbf{02}4\mathbf{442}3\mathbf{04}4\mathbf{1122123214}0\mathbf{2430}3\mathbf{114}34\mathbf{04}0\mathbf{4}$$
$$\mathbf{4}4\mathbf{100}1\mathbf{4044430403420341034}3$$

Tal como nos os estudos anteriores, a população com um invasor foi submetida inicialmente a uma evolução com 60 iterações temporais ($t = 60$) e os resultados da evolução encontram-se na tabela 6.3. Considerando o instante $t = 0$, da população a (na tabela 6.1) e da população atual, é possível encontrar diferenças consideráveis, com a distância de Hamming média a passar de 5.11 (na população a), para 21.69 na população com um invasor e com o número de *keypoints* a diminuir aproximadamente 100 *keypoints*, na população com um invasor. Apesar das diferenças, estes valores iniciais vão de encontro com o que era esperado, uma vez que foi introduzido um genótipo completamente distintos dos existentes que afetou consideravelmente o "ecossistema".

Passando agora para a análise das outras iterações temporais ($t = 30$ e ($t = 60$) é possível observar que ao longo da evolução a tendência tanto dos genótipos como dos fenótipos é ficarem mais idênticos entre si, com o impacto inicial de um autómato celular invasor a ser anulado ao longo da evolução. Relativamente ao genótipo, a DH média passa do valor inicial de 21.69 para 10.61 na última iteração, ou seja, os genótipos ao longo da evolução ficaram consideravelmente mais idênticos entre si.

Considerando agora os fenótipos, as diferenças não foram tão consideráveis como no genótipo, mas existem diferenças, com o número de *keypoints* médio a aumentar 27.29. Apesar de não ser uma subida muito considerável, esta subida indica uma tendência crescente de semelhança, ou seja, ao longo da evolução temporal da população e apesar da existência de um invasor, os fenótipos tendem a aproximar-me em termos de semelhança.

De forma a confirmar a tendência da população "anular" o invasor, foi efetuada uma nova evolução da mesma população, mas com um número de iterações temporais maior, com $t = 100$. É necessário ter em consideração que o número de *keypoints* iniciais diferem do estudo anterior porque apesar de serem

Iteração Temporal	Tamanho da população	Distância de Hamming Média	Média de Keypoints
$t = 0$	11	21.69	928.63
$t = 30$	41	11.90	952.32
$t = 60$	71	10.61	955.92

Tabela 6.3: Tabela do estudo da população com genótipos idênticos mas com um invasor

os mesmos genótipos, as condições iniciais são aleatórias, pelo que os fenótipos vão ser significativamente diferentes.

Na tabela 6.4 encontram-se os resultados desta nova evolução que confirma o estudo anterior, com os genótipos e fenótipos a aproximarem-se geneticamente entre si ao longo da evolução. É também importante realçar que a partir da iteração temporal $t = 50$, apesar do elevado número de comparações de genótipos e de fenótipo possíveis de serem efetuadas, tanto a DH como a média de *keypoints* apontam para uma maior proximidade dentro da população.

Iteração Temporal	Tamanho da população	Distância de Hamming Média	Média de Keypoints
$t = 0$	11	21.69	919.63
$t = 50$	61	10.25	1034.49
$t = 100$	111	9.72	1035.66

Tabela 6.4: Tabela do estudo da população com genótipos idênticos mas com um invasor

6.5 Considerações finais

Em conclusão, apesar da evolução da população a ter levado a ligeiras diferenças médias no fenótipo, (este acontecimento é justificado com o facto de na população inicial a todos os genótipos selecionados originarem fenótipos idênticos) é possível observar que tanto na população b como na população com um invasor existe uma tendência da evolução conduzir a uma maior proximidade ao nível do fenótipo ao longo da evolução.

Ao nível do genótipo, é notório em todas as populações estudadas que existe uma maior proximidade média do genótipo à medida que a população evolui, sendo a maior diferença registada na população com um invasor, onde a DH diminuiu aproximadamente 50% ao longo da evolução.

Devido ao elevado número de combinações possíveis de serem efetuadas entre todos elementos da população (para efetuar a comparação de genótipo e de fenótipo), apenas foi possível realizar um número limitado de iterações temporais. Para um $t = 60$ e uma população inicial com 10 elementos já existiam 2415 comparações possíveis de serem efetuadas e se o número de iterações aumentasse para 100, já estaríamos perante 5995 comparações possíveis, algo que computacionalmente já é uma tarefa muito exaustiva.

Apesar do número limitado de iterações temporais, esta limitação não influenciou o estudo, tendo sido possível retirar conclusões na mesma e perceber de que forma as populações de autómatos celulares evoluem e de que forma os genótipos e fenótipos são afetados.

7

Conclusão e trabalho futuro

Esta dissertação teve como principal objetivo o estudo das dinâmicas evolutivas dos autómatos celulares quando sujeitos a operações genéticas. Tendo em consideração este objetivo, na primeira parte do trabalho foi implementado um sistema que permite gerar o fenótipo de um autómato celular (a partir de um determinado genótipo) e a partir de um determinado autómato celular inicial estudar de que forma as operações genéticas de mutação e replicação afetam os padrões e características do fenótipo.

A segunda parte do trabalho também vai de encontro com o principal objetivo deste trabalho, o estudo das dinâmicas evolutivas, mas com ênfase nas populações de autómatos celular. Para as populações também foi implementado um algoritmo que a partir de uma determinada população inicial pré-definida (conjunto de genótipos), evolui a população em t iterações temporais e que depois estuda a evolução da população ao nível da estabilidade do fenótipo e do genótipo.

Na primeira parte do trabalho (que se encontra no capítulo 5) o foco foi o estudo de autómatos celulares isolados, tendo sido efetuada uma análise que permitiu perceber de que forma a/as posições do genótipo sobre as quais é efetuada a operação genética de mutação vai afetar a estabilidade do fenótipo. Deste estudo foi possível concluir que, independentemente do autómato celular escolhido, existem operações singulares de mutação que destroem completamente o fenótipo e que existem operações complexas de

mutação (em quatro, cinco ou até mesmo seis dígitos do genótipo) que resultam em autómatos celulares cujos fenótipos são idênticos ao original, sendo a posição do genótipo escolhida para a operação, um fator decisivo para a manutenção de determinadas características visuais do fenótipo.

Deste primeiro estudo também foi possível concluir que a operação genética de replicação torna os autómatos celulares mais robustos perante mutações, com os autómatos celulares originais a serem mais sensíveis e a apresentarem maiores diferenças nos fenótipos quando sujeitos a operações de mutação comparativamente com os autómatos celulares provenientes da operação de replicação. Neste estudo foram escolhidos três genótipos distintos, com diferentes tamanhos e número de estados, e foi aplicada a operação genética de replicação a todos estes genótipos, individualmente. Posteriormente foram efetuadas todas as operações de mutações singulares e mutações em dois dígitos possíveis de serem efetuadas em ambos os genótipos (original e replicado) e comparado o nível de similaridade de todas as mutações com a realização original, tendo sido possível concluir que as mutações quando aplicadas ao genótipo replicado, resultam em fenótipos mais idênticos ao original, comparativamente com as mutações aplicadas ao genótipo original.

Na última parte do trabalho, que teve como principal objetivo o estudo da evolução de populações de autómatos celulares através da operação genética de recombinação e, como já referido, foi implementado um algoritmo que a partir de uma determinada população inicial pré-definida, evolui a população em t iterações temporais, utilizando a técnica de recombinação entre genótipos e que no final da evolução permite estudar de que forma os genótipos e os fenótipos acompanharam a evolução da população. Desta implementação foi possível estudar três populações distintas, uma população em que inicialmente existiam genótipos muito idênticos, uma população com genótipos variados e uma população com genótipos muito idênticos novamente, mas com a diferença que foi inserido um genótipo totalmente distinto dos outros (definido como um invasor).

Do estudo da evolução destas três populações foi possível concluir que a operação genética de recombinação aplicada a populações de autómatos celulares vai conduzir a uma evolução em que a tendência é a que os genótipos e os fenótipos se aproximem geneticamente, mesmo quando é introduzido um invasor numa população geneticamente idêntica, ao longo da evolução, os fenótipos e genótipos tendem a aproximar-se, anulando o invasor.

Por fim, o estudo efetuado sobre os autómatos celulares e sobre populações de autómatos celulares obteve bons resultados e permitiu concluir e perceber alguns aspetos importantes das dinâmicas evolutivas em autómatos celulares. Também a utilização do algoritmo SIFT para a comparação de fenótipos apresentou muitos bons resultados, tendo sido possível identificar fenótipos idênticos facilmente, através do número de *keypoints*.

Uma limitação deste trabalho é não ser possível comparar um número elevado de fenótipos, visto que vai ser computacionalmente exaustivo. Ao nível da comparação de fenótipos provenientes da operação genética de mutação, apenas foi possível comparar no máximo 1800 mutações (aqui era necessário gerar o autómato celular, guardar a imagem da realização e comparar com o fenótipo original) e ao nível das populações foram efetuadas no máximo 5995 comparações, sendo que aqui apenas era necessário comparar as imagens, visto que os fenótipos de os fenótipos de todos os elementos da população são gerados inicialmente, sendo apenas necessária a execução do algoritmo SIFT sobre as imagens que já estão geradas.

7.1 Trabalho futuro

Na área do estudo das dinâmicas evolutivas de autómatos celulares existem diversos estudos e experiências que podem ser efetuadas futuramente, sendo um estudo mais avançado na evolução de populações uma destas hipóteses. Um estudo mais avançado na evolução de populações seria, por exemplo, ao invés da

população evoluir e aumentar o número de elementos infinitamente (como acontece neste estudo), criar critérios para determinar quando ocorre a morte de um determinado elemento da população e evoluir a população desta forma.

Um critério para determinar a morte de um autómato celular pode ser, por exemplo, a sucessiva repetição do fenótipo, pois apesar de alguns autómatos celulares apresentarem um comportamento quase que aleatório, vai existir um ponto na iteração temporal do próprio autómato que os padrões vão começar a repetir, sendo aí o ponto que vai determinar a morte do autómato. Por esta lógica o nascimento do autómato celular seria dado na execução de uma operação de recombinação, a vida do autómato a sua realização e a sua morte quando o padrão começasse a repetir. Para implementar esta lógica seria necessário os autómatos celulares estarem a gerar o seu fenótipo ao mesmo tempo que ocorre operações de replicação, de forma a simular o desenvolvimento da vida de uma população.

Algo que também pode ser objeto de estudo futuro é a operação de *assembly* de genótipos, uma vez que neste trabalho apenas foi apresentada uma visão geral deste algoritmo.

A

Software desenvolvido

A.1 Script auxiliar de funções

```
1
2  from enum import Enum
3  from itertools import product, combinations
4  import re
5  import random
6  import numpy as np
7  import cv2
8
9  # Algoritmo de recombinação entre dois genótipos
10
11 def recombine(genotype1, genotype2):
12     binary_sequence = [random.randint(0, 1) for _ in range(len(genotype1))]
13     new_genotype = [genotype1[i] if binary_sequence[i] == 0 else genotype2[i]
       for i in range(len(genotype1))]
14     return ''.join(new_genotype)
```

```
15
16  # Função auxiliar do algoritmo de recombinação de populações
17
18  def recombine_populations(population):
19      genotype1 = random.choice(population)
20      genotype2 = random.choice(population)
21      while genotype1 == genotype2:
22          genotype2 = random.choice(population)
23      new_genotype = recombine(genotype1, genotype2)
24      return new_genotype
25
26  # Algoritmo de recombinação de populações
27
28  def recombine_populations_iterations(populations, num_iterations):
29      for _ in range(num_iterations):
30          populations.append(recombine_populations(populations))
31
32      return populations
33
34  class Boundary(Enum):
35      FIXED = 1
36      REFLEXIVE = 2
37      PERIODIC = 3
38
39  # Função para converter o genótipo em lista
40
41  def convert(string):
42      return re.findall('[0-9]', string)
43
44  # Função que retorna todas as combinações de vizinhança possíveis
45
46  def get_neighborhood_combination(combination, cellular_genotype,
        cellular_rules):
47      index = cellular_rules.index(combination)
48      return cellular_genotype[index]
49
50  # Função que vai retornar a vizinhança de uma determinada posição da linha do
        autómato celular
51
52  def get_neighborhood(list_elements, pos, neighborhood, automata_size,
        fixed_value, cellular_genotype, cellular_rules,
53                       boundary):
54      neighborhood_combination = []
55      # FRONTEIRA
56      pos_new = 0
57      if int(pos - (neighborhood / 2)) < 0:
58          if boundary == Boundary.FIXED:
59              for i in range(neighborhood):
60                  list_elements = np.insert(list_elements, 0, fixed_value)
61                  pos_new += 1
62          elif boundary == Boundary.PERIODIC:
63              pos_new += list_elements.size
64              list_elements = np.concatenate((list_elements, list_elements))
65          # Boundary.REFLEXIVE
66          else:
67              pos_new += list_elements.size
```

```
68              new_array = list_elements[::-1]
69              list_elements = np.concatenate((new_array, list_elements))
70      elif pos + (neighborhood / 2) + 1 > automata_size:
71          if boundary == Boundary.FIXED:
72              for i in range(neighborhood):
73                  list_elements = np.insert(list_elements, list_elements.size -
    1, fixed_value)
74          elif boundary == Boundary.PERIODIC:
75              list_elements = np.concatenate((list_elements, list_elements))
76          # Boundary.REFLEXIVE
77          else:
78              new_array = list_elements[::-1]
79              list_elements = np.concatenate((list_elements, new_array))
80      # TODOS OS OUTROS ELEMENTOS
81      start_pos = int(pos - (int(neighborhood / 2)) + pos_new)
82      for w in range(neighborhood):
83          neighborhood_combination.append(list_elements[start_pos])
84          start_pos += 1
85      tupla = tuple(neighborhood_combination)
86      return get_neighborhood_combination(tupla, cellular_genotype,
    cellular_rules)
87
88  # Função que vai realizar o cálculo MSE entre duas imagens
89
90  def mse(img1, img2):
91      h, w = img1.shape
92      diff = cv2.subtract(img1, img2)
93      err = np.sum(diff ** 2)
94      mse = err / (float(h * w))
95      return mse, diff
96
97  # Função que calcula a distância de Hamming para dois genótipos s1 e s2
98
99  def hamming_distance(s1, s2):
100     assert len(s1) == len(s2)
101     return sum(ch1 != ch2 for ch1, ch2 in zip(s1, s2))
102
103 # Gera as mutações possíveis para um determinado genótipo, fornecendo o número
    de mutações desejado
104 # Nos casos em que existem milhões de mutações é possível utilizar a parâmetro
    limit para limitar
105 # o número de mutações retornado
106
107 def mutations_generator(genotype, num_chances, num_states, limit=None):
108     mutations = set()
109     digits = list(genotype)
110     indices = list(range(len(digits)))
111     count_mutations = 0
112
113     for alteracoes in combinations(indices, num_chances):
114         for valores in product(range(num_states), repeat=num_chances):
115             mutacao = digits.copy()
116             for indice, valor in zip(alteracoes, valores):
117                 mutacao[indice] = str(valor)
118             if mutacao != digits:  # Verifica se a mutação é diferente do
    genótipo original
```

```
                if hamming_distance(genotype, "".join(mutacao)) == num_chances:
                    mutations.add("".join(mutacao))
                    count_mutations += 1
                if limit is not None and count_mutations >= limit:
                    return mutations  # Retorna as mutações geradas até o limite

        if limit is not None and count_mutations >= limit:
            return mutations  # Retorna as mutações geradas até o limite

    return mutations

# Função que vai gerar o autómato celular

def generate_celullar_automata(genotype, first_line, result,
    cellular_automata_neighborhood, boundary, size,
                                interations, fixed_value):
    cellular_automata = []
    next_line = []
    actual_line = first_line.copy()
    cellular_automata.append(actual_line.copy())
    for y in range(interations):
        line = np.array(actual_line, dtype=int)
        for x in range(size):
            next_line.append(get_neighborhood(line, x,
    cellular_automata_neighborhood,
                                              size, fixed_value,
                                              genotype,
                                              result,
                                              boundary))
        cellular_automata.append(next_line.copy())
        actual_line.clear()
        actual_line = next_line.copy()
        next_line.clear()
    return cellular_automata


# Função que gera as regras, de acordo com o número de estados e vizinhança
# Regras, 000, 001, 002.. De acordo com o número de estados e vizinhança

def generate_rules(cellular_automata_number_states_per_cel,
    cellular_automata_neighborhood):
    rules = []
    for x in range(cellular_automata_number_states_per_cel):
        rules.append(x)

    result = list(product(rules, repeat=cellular_automata_neighborhood))
    return result


# Função que compara dois fenótipos e retorna o número de keypoints
    correspondentes pelo algoritmo SIFT

def do_image_comparation(phenotype1, phenotype2):
    img1 = cv2.imread(str(phenotype1) + ".png")
    img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
    sift = cv2.SIFT_create()
```

```
    keypoints_1, descriptors_1 = sift.detectAndCompute(img1, None)
    bf = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)
    img2 = cv2.imread(str(phenotype2) + ".png")
    img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
    keypoints_2, descriptors_2 = sift.detectAndCompute(img2, None)
    matches = bf.match(descriptors_1, descriptors_2)
    return len(matches)
```

A.2 Script auxiliar que define as populações

```

population_initial_a = [
    "0121401201020333001000013211430122102200323232303210202222042
    12144231330430433121211104303324034420334444121100040211034304333",
    "3121101201020333001000013211430122102200323232313210202222042
    12144234330430433121211104303324034420334444121100040211034304333",
    "0101201231020333001000013211430122102200323232303210202222042
    12144234330430433121211104303324034420334444121100040211004304333",
    "2101201201120033001000013211430122102200323232303210202222042
    12144234330430433121211104303324034420334444121100040211034304333",
    "0101201241020333001000013211430122102200323232303210202222042
    12144234330430413121211104303324034420334444121100040211034304333",
    "1011201201020333001000013211430122102200323232303210202222042
    12144234330430433121211104303324034420334444121100040211034304333",
    "2111201201020333001000413211430122102200323232303210202222042
    12144234330430433121211104303324034420334244121100040211034304333",
    "0101301201020333101000013211430122102200323232303210202222042
    12144234330430433121211104303324034420334444121100040211034304333",
    "0101201211020333001000013211430122102200323232303210202222042
    12144234330420433121211104303324034420334444121100040211034304333",
    "4121201204020333001000013211430122102200323232303210202222042
    12144234330430433121111104303324034420334444121100040211034304333"]

population_initial_b = [
    "2103400243200130120423112001301004400033424042010400000400021
    01134043403232244144133233210414234342444043334232023132434342133",
    "2023300011211231123222234011430100022202324210131022223410123
    11101020104314211344341033324100234130443131401142431211113442233",
    "2020200012211303242131010011420103322214434322334222221210144
    11143240333343144313311244033303234401441403023000343213413422443",
    "2110410013122242003400133100340120311212040020332221022221200
    21024442304411221232140102430311434040444100140444304034203410343",
    "2114203111004342000331244012242402020301331430241000104431O1
    41022301132444413332320114404030134101332443000330400204234324433",
    "1011201201020333001000013211430122102200323232303210202222042
    12144234330430433121211104303324034420334444121100040211034304333",
    "101141201212004030441230310200033023004002120420221004300141
    32120244223214223410323321102323034113444213031331220243443441243",
    "2020300041211341333003421011440102222212303123144122222210113
    11100203334232414111400132230011234304440004243013401102234301133",
    "2023200043211224103021023011130104022232404420142022221210133
    11132420422132323323341141231431134301443031341112241141244301433",
    "20214000312110414441301040113101003222024422310102222401010O1
    1131233232132432230312032113123134301441221010142403443434321033"]
```

```

population_initial_c = [
    "012140120102033001000013211430122102200323232303210202 2
    204212144231330430433121211104303324034420334444121100040211034304333",
    "312110120102033001000013211430122102200323232313210202 2
    204212144234330430433121211104303324034420334444121100040211034304333",
    "010120123102033001000013211430122102200323232303210202 2
    204212144234330430433121211104303324034420334444121100040211004304333",
    "210120120112003300100001321143012210220032323230321022 2
    204212144234330430433121211104303324034420334444121100040211034304333",
    "010120124102033001000013211430122102200323232303210202 2
    204212144234330430413121211104303324034420334444121100040211034304333",
    "101120120102033001000013211430122102200323232303210202 2
    204212144234330430433121211104303324034420334444121100040211034304333",
    "211120120102033001000413211430122102200323232303210202 2
    204212144234330430433121211104303324034420334244121100040211034304333",
    "010130120102033101000013211430122102200323232303210202 2
    204212144234330430433121211104303324034420334444121100040211034304333",
    "010120121102033001000013211430122102200323232303210202 2
    204212144234330420433121211104303324034420334444121100040211034304333",
    "412120120402033001000013211430122102200323232303210202 2
    204212144234330430433121111104303324034420334444121100040211034304333",
    "211041001312224200340013310034012031121204002033222102 22
    120021024442304411221232140102430311434040444100140444304034203410343"]
```

A.3 Script do estudo das mutações e replicação

```
import os
from operator import itemgetter
import matplotlib.pyplot as plt
from functions import *

if __name__ == '__main__':
    fixed_value_boundary_condition = 0
    cellular_automata_size = 180
    cellular_automata_iterations = 250
    cellular_automata_number_states_per_cel = 2
    cellular_automata_neighborhood = 4
    cellular_automata_genotype = "0111100001111000"
    num_mutations = 2  # PARA O ESTUDO DAS MUTAÇÕES - NUMERO INDIVIDUAL DE
    ALTERAÇÕES/MUTAÇÕES EM ESTUDO
    MUTATION_STUDY = True
    result = generate_rules(cellular_automata_number_states_per_cel,
    cellular_automata_neighborhood)
    genotype = convert(cellular_automata_genotype)
    neighborhood = [Boundary.FIXED, Boundary.PERIODIC, Boundary.REFLEXIVE]
    first_line = []
    #Se necessário, colocar a linha inicial pretendida
    # first_line = []
    if len(first_line) == 0:
        for x in range(cellular_automata_size):
            first_line.append(random.randint(0,
    cellular_automata_number_states_per_cel - 1))
```

```
24      file_name_study = str(cellular_automata_genotype) + "_Boundary.PERIODIC.png"
25      check_file = os.path.isfile(file_name_study)
26      if not check_file:
27          for z in neighborhood:
28              cellular_automata_boundary = z
29              cellular_automata = generate_celullar_automata(genotype,
    first_line, result
30                                                             ,
    cellular_automata_neighborhood,
31
    cellular_automata_boundary,
32
    cellular_automata_size, cellular_automata_iterations,
33
    fixed_value_boundary_condition)
34              cellular_automata_final = np.array(cellular_automata, dtype=int)
35              img = plt.imshow(cellular_automata_final, interpolation='nearest')
36              img.set_cmap('hot_r')
37              plt.axis('off')
38              file_name = str(cellular_automata_genotype) + "_" +
    str(cellular_automata_boundary) + ".png"
39              plt.savefig(file_name, bbox_inches='tight', transparent='true')
40
41      if MUTATION_STUDY:
42          mutations = mutations_generator(cellular_automata_genotype,
    num_mutations, cellular_automata_number_states_per_cel)
43          #mutations = random.SystemRandom().sample(mutation, 300) # Recolher
    amostra aleatória, nos casos em que sao muitas mutações
44          print("Número de mutações: " + str(len(mutations)))
45          cellular_automata_boundary = Boundary.PERIODIC
46          errors = []
47          keypoints_study = []
48          values = []
49          count = 0
50          # IMAGEM DO FENÓTIPO ORIGINAL
51          img1 = cv2.imread(str(cellular_automata_genotype) + "_" +
    str(cellular_automata_boundary) + ".png")
52          img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
53          sift = cv2.SIFT_create()
54          keypoints_1, descriptors_1 = sift.detectAndCompute(img1, None)
55          bf = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)
56          # LINHA INICIAL ALEATÓRIA
57          first_line = []
58          for x in range(cellular_automata_size):
59              first_line.append(random.randint(0,
    cellular_automata_number_states_per_cel - 1))
60          for y in mutations:
61              genotype = convert(y)
62              cellular_automata = generate_celullar_automata(genotype,
    first_line, result,
63
    cellular_automata_neighborhood,
64
    cellular_automata_boundary, cellular_automata_size,
65
    cellular_automata_iterations,
```

```
66
    fixed_value_boundary_condition)
67
68              cellular_automata_final = np.array(cellular_automata, dtype=int)
69              img = plt.imshow(cellular_automata_final, interpolation='nearest')
70              img.set_cmap('hot_r')
71              plt.axis('off')
72              file_name = "temp" + ".png"
73              plt.savefig(file_name, bbox_inches='tight', transparent='true')
74              img2 = cv2.imread(file_name)
75              img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
76              keypoints_2, descriptors_2 = sift.detectAndCompute(img2, None)
77              matches = bf.match(descriptors_1, descriptors_2)
78              error, diff = mse(img1, img2)
79              errors.append(error)
80              keypoints_study.append(len(matches))
81              values.append(list())
82              values[count].append(y)
83              values[count].append(len(matches))
84              values[count].append(cellular_automata)
85              count += 1
86              print(count)
87
88          # ESTUDO
89          print("A média dos erros quadráticos médios entre todas as mutações é
    de: " + str(
90              sum(errors) / len(errors)))
91
92          # Media de keypoints
93          print("A média do número de keypoints entre todas as mutações é de: " +
    str(
94              sum(keypoints_study) / len(keypoints_study)))
95
96          # Número de mutações
97          print("\nO número de mutações é de: " + str(len(errors)))
98
99          ordered = sorted(values, key=itemgetter(1))
100         print("\nOs fenótipos com mutação menos idênticos ao original, por
    ordem, são: ")
101         for i in range(3):
102             cellular_automata_final = np.array(ordered[i][2], dtype=int)
103             img = plt.imshow(cellular_automata_final, interpolation='nearest')
104             img.set_cmap('hot_r')
105             plt.axis('off')
106             file_name = ordered[i][0] + "mutation_" + str(i) + ".png"
107             plt.savefig(file_name, bbox_inches='tight', transparent='true')
108             print(ordered[i][0] + " com o número de keypoints: " +
    str(ordered[i][1]))
109
110         print("\nOs fenótipos com mutação mais idênticos ao original, por
    ordem, são: ")
111         for i in range(3):
112             cellular_automata_final = np.array(ordered[count - 1 - i][2],
    dtype=int)
113             img = plt.imshow(cellular_automata_final, interpolation='nearest')
114             img.set_cmap('hot_r')
```

```
            plt.axis('off')
            file_name = ordered[count - 1 - i][0] + "mutation" + str(i) + ".png"
            plt.savefig(file_name, bbox_inches='tight', transparent='true')
            print(ordered[count - 1 - i][0] + " com o número de keypoints: " +
    str(ordered[count - 1 - i][1]))

        print("\n*****Lista de todas as mutações*****\n")
        for i in range(len(ordered)):
            print(ordered[i][0] + " com keypoints - " + str(ordered[i][1]))
```

A.4 Script do estudo da evolução de populações

```
import os
from operator import itemgetter
import matplotlib.pyplot as plt
from functions import *
from itertools import combinations
from populations import *

if __name__ == '__main__':
    fixed_value_boundary_condition = 0
    cellular_automata_size = 180
    cellular_automata_iterations = 250
    cellular_automata_number_states_per_cel = 5
    cellular_automata_neighborhood = 3
    cellular_automata_boundary = Boundary.PERIODIC
    t = 5

    print("Estudo da evolução da população")
    result = generate_rules(cellular_automata_number_states_per_cel,
    cellular_automata_neighborhood)

    population_initial = population_initial_c.copy()
    print("A população inicial: ", population_initial)

    aux_list = population_initial.copy()

    population_medium = recombine_populations_iterations(population_initial,
    int(t / 2))
    print("A população no meio da iteração: ", population_medium)

    aux_list_2 = population_medium.copy()

    population_final = recombine_populations_iterations(population_medium,
    int(t / 2))
    print("A população final: ", population_final)

    # Gerar os fenótipos de toda a população
    # Condição inicial aleatória
    first_line = []
    for x in range(cellular_automata_size):
        first_line.append(random.randint(0,
    cellular_automata_number_states_per_cel - 1))

```

```
    # GERAR FENÓTIPOS

    for auto in population_final:
        genotype = convert(auto)
        cellular_automata = generate_celullar_automata(genotype, first_line,
    result,

    cellular_automata_neighborhood,

    cellular_automata_boundary, cellular_automata_size,

    cellular_automata_iterations,

    fixed_value_boundary_condition)

        cellular_automata_final = np.array(cellular_automata, dtype=int)
        img = plt.imshow(cellular_automata_final, interpolation='nearest')
        img.set_cmap('hot_r')
        plt.axis('off')
        file_name = str(auto) + ".png"
        plt.savefig(file_name, bbox_inches='tight', transparent='true')

    # COMPARAR OS FENÓTIPOS ORIGINAIS

    keypoints_original = []
    hamming_original = []

    for phenotype1, phenotype2 in combinations(aux_list, 2):
        if phenotype1 == phenotype2:
            print("Existe algum genótipo igual")
        keypoints_original.append(do_image_comparation(phenotype1, phenotype2))
        hamming_original.append(hamming_distance(phenotype1, phenotype2))

    # COMPARAR POPULAÇÃO NO MEIO DA ITERAÇÃO

    keypoints_medium = []
    hamming_medium = []

    for phenotype1, phenotype2 in combinations(aux_list_2, 2):
        keypoints_medium.append(do_image_comparation(phenotype1, phenotype2))
        hamming_medium.append(hamming_distance(phenotype1, phenotype2))
        print(len(keypoints_medium))

    # COMPARAR POPULAÇÃO FINAL - ESTUDAR A EVOLUÇÃO

    keypoints_final = []
    hamming_final = []

    for phenotype1, phenotype2 in combinations(population_final, 2):
        keypoints_final.append(do_image_comparation(phenotype1, phenotype2))
        hamming_final.append(hamming_distance(phenotype1, phenotype2))
        print(len(keypoints_final))

    # RESULTADOS

    print("***** ESTUDO DA POPULAÇÃO INICIAL *****\n")
```

```
90
91      print("\n Tamanho da população: " + str(len(aux_list)))
92
93      print("\nA média da distância de Hamming entre todos os genótipos da
        população inicial é de: " + str(
94          sum(hamming_original) / len(hamming_original)))
95
96      print("\nA média do número de Keypoints entre todos os fenótipos da
        população inicial é de: " + str(
97          sum(keypoints_original) / len(keypoints_original)))
98
99      print("\n***** ESTUDO DA POPULAÇÃO EM t/2 *****\n")
100
101     print("\n Tamanho da população: " + str(len(aux_list_2)))
102
103     print("\nA média da distância de Hamming entre todos os genótipos da
        população é de: " + str(
104         sum(hamming_medium) / len(hamming_medium)))
105
106     print("\nA média do número de Keypoints entre todos os fenótipos da
        população é de: " + str(
107         sum(keypoints_medium) / len(keypoints_medium)))
108
109     print("\n***** ESTUDO DA POPULAÇÃO FINAL *****\n")
110
111     print("\n Tamanho da população: " + str(len(population_final)))
112
113     print("\nA média da distância de Hamming entre todos os genótipos da
        população final é de: " + str(
114         sum(hamming_final) / len(hamming_final)))
115
116     print("\nA média do número de Keypoints entre todos os fenótipos da
        população final é de: " + str(
117         sum(keypoints_final) / len(keypoints_final)))
```

Bibliografia

[AD08] Enrique Alba and Bernabé Dorronsoro. Introduction to cellular genetic algorithms. In *Cellular Genetic Algorithms*, pages 3–20. Springer, 2008.

[BD19] Kamalika Bhattacharjee and Sukanta Das. Random number generation using decimal cellular automata. *Communications in Nonlinear Science and Numerical Simulation*, 78:104878, 2019.

[CDM20] Umberto Cerruti, Simone Dutto, and Nadir Murru. A symbiosis between cellular automata and genetic algorithms. *Chaos, Solitons & Fractals*, 134:109719, 2020.

[CHH16] Lorena Caballero, Bob Hodge, and Sergio Hernandez. Conway's "game of life" and the epigenetic principle. *Frontiers in cellular and infection microbiology*, 6:57, 2016.

[Cio22] Gheorghe Ciolpan. Autómatos celulares. Accessed at 06/12/2022, 2022.

[CM99] Bastien Chopard and Alexandre Masselot. Cellular automata and lattice boltzmann methods: a new approach to computational fluid dynamics and particle transport. *Future Generation Computer Systems*, 16(2):249–257, 1999.

[dLDC99] Estéfane GM de Lacerda and ACPLF De Carvalho. Introdução aos algoritmos genéticos. *Sistemas inteligentes: aplicaçoes a recursos hıdricos e ciências ambientais*, 1:99–148, 1999.

[HCC+12] David HK Hoe, Jonathan M Comer, Juan C Cerda, Chris D Martinez, and Mukul V Shirvaikar. Cellular automata-based parallel random number generators using fpgas. *International Journal of Reconfigurable Computing*, 2012:4–4, 2012.

[ICS15] E. M. Izhikevich, J. H. Conway, and A. Seth. Game of Life. *Scholarpedia*, 10(6):1816, 2015. revision #150735.

[JLH+19] Paul C Jennings, Steen Lysgaard, Jens Strabo Hummelshøj, Tejs Vegge, and Thomas Bligaard. Genetic algorithms for computational materials discovery accelerated by machine learning. *npj Computational Materials*, 5(1):1–6, 2019.

[KCK21] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80(5):8091–8126, 2021.

[KP20] Dedy Kristanto and Windyanesha Paradhita. Simulation study of fluid flow and estimation of a heterogeneous porous media properties using lattice gas automata method. *J. Pet. Geotherm. Technol*, 1(2):71–82, 2020.

[Kun03] Daniel R Kunkle. Automatic classification of one-dimensional cellular automata. Master's thesis, Rochester Institute of Technology, 2003.

[LP90] Wentian Li and Norman H. Packard. The structure of the elementary cellular automata rule space. *Complex Syst.*, 4, 1990.

[LZ19] Dávid Lehotzky and Günther K.H. Zupanc. Cellular automata modeling of stem-cell-driven development of tissue in the nervous system. *Developmental Neurobiology*, 79(5):497–517, 2019.

[Mac93] M D Macleod. 14 - coding. In Fraidoon Mazda, editor, *Telecommunications Engineer's Reference Book*, pages 14–1–14–13. Butterworth-Heinemann, 1993.

[MCD+96] 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.

[Mel09] Gledson Melotti. Aplicaçao de autômatos celulares em sistemas complexos: Um estudo de caso em espalhamento de epidemias. Master's thesis, Universidade Federal de Minas Gerais, 2009.

[MM04] TG Mattos and JG Moreira. Universality classes of chaotic cellular automata. *Brazilian Journal of Physics*, 34:448–451, 2004.

[MSTMZ12] Genaro J Martinez, Juan C Seck-Tuoh-Mora, and Hector Zenil. Wolfram's classification and computation in cellular automata classes iii and iv. *arXiv preprint arXiv:1208.2456*, 2012.

[NZRC09] C. Narteau, D. Zhang, O. Rozier, and P. Claudin. Setting the length and time scales of a cellular automaton dune model from the analysis of superimposed bed forms. *Journal of Geophysical Research: Earth Surface*, 114(F3), 2009.

[RBTF21] Carlos Ramos, Nada Bouziani, El, Mouhaydine Tlemçani, and Sara Fernandes. Characterization of marble blocks. *5th International Conference on Numerical and Symbolic Computation: Developments and Applications Proceedings in digital support*, pages 381–394, 2021.

[RBTF23] Carlos Ramos, Nada Bouziani, El, Mouhaydine Tlemçani, and Sara Fernandes. Simulation of ideal material blocks using cellular automata. (submetido), 2023.

[RCC22] Carlos Ramos, Fernando Carapau, and Paulo Correia. Cellular automata describing non--equilibrium fluids with non-mixing substances. *Recent Advances in Mechanics and Fluid--Structure Interaction with Applications*, page 229–245, 2022.

[RR09] Carlos Ramos and Marta Riera. Evolutionary dynamics and the generation of cellular automata. *Grazer Mathematische Berichte*, 354:219–236, 01 2009.

[Sar00] Palash Sarkar. A brief history of cellular automata. *ACM Comput. Surv.*, 32(1):80–107, mar 2000.

[Sut89] Klaus Sutner. Linear cellular automata and the garden-of-eden. *The Mathematical Intelligencer*, 11(2):49–53, 1989.

[Tya20] Deepanshu Tyagi. Introduction to sift(scale invariant feature transform), 4 2020.

[VR09] Samuel Van Ransbeeck. Composição com autómatos celulares. Master's thesis, Instituto Politecnico do Porto (Portugal), 2009.

[Wol83] Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of modern physics*, 55(3):601, 1983.

[Wol86a] Stephen Wolfram. Cryptography with cellular automata. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 429–432, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[Wol86b] Stephen Wolfram. Random sequence generation by cellular automata. *Advances in Applied Mathematics*, 7(2):123–169, 1986.

UNIVERSIDADE DE ÉVORA
INSTITUTO DE INVESTIGAÇÃO
E FORMAÇÃO AVANÇADA

Contactos:
Universidade de Évora
Instituto de Investigação e Formação Avançada — IIFA
Palácio do Vimioso | Largo Marquês de Marialva, Apart. 94
7002 - 554 Évora | Portugal
Tel: (+351) 266 706 581
Fax: (+351) 266 744 677
email: iifa@uevora.pt