Østfold University College

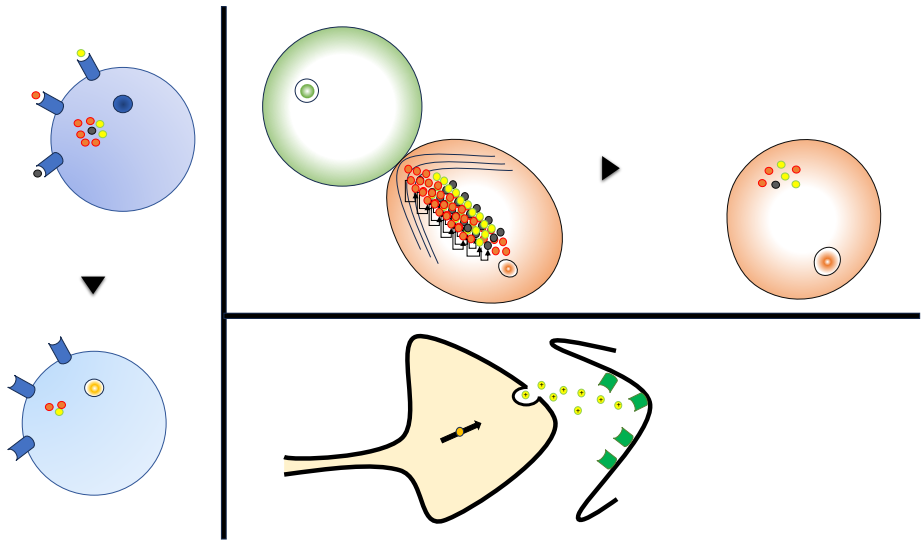
Master's Thesis

Sanyam Jain

AI Generating Algorithms with Self-Organizing Neural Cellular Automata

Master's Thesis in Applied Computer Science
Supervisors: Stefano Nichele, Felix Simon Reimers
June 2024

HiØ
Østfold University College,
Faculty of Computer Science, Engineering and Economics
Department of Computer Science and Communication



AI Generating Algorithms with Self-Organizing Neural Cellular Automata

Quantifying Genetic and Phenotypic Diversity, Emerging Growth, and Speciation

Master's Thesis

Sanyam Jain

Department of Computer Science and Communication
Østfold University College
Halden, Norway (1783)
June 25, 2024

Preface

This work is part of my Master's in Applied Computer Science final year thesis project (batch 2024). It explores various concepts studied during my MSc tenure, starting with artificial life models like elementary Cellular Automata (CA), Game of Life (Discrete 2D CA), Continuous CA (Lenia), and Neural CA. These models simulate dynamics when cells interact locally based on specific rules. Getting associated in research assistantship with my supervisor also helped me to explore these models. For instance, (1) Published article in Nordic Machine Intelligence (2024) on Frequency Histogram Coarse Graining where we used Elementary CA and 2D CA for the analysis, (2) Published article in Italian Workshop on Artificial Life (WIVACE 2023) conference at Ca'Foscari where we used Continuous CA Lenia to capture long term emerging dynamics and interestingness, and (3) Poster presentations at NORA Annual Conference, Tromsø (2023) and Nordic AI Meet, Copenhagen (2023). This thesis is in continuation and progression of all of our previous works.

Artificial Life addresses questions such as how simple entities become interesting through randomness and the emergence of intelligence, contributing to Artificial General Intelligence. To study these phenomena, we use models like CA, which are fascinating computational architecture where information processing, transmission, and storage are massively distributed and parallelized, with each component interacting locally with its neighbors. An example is the Game of Life, which is computationally universal. Recent work by Google on Growing Neural Cellular Automata (NCA), a continuous CA, uses neural networks to control shape growth and regeneration, resembling to biological morphogenesis. Our follow-up work uses NCA to study the emergence of intelligence, aiming to create a substrate that can produce long-term emergent dynamics.

Sanyam Jain
Halden, June 25, 2024

Abstract

The road from Artificial Intelligence (AI) to Artificial General Intelligence (AGI) requires advancements that can match human-level capabilities. The path essentially goes through Computer Science, Physics, Biology, Chemistry and Linguistics, if not any other. Our research thrives at the intersection of Computer Science and Biology and hence computational biology. On a broader view, in this thesis project, we try to embed a certain level of AI into a very small and simple (cellular) agents restricted within an environment with enough resources using local life-like interactions within them, for example in [33]. These agents can be thought of as an headless neural connections for now, where they are placed in an environment and tries to interact within. The hope is that such systems undergo sustained growth while revealing interesting properties, such as the formation of communities with specific biological properties and intelligence [38]. We call such system Neural Cellular Automata (NCA) built on fundamental Cellular Automata (CA) with a Neural operation layer. The proposed NCA system in this thesis is embedded with agency (brain and body). Neural CA architecture has been successfully trained to grow complex shapes and solve specific AI tasks stably [30], [31]. It has been shown that evolving non-uniform CA can self-organize into open-ended emergent dynamics [33]. This project combines the open-ended potential of evolving non-uniform CA with neural-based CA, aiming to create an ecosystem capable of generating intelligence and long term emergent dynamics. Specifically, the project aims to (1) explore the potential for open-endedness in evolving non-uniform neural CA, emergent long-term dynamics, and ways to incorporate tasks into the CA substrate so evolution can autonomously find solutions (becoming intelligent, automated discovery), and (2) find ways to quantify the emergent complexity in such substrates, focusing on complexity at different scales (phenotype-genotype), including complexity as a degree of hierarchy, coarse graining, and computing with particles [6], [25].

Keywords: Cellular Automata, Neural CA, Artificial General Intelligence

Acknowledgments

I sincerely thank to my star supervisor, Stefano Nichele, for his unwavering dedication and believe in me in order to introduce this interesting field of research with me and making me explore multiple venues with free mind as part of this process. His cutting-edge guidance, unbeatable insights, and constructive feedback have significantly enhanced the competitiveness of this thesis project. Addition to that, I extend my thanks to Felix, Dept of Computer Science and Communication at HiØ, Simula Research Laboratory (Research Council Norway), OsloMet university and Norwegian Artificial Intelligence Research Consortium (NORA) to provide enough resources and logistics during this time. I would also thank my mom, dad and sister to have enough confidence in me all this time.

Abbreviations

ALife Artificial Life

AGI Artificial General Intelligence

AI Artificial Intelligence

CA Cellular Automata

NCA Neural Cellular Automata

PD Phenotypic Diversity

GD Genotypic Diversity

CTFP Cellular Type Frequency Plot

GEP Global Entropy Plot

GCVP Gross Cell Variance Plot

CLOGV Cell Local Organisation Global Variance

GHC Genotypic Hash Coloring

RWSP Random Weight Selection Plot

CNWA Clustering Neural Weights Approach

DNA Deoxyribonucleic acid

GoL Game of Life

ECA Elementary Cellular Automata

MNIST Modified National Institute of Standards and Technology

SIM Self-Organizing Intelligent Matter

RNN Recurrent Neural Network

GPU Graphics Processing Unit

MSE Mean Squared Error

ANN Artificial Neural Network

PCA Principal Component Analysis

LCA Last Common Ancestor

ReLU Rectified Linear Unit

SLURM Simple Linux Utility for Resource Management

AWS Amazon Web Services

CUDA Compute Unified Device Architecture

Contents

List of Figures

List of Tables

List of Code

Chapter 1

Introduction

The natural sciences (but not limited to) physics, chemistry, and biology, have helped us understand life by explaining how matter, energy, and living organisms work. Physics has shown us the basic laws of the universe, chemistry has explained how different substances interact, and biology has taught us about living things and their processes. It is computer science and AI that are giving us new ways to learn about life and intelligence aiding the natural sciences. By using computers to simulate biological processes on fundamental physical laws and then analyze large amounts of simulation data, AI help us understand the details of how life works by reporting critical analytics. This new paradigm promises to bring even more discoveries and advancements in our knowledge of life. For example, AI can model the complex behaviors of cells and predict how they will respond to different treatments (along with environmental conditions), helping in the development of new medicines. It can also simulate brain functions (Neuro-AI) to understand how we think, learn and memorise, offering insights into cognitive science and neurology. Additionally, AI-driven analysis of genetic data can uncover the secrets of our DNA, leading to personalized medical treatments. As we continue to integrate AI with the natural sciences, we can expect to solve many of the mysteries of life and push the boundaries of what we know about our existence and intelligence. Following sections and subsections try to first build motivation (Life, Energy and Matter) using concepts from natural sciences, and then introduce computer science based approach to this inter-disciplinary study (ALife).

1.1 Motivation from the Origins

A major motivation for this research lies in Life, Energy, and Matter! In this section, we aim to provide some philosophical ideas and answers to the questions such as Evolution, Life, Intelligence, Entropy, Quantification, Emergence, etc. While it is important to picturize the motivation for the research thesis, it is much more relevant to study such paradigms to clearly understand the outcomes of the project. We intend to use these concepts in our ALife framework repetitively in rest of the thesis.

1.1.1 Evolution

Evolution is a continuous process of the development of Life on Earth that began billions of years ago. It reveals how a single uni-cellular organism has transformed into highly complex multi-cellular human beings. One animal can develop into a whole new species of

animals within evolutionary dynamics. A species is a category of animals that are capable of reproducing with one another majorly. Every creature (Life form) is made up of cells, which is further decomposed to the nucleus, which contains chromosomes. Chromosomes hold DNA that consists of different genes. Genes are life-information carriers, such as instructions and orders for the cells and determine the characteristics and traits (phenotype) that a creature may have. For survival, every species produces more offspring than required for their survival because of early death. Hence, more differences in traits occur. Two essential factors come into play in this process: recombination of genes and mutation via *random* perturbation due to which descendants receive a mix of the characteristics of their ancestors. Each of the varied offspring undergoes a natural process of selection to ensure only the right mix of traits thrive, increasing their chance of survival, and the right set of genes are passed to their offspring. It hints at the importance of variety, and creatures are well-adapted to their environment.

1.1.2 Life

There have been numerous ideas about Life for decades. We propose it as part of the evolution of a living cell or creature that abstracts the notion of keeping information hidden inside the genetic code, managing to ensure its continued existence. It is still a question of interest whether it is actually the dead matter (proteins) that together forms life, called a cell. We keep our focus on *cell* here for the relevance of the research and consider dead-matter theory for the core components of cell. A cell has an inherent cell wall that separates itself from the rest of the world to create order, regulates itself and maintains a constant state, consumes energy to stay alive, grows and develops, reacts to the environment, undergoes evolution, and self-replication. It is agreed that all cell components are not alive in themselves, and hence everything is dead matter inside. These dead matters undergo millions of chemical chain reactions forming complex proteins (complex machines). Cells are mindless robots that group together to form specialized tissues and then organ systems, be it heartbeat muscles or brain cells.

Cells (protein robots) are inherently interesting, which can be termed as *the language of life*, filled with millions of complex structures of proteins and simpler molecules like water. Complex and continuous self-replicating processes happen within cells. For criticality, cells have to constantly work to keep themselves from achieving entropy between boring and dead. Work done in order to *self-maintain* consumes *energy*. The energy-building protocol of life is Adenosine Triphosphate (ATP). Cells break down ATP molecules in the system and consume chemistry to heal and thrive. Overall, the dynamic and intricate processes within cells, involving complex protein structures, self-replication, and energy consumption through ATP, result life.

1.1.3 Emergence, Intelligence and Consciousness

Emergence describes smaller things forming bigger things that have different properties than the sum of their parts. It's a *complexity arising from simplicity*. For example, atoms form molecules, molecules form proteins, proteins make up cells, cells give rise to organs, organs form creatures, then societies. And hence at each level, a rule set is followed that makes order through chaos. Within the open-ended environment, communication within numerous cells gives rise to complex and critical behaviors to maintain itself. Cells combine, cooperate, and specialize to respond to one another over time to develop into

more complex structures. This again leads to a final question: how do dumb individual cells know what to do? Cells communicate with their neighboring cells via chemical information and then decide what to do. This is how cells become intelligent to trigger the right information at the right time to stay alive, which we also call the emergence of intelligence. While chemistry helps simpler organisms survive with some level of intelligence, perception helps them guide and make decisions based on the perceived environment. A topology of long-term memory and widening the sensory range enables inner representation of the world even in the absence of the environment. Together, it gives rise to consciousness. It can be hypothesized as a motion of a self towards a source of energy. This also prolongs survival over their competitors.

1.2 Artificial Life

Artificial Life (ALife) is an interesting exploration to create systems mirroring lifelike properties and computational complexity, delving into the fundamental principles of living systems. ALife seeks not only to understand but also to replicate or extend these principles through artificial means, parallel to biological systems and intelligence. Unlike the more task-centric focus of traditional AI, ALife, contrary to a concept loosely resembling AGI, envisions a broader, long term novelty driven goals. Open-endedness, a key concept in ALife, entails the ability of a system to perpetually generate an infinite variety of novel and meaningful outcomes, fostering ongoing evolution, adaptation, and the emergence of unpredictable behaviors. These open-ended ALife systems exhibit autonomy and self-organization, i.e., entities organizing themselves into diverse higher-order structures within their environment, giving rise to complex patterns and behaviors without explicit external control. This intersection of intelligence, complexity, and self-organization in ALife encourages not just the resolution of specific tasks but an exploration of vast solution spaces, perpetuating self-replication, adaptability, and evolvability. The core essence of ALife lies in understanding the fundamental principles of life, fueling a continual exploration of possibilities and contributing to the emergence of complex and intelligent behaviors.

For example, in recent work [44], authors explain how open-endedness and complexification are intrinsically linked, with endless variation, sensitivity to initial conditions, selection pressure, and the interplay of genetic and phenotypic changes over a dynamic landscape. In the absence of specific tasks, the evolution of smart and interesting phenomena occurs organically over this task landscape. The concept of endless variation becomes pivotal, implying that without complexification, any fixed level of complexity would inevitably exhaust existing interestingness or variation. Evolvability, facilitated by heritable genetic traits and selectable phenotypes with variation, is fundamental; without it, there will be no discovery of new behaviour. The changing task environment, whether self-directed or evolving together, includes changes that help organisms (agents) adapt, act independently through inherited traits, and make adjustments for their long term survival.

The idea of abstraction of real-world cells as intelligent entities capable of reproduction, decision-making, and adaptive responses is intriguing. Consider a cellular embryo knowing what to create and where to create it. In a recent study on synthetic biology [38], the author explores the creation of new living forms. This study, along with an experimental approach [27], challenges the conventional blueprint-driven perspective of cellular development. It questions the idea that genetic information alone determines complex morphologies. Unlike the traditional belief that DNA encodes specific blueprints for organs or tissues, synthetic

biology recognizes the dynamic nature of cellular behavior. Cells, seen as active entities, engage in communication and respond to external signals, giving rise to diverse structures and functions (phenotypical traits). This paradigm shift underscores the significance of cell communication via chemical signals, mechanical signals, and bioelectric signaling (as shown in Figure 1.1) in shaping biological forms, as extensively studied in [30].

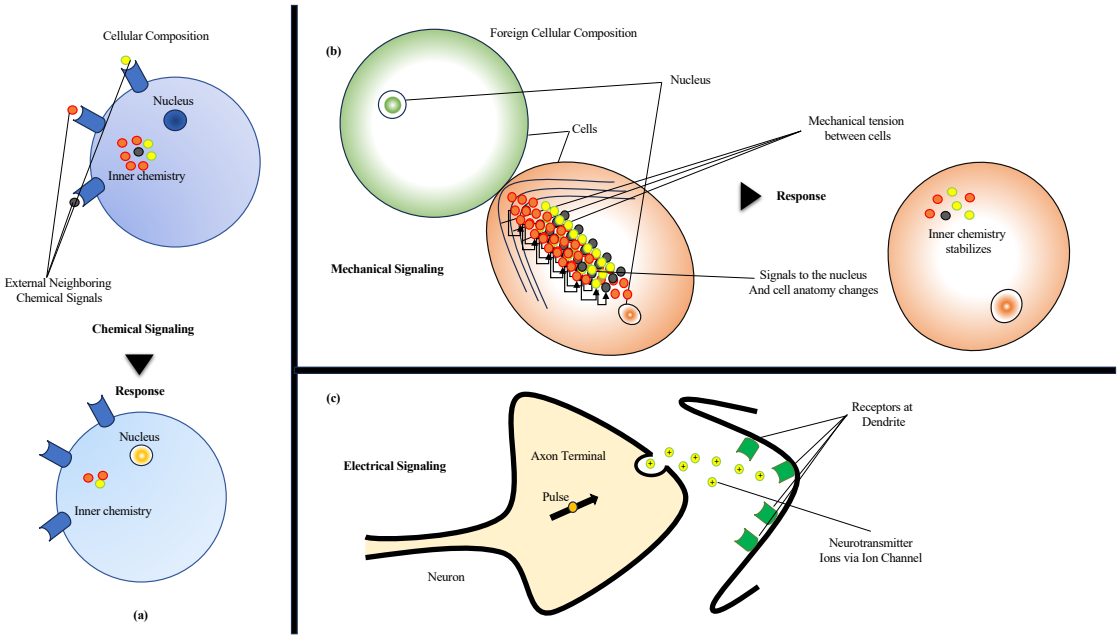


Figure 1.1: Intra-cellular communication via three signals types. (a) Chemical Signals, (b) Mechanical Signals and (c) Bio-electrical Signals

Philip, in [38], also discusses four categories for the classification of cellular developmental agency, including (1) agents attaching to one another, (2) assembly via swarms, (3) complex behavior through growth and multiplication, and (4) self-replication or the generation of a clone through algorithmic repetition, such as natural fractals, also shown in Figure 1.2 with respective captions inside. Mitchell's work in computational complexity [9], [11] also supports and verifies this classification. We have seen how [33] utilised these concepts, especially cell communication via chemical, enzymes and energy and neurons in their work for evolving complex behaviour or AI Generating Algorithm (AIGA) [24]. Additional literature [23], [32] aligns as well with the discussion.

To study such complex and dynamical behaviours, researchers have been utilising various tools to perform experiments *In vitro*, *In vivo* and *In silico* for different use cases discussed in previous paragraph. Our focus in this research work is *In silico*. In the next Chapter we discuss and detail the systems that are widely utilised to study emergent behaviour within a computational model giving opportunity for endless variation and open-ended evolution. ALife research is of importance in advancing our comprehension of life, biology, and complex systems, with applications across various domains. Through the simulation of life-like processes, this research delves into fundamental questions concerning the emergence of complexity, evolution, self-organization, contributing to the development of biologically-inspired algorithms, and adaptive systems. This research could be crucial for AGI, driving interdisciplinary advancements and offering valuable insights, ultimately

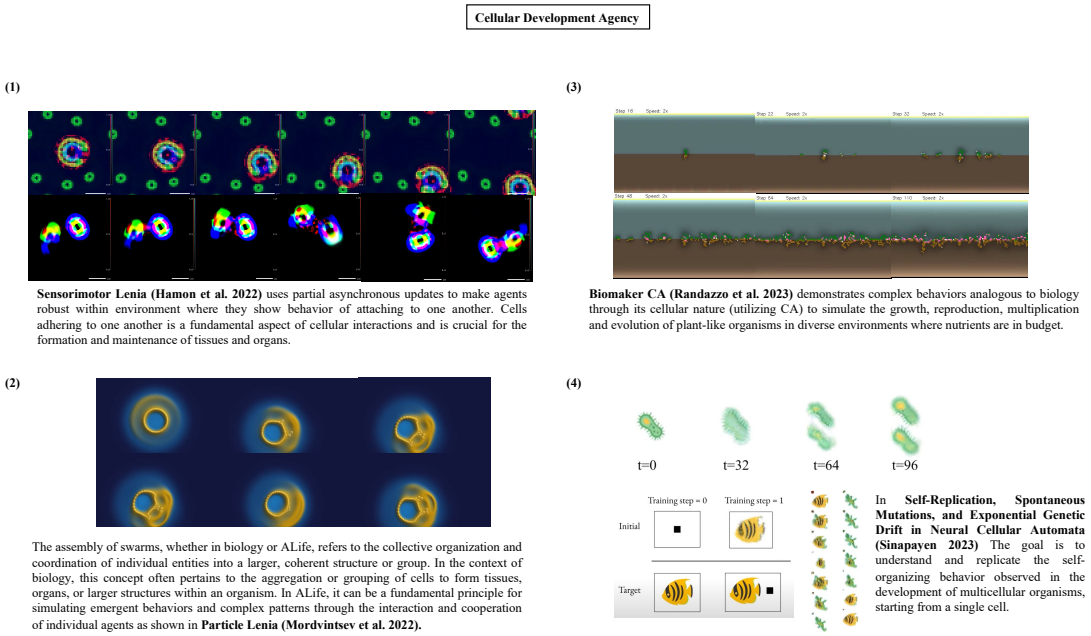


Figure 1.2: (1) introduced in [35], (2) in [36] , (3) in [44] and (4) in [45]

bridging the computational biology and AI. Following subsections related ALife introduces few of the concepts that are useful for developing advanced concepts in rest of the thesis.

1.2.1 Open-Endedness

Open-Endedness in AI means creating systems that can keep learning and growing on their own, discovering new things and solving new problems without needing constant human input. Imagine you have a video game where the levels keep changing and getting harder each time you play, but there's no final level or ultimate goal. Instead of just following preset levels, the game can create endless new challenges and surprises based on how you play. This is similar to open-endedness in AI, where the AI keeps learning and improving by exploring new possibilities and creating new solutions, rather than just sticking to a fixed set of tasks. For example, an AI designed for art could keep inventing new styles and techniques, producing unique pieces of art that no one has ever seen before, always exploring and expanding its creative abilities. One example is picbreeder [10] very well discussed in [16] how a user was able to evolve a butterfly, or a car without any predetermined notion of creating anything. In simpler words, *the best way to generate an art in picbreeder is not to generate it.* Formulating it, [26] Open-Endedness is beyond an algorithmic explanation that is driven by increasing complexity and creativity resulted by interactions of stupid entities to become informative itself (should not be confused with evolution). In other words, if by all means we can build an Open-Ended algorithm that corresponds to the one that drives nature, then it allows us to reveal origin, emergence of intelligence and many other interesting things. Considering this fact, it is easy to take analogy from humans (and hence intelligence) are result of an Open-Ended process of nature. This means it could be possible that Open-Endedness itself is a core component of emergence of intelligence. In [21], four necessary ingredients for Open-Endedness are referred. Those conditions are as follows.

1. *A rule should be enforced that individuals must meet some minimal criterion (MC) before they can reproduce, and that criterion must be nontrivial.*

    - *Corollary: The initial seed (from which evolution begins) must itself meet the MC and thereby be nontrivial enough to satisfy Condition 1.*

2. *The evolution of new individuals should create novel opportunities for satisfying the MC.*

3. *Decisions about how and where individuals interact with the world should be made by the individuals themselves or tractable environment.*

4. *The potential size and complexity of the individuals' phenotypes should be (in principle) unbounded.*

1.3 Research Questions and Objectives

The primary objective of our research is to conduct a comprehensive analysis of the evolved ALife model (or a substrate), building on the principles outlined in the earlier introduction. The research aims to delve into the intricate dynamics and measure the growth of complexity of the model based on interactions, primarily in a simulated (*In silico*) environment. Motivated by the foundational concepts of ALife, specifically the open-ended evolution and self-organization inherent in such frameworks, the study focuses on understanding the species composition and diversity that emerge through this computational model via self-replication, clone and mutation. To perform this analytical study, we will employ a set of carefully formulated tools inspired by the concepts of Genetic Diversity (GD) and Phenotypic Diversity (PD). The research questions driving this investigation revolve around understanding the dynamics of the model, with a particular focus on how these tools can contribute to the analysis of species composition and diversity, both genotypically and phenotypically. Overall, we aim to study the framework on the verge of evolving complexity and measuring at two scales PD and GD. We envision to propose tools for both type of analysis. In the rest of this thesis, we'll be using a framework based on cellular automata (CA), where each element of the model is a cell with some genetic material. These cells interact with each other based on conditions we set up beforehand. We think of this model as a closed system with an endless supply of energy, therefor no external adversary. So, the environment where these cells interact only stops working when enforced (with a life budget). Other than just living, these cells are allowed to self-replicate by cloning and inheritance. For this Chapter, we call such ecosystem as substrate and formally define CA and Neural CA (NCA) at later chapters.

First, we design and develop an artificial substrate *in silico* that is comparable to a substrate *in vitro*. Second, we seed few *founders* or living cells in beginning of evolution and let them evolve. The evolution is based on self-replication, inheritance and mutation to promote overall growth. Third, we analyse and capture the emerging complexity using various PD and GD tools. Finally, we report corresponding analysis in the form of generation plots, speciation plots and genealogy plots. These plots are expected to abstract the dynamics of complex cellular activities throughout the generations. We start this thesis with following research questions and ideas that we want to answer by experimenting, analysing and studying this model and corresponding outcomes. These research questions

not only put our framework in court of interpretability but also outline the major results we expect at the end of this thesis. We answer these questions in rest of the chapters.

Research Questions:

RQ1 What are the key species that emerge, survive, reproduce and becomes extinct (dead) in the evolved substrate?

In a dynamic environment where only way of survival for agents is local interactions (and reproduction via replication) with potential changes (or mutations), essentially, at the time of inheritance; It is intriguing to discover how such dynamics make the system stable with specific type of traits that enable them to grow, collaborate for survival, die, or becomes extinct because of new species invading the system.

RQ2 How does the genotypic diversity evolve over successive generations of the substrate?

Within this substrate, we also envision to track the shared genetic traits over generations among different agents. This is to ensure, even if the environment may have similar phenotypic traits in long-term, we should be able to check how genes have traveled as genealogy trajectories. It could also benefit us in analysing what specific genes stays over long generations and which not.

RQ3 What is the impact of the self-replication and self-maintenance processes, characteristic of autopoiesis, on the genetic makeup of the evolving substrate?

Autopoiesis refers to a system's ability to self-produce and self-maintain its own organization, emphasizing self-sustainability. The self-replication process inherently involves copying genetic information. Over successive generations, mutations may occur, contributing to genetic diversity. Genetic diversity is crucial for adaptation. The self-maintenance processes ensure the persistence of functional and viable entities, allowing for the selection and propagation of advantageous genetic traits. Autopoiesis ensures that entities maintaining themselves successfully have a chance to persist, leading to the potential emergence of novel and adaptive genetic information.

RQ4 How do phenotypic variations manifest within the substrate, and what are the underlying factors contributing to this diversity?

The phenotypic variations manifest as observable differences in traits of individual entities. In our system, we expect the major contributions towards the phenotype diversity is led by intra-cellular communications, evolutionary pressures driving the selection of specific genetic traits, leading to the emergence of phenotypic variations that confer advantages in terms of survival and reproduction. And inherent stochasticity of the environment can lead to diverse population.

RQ5 To what extent do the cellular activities within the substrate exhibit sensitivity to initial conditions (for example number of time steps) and how does this influence the genetic and phenotypic outcomes?

For the proposed methods, we plan to investigate multiple types of initial conditions for the environment that could lead to varied results to analyse. For example, the size of the environment and its compactness in which agents interact, number

of generations to evolve, the amount of genetic information to transfer, degree of perturbation, the threshold below which all agents die, etc.

Objective:

This thesis is multi-objective. The primary objective is to analyse genotypic and phenotypic relationships that may build over generations within the system. Specifically, how simple and local cellular interactions gives rise to intricate emerging properties and behaviour. To achieve such objective, we have intermediate objectives including building and testing the substrate (the environment) where agents have favorable conditions to locally communicate, reproduce, inherit and mutate. The objective is not only to evolve such agents and environment, but also to measure the growth, diversity and speciation of the evolved agents using mathematical models that could further help us achieve answers to the research questions discussed previously.

1.4 Overview of Research Methods

In this study, our aim is to comprehend the dynamics and measure the growth of complexity in substrate, building upon previous work in the field of wetware technologies [40]. Employing tools tailored for the examination of genotypic and phenotypic levels [15], [25], we introduce two genotypic tools and four phenotypic tools. These tools provide valuable insights into genetic variations, evolutionary patterns, unique gene counts, and the behavior of cellular states. Genotypic tools visually represent genetic information, capturing the evolution of randomly chosen genes and tracking gene behavior and trajectories. On the phenotypic side, tools calculate frequencies to observe emergent species, quantify system-level diversity, measure individual cell state variations, and evaluate local organization of cell states. Please note that we use words *agents* and *cells* interchangeably for specific concepts. For PD level tools we use *cells* and for GD level tools we use *agents*.

1.5 End Deliverables

This project involves dealing with large volumes of data. We divide the project and the simulation process in two sub-parts. First part involves training the system parameters (also called as genes or weights parameters of the neural network) and system environment parameters (or grid values). These parameters are exported as pickle (package) file optionally. Second part, we unload these weights parameters separately to perform the PD and GD analysis. This decision was done in order to balance load on the computational resources and use these pickles to use anytime without running the complete system again. The final deliverable include the system environment (NCA grid values and weight parameters if required) variables and the corresponding results of PD and GD tools. We release following deliverables with this thesis:

1. The running code is available at GitHub repository `https://github.com/s4nyam/Self-Replicating-NCA`.

2. All *tar* files as part of compressed experimental data for small runs and large runs are available at `https://archive.org/details/@evolutionary_lenia`.

3. We also release high quality resulting animations as YouTube playlist at `https://www.youtube.com/@growingnca/playlists`.

4. Google Colab Notebook is also provided to try small runs on the go in small runtime environments at `https://bit.ly/neuralCA`.

1.6 Thesis Outline

This segment offers a roadmap for navigating the remainder of the thesis. Following the introductory chapter (this), we proceed with related work and a literature survey where we discuss few of the recent works in this paradigm. We focus our survey specifically in a manner to study substrate level work and corresponding analysis driven work. In simpler words, we thoroughly study types of alternatives for framework that can represent our ALife substrate. We also study how emerging complexity can be represented and analysed critically on the basis of mathematical models (statistics) and physical concepts (entropy). Subsequently, we delve into the details of the proposed methodologies and implementation of the selected ALife framework along with the PD and GD tools. Lastly, we present the plots and figures as results to provide support to the research questions and their answers following discussions, and conclusions. I hope you, as the reader, will remain as enthusiastic throughout the duration of this thesis as I have been while working on it, and that you enjoy reading it.

Chapter 2

Related Work and Literature

Firstly, we start with explanation of the research topic that we have chosen for this thesis work and hence debunk the research title and overall idea of the topic. Secondly, we study literature related to the simulation framework or system that we have chosen which is Neural CA (our substrate). Thirdly, we study tools that are proposed to measure emerging growth and complexity in such frameworks like discrete CA, Lenia or NCA. However as we will see in further chapters, we restrict our focus in proposed methodology and implementation to Neural CA. We also study genotypic and phenotypic level ideas from recent work that will be described in later sections of this chapter. It is highly possible that tools and framework that we are going to study maybe described for a different experimental setup or use case in the origin work. It is also worth noting that literature we picked to study and analyse is ideated by three factors: (1) NCA Framework or substrate that also works as environment or biome where cells (or agents) evolve, grow and interact. (2) Analytical tools or metrics that can help in performing quantitative (phenotypic) and qualitative (genotypic) studies on such emerging complex behavioural dynamics. (3) Some other miscellaneous concepts related to applications of this research in morphogenesis.

2.1 Background

Systems that can undergo self-replication, self-production and self-maintenance (together called as autopoiesis) introduced in [5] are fundamentals to study ALife. An autopoietic system is characterized by a network of processes and components that work together to maintain and reproduce the organization of the system itself. The key idea is that the processes within the system continuously contribute to the production and maintenance of the system's components, and these components, in turn, participate in the same processes that produced them. It's a concept that explores the self-sustaining nature of living entities and their ability to autonomously regulate and maintain their internal organization finally giving rise to a complex emergent phenomenon.

Numerous tools and concepts have been proposed to simulate such phenomena in the realm of ALife. A focal point of enduring interest lies in Elementary CA [1], [6], 2D CA, and the Game of Life (GoL) [2], [3] which are very early work in self-replication, all of which have undergone extensive examination in previous years. Nevertheless, recent research has delved into more intricate systems, such as exploring the 2D-CA rule space through evolution [34] as detailed in [42] to evolve complex behaviours in discrete space. Moreover, a departure from discretizing the search space has been witnessed in the application of the

concept of exploiting continuous search space, in Lenia [22], [28] and its newer versions like Sensorimotor-Lenia [35] where the self-organising agents are controlled by a feedback loop using auto-differentiation such that these agents (organisms) shows properties like intelligence, self-maintenance, cognition and robustness to external adversaries in new environment governed by the laws of physics applied on atomic elements. Particle-Lenia [36] utilises physio-chemical concepts for energy based particle systems and their morphology based on local interactions producing emergent agency (for example, behavior of microscopic organisms in water), Flow-Lenia [37] and Evo-Lenia [39], [43].

Finally, the driving motivation for this research is Neural CA (NCA). In this work, we exclusively use simplest models of Uniform and Non-Uniform NCA which are discussed in much great details in upcoming sections. Growing NCA proposed in [30] is a classical example of self-assembly and autopoiesis (self-maintenance). The idea that was discussed earlier and also explained in [38], which is prominent in all living organisms operating at different scales of cellular activities and communication within neighborhoods (morphogenesis in multi-cellular organisms), such that each cell knows what to grow and where to grow. The idea of utilising a small neural network that work as brain and cognition for each cell is differentiated in order to learn policies so that they self-organise to result into intricate anatomical structures. In Growing NCA, first they experiment to grow cells intelligently, which leads to another paradigm to learn to stop and how much of growth is required. Finally, learning to regenerate resembling the self-healing nature of living organisms. Similar study performed in Self-Classifying MNIST digits [31] answers to the question on evaluation of the growth towards a specific target morphology, the mechanism by which cell groups ascertain the correctness of an organ or tissue pattern and decide whether the existing anatomy requires remodeling towards a specific target morphology, exemplified by phenomena, for example, in [17] authors discuss about regenerating unexpected anatomies via bio-electrical signals. They suggests that cells receive this information through bioelectric signaling, a mechanism observed in various organisms, including fish, frogs, and humans, and when manipulating planarian fragments to modify their bioelectric state, unexpected anatomical outcomes, like worms with heads at both ends, were produced by regenerating cells. Additionally, they also perform perturbations to analyse development of neural structures in the frog brain. This phenomenon is also studied computationally in [45]. We study [20] in details and other relevant cited references in computational simulation (implementation) details as part of literature and related work in further sections. Even though exploring complex dynamics of all these venues is worth visiting, but we keep our scope of experimentation and research to perform fine-grained and coarse-grained analysis on Non-Uniform Self Replicating NCA.

2.2 Literature Study and Related Work

In this section, we will narrow our discussion to specific literature and related work that is going to be crucial for rest of the research. We start with understanding different frameworks, that include discrete or classical Cellular Automata (CA), Continuous CA and Neural CA. Within this subsection, we also outline, how these frameworks have been modified in different ways to achieve robustness and their complexification for interesting dynamics. Next to that, in another subsection we analyse numerous tools to measure PD, GD and their interplay (PD to GD relationships and vice versa). Finally in the last subsection, we study miscellaneous concepts and related work that are borrowed as

evidences for our research work, especially for GD to PD relationships and vice-versa. We limit our scope to not re-implement or prove those evidences. And hence we use those small concepts as it is highlighting the larger picture of this project.

2.2.1 Discrete Cellular Automata

Life in a grid based system is governed by set of rules, and local interactions between neighboring cells where each cell updates itself in a time step based on the neighbor it had in previous step. Classical ECA and 2D CA [4] are computational models consisting of a grid of cells, arranged in one or two dimensions with time (space-time). For example, 2D CA each cell can exist in one of a finite number of states, and the grid evolves over discrete time steps according to a set of rules. The rules dictate how the state of each cell changes based on its current state and the states of its neighboring cells. These automata exhibit emergent behaviors, where complex patterns and structures arise from the interactions of simple rules applied to individual cells and their local neighborhoods. In a 2D CA, each cell has a fixed number of neighboring cells. The evolution of the entire grid occurs simultaneously at each time step, with each cell updating its state based on the defined rules and the states of its neighbors. The dynamic evolution of the grid can result in a wide range of patterns, from stable and repetitive to chaotic and unpredictable. A classic example of a 2D CA is Conway's GoL (B3S23 [2], [42]), where cells are either alive or dead, and their states change according to a set of rules based on the number of live neighbors. Let's define a 2D CA with a grid represented by an $N \times M$ matrix, where N is the number of rows and M is the number of columns. Each cell in the grid is denoted by $C_{i,j}$, where i and j are the row and column indices, respectively. The state of a cell at time t is represented by $C_{i,j}^{(t)}$. The state space S is the set of possible states a cell can take. In a binary CA like GoL, $S = \{0, 1\}$ representing *dead* or *alive* states. The evolution of the CA is governed by a set of transition rules. Let R be the set of rules that determine the state of a cell at the next time step based on its current state and the states of its neighbors. The rules can be expressed as a function:

$$C_{i,j}^{(t+1)} = F\left(C_{i,j}^{(t)}, C_{i-1,j}^{(t)}, C_{i+1,j}^{(t)}, C_{i,j-1}^{(t)}, C_{i,j+1}^{(t)}, \ldots\right)$$

Here, F is the rule function, and the arguments are the states of the cell and its neighbors. The ellipsis (…) indicates that the function may depend on additional neighbors, depending on the specific neighborhood configuration chosen for the CA. The rules are applied simultaneously to all cells in the grid at each time step, leading to the next state of the entire automaton. This process is iterated over successive time steps, producing the temporal evolution of the 2D CA. Pictorially, Figure 2.1 represents a snapshot of the working of a 2D CA. However its a very small abstraction of a 2D CA, which is highly complex and If each cell can be in one of two states (0 or 1) and using a 3×3 Moore neighborhood (8 neighbors), the total rule space would be $2^{2^8} = 2^{256}$.

2.2.2 Continuous Cellular Automata

Beyond discrete CA, another model of exploiting 2D grid with continuous values which is also called as continuous CA. An excellent example for such CA is Lenia [22], [28]. Lenia, a digital life simulation software, employs a unique growth function as update function to simulate diverse patterns and behaviors, controlled by the parameters μ and σ. The

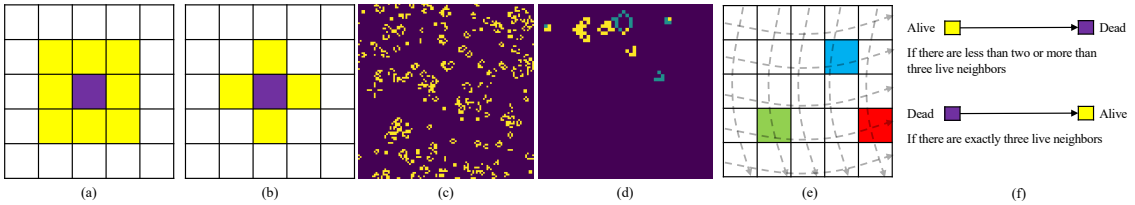


Figure 2.1: Summary of working of 2D CA. (a) Moore neighborhood (b) Von Neumann neighborhood (c) GoL in action (d) Glider Gun in action (e) Wrap around conditions in CA environments, also CA grid is initialised with three different states (also called as MNCA [34]) (f) GoL rule conditions

growth function captures the process of growth and decay in shapes and structures within the simulation. The μ parameter, representing the mean, influences the overall behavior, while σ, the standard deviation, controls the randomness in growth and decay as shown in Figure 2.2. With over 500+ discovered species, each characterized by a unique combination of μ and σ, Lenia works as a habitat for different life forms classified under biological taxonomy. It follows a hierarchical structure, organized from broader categories (e.g., class, order, family) to more specific ones (e.g., subfamily, genus, species). The update rule for Lenia is expressed as

$$A_{t+1} = [A_t + \Delta t\, G(K * A_t)]$$

where A_t is the current state at time t, Δt is the step size, G is the growth function, K is the neighborhood kernel, and $*$ denotes the convolution operation. The growth function utilizes the Gaussian function given by

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

to calculate growth values based on the neighborhood sum array. This approach enables a smooth and continuous evolution of the board state, offering precise control over growth or decay through adjustments to μ and σ. The Gaussian kernel is applied for smoothing and blurring, with the number of shells (kernels) determined by the peaks parameter. Additionally the intuition from discrete to continuous CA is provided in Figure 2.3. These elements collectively contribute to Lenia's ability to generate intricate and dynamic patterns.

Along with original Lenia, there are many other models that are introduced recently of which Evo-Lenia [43], Flow-Lenia [37], Sensorimotor-Lenia [35] and Large Scale Lenia [39]. This comprehensive exploration encompasses diverse facets of Lenia, an artificial life platform. Evo-Lenia employs evolutionary computation, utilizing Lenia kernels as genotypes and distinct fitness functions, to reveal increased complexity in the kernel's center of mass and overall measures over 500 generations. Flow-Lenia, a mass-conservative extension, addresses challenges in discovering life-like creatures by optimizing update rule parameters and enabling multi-species simulations. Sensorimotor-Lenia introduces an approach, employing curriculum learning and gradient descent over a differentiable CA, to learn self-organizing agents with sensorimotor capabilities, demonstrating adaptability and robustness. Large Scale Lenia explores open-ended evolution in AI, emphasizing implicit genetic operators and proposing factors for further open-ended evolution, such as virtual environment design and energy constraints. These studies collectively illuminate Lenia's potential as an ecosystem for emergent complexity, intrinsic evolution, and self-organizing intelligent systems.

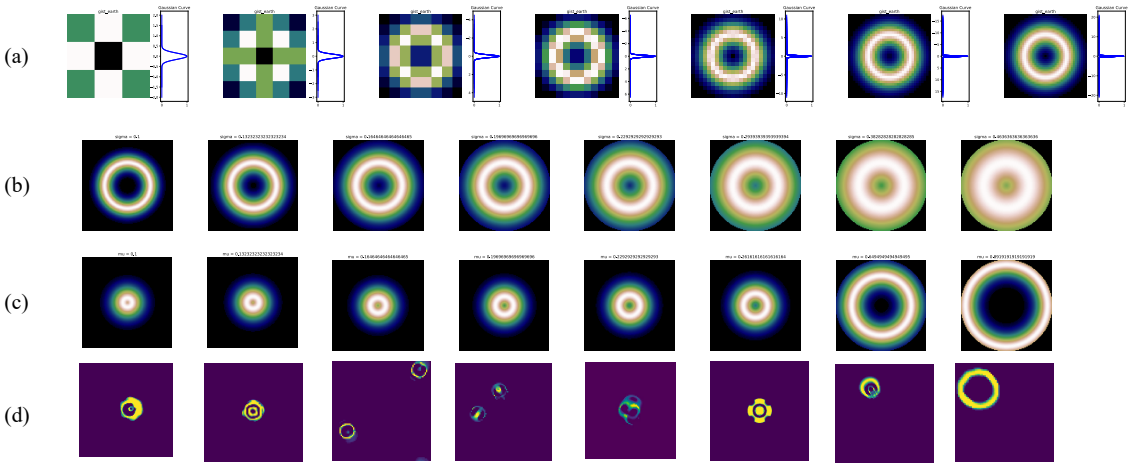


Figure 2.2: Working of Lenia in a snapshot. (a) How kernel is evolved. Gaussian kernel with a fixed value of μ and σ is used. From left to right, it is clearly visible how Gaussian smoothing works in action to produce a smoothen ring kernel. (b) Kernel with varying σ. (c) Kernel with varying μ. (d) Different life-forms evolved in Lenia-verse that follows the taxonomy in [22], [28] which also discusses Physics, Taxonomy, Ecology, and Morphology which further explains genotypic to phenotypic relations and vice versa for speciation.

2.2.3 Growing NCA

Now, We study Neural CA which is core to this research work as we consider NCA as the base framework to implement and study proposed methods of GD and PD and their corresponding tools. In this part we discuss five different frameworks that are recently proposed using NCA as baseline framework. Particularly, we study Growing-NCA [30], Self-Organizing Intelligent Matter (SIM) [33], Self-Classifying MNIST [31], Biomaker CA [44] and finally some intuitions from Lana's work on Self-Replication, Spontaneous Mutations, and Exponential Genetic Drift in NCA [45]. A large motivation from all these references has been already discussed in previous sections. We limit our next discussion to the models and frameworks proposed in the referenced research works.

Preliminary related work for our proposed research is Growing NCA. It focuses on developing a complex model for simulating the regenerative behavior based on cell-level rules. The goal is to create a CA that, starting from a single cell, produces a predefined multi-cellular pattern on a 2D grid, serving as an analogous toy model of organism development. The complete model can be studied in five parts as follows (we refer Figure 2.4). (a) Cell State Representation: Each cell state is represented as a vector of 16 real values. The first three channels represent the RGB color of the cell, with an α channel indicating whether the cell is part of the pattern ($\alpha > 0.1$). Hidden channels are open to interpretation and can represent concentrations of chemicals or other signaling mechanisms. (b) Differentiable Update Rules: Differentiable update rules are essential for leveraging loss functions and gradient-based optimization. The update rule is represented as a series of operations applied to a perception vector. The use of continuous values in cell states allows the update rule to be a differentiable function. (c,d) Perception and Update Rule: Perception is implemented using a 3x3 convolution with fixed Sobel filters to estimate partial derivatives, forming a 48-dimensional perception vector for each cell. The update rule, consisting of operations like 1x1 convolutions and ReLU nonlinearities, is applied to

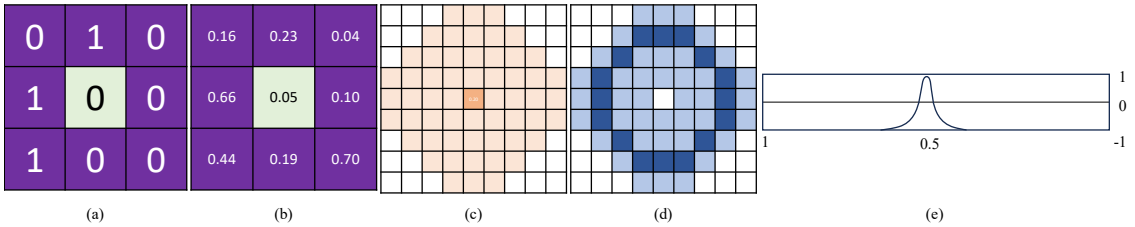


Figure 2.3: From GoL to Lenia. (a) Sum = Σ(neighborhood) and then use lookup table to perform *if-then-else* rule for current cell update. (b) Same methodology but with floating point values (for example, Smooth Life). (c) Now we are transitioning from using small 3x3 neighborhood to a significantly larger neighborhood which is circular. (d) Kernel (also called as weights) is convolved over the grid to perform weighted sum. Mathematically it should be $K * A_t$ where K and A_t are part of Lenia update rule. In simpler words, Sum = Σ(neighborhood $*$ weights). Finally, (e) Growth function (Gaussian) is applied on the resultant value with a fixed $\mu \approx 0.55$ and $\sigma \approx 0.02$. And hence Lenia update rule performs smooth update.

the perception vector. The update rule is parametrized by a network with approximately 8,000 parameters. (e) Stochastic Cell Update: Cells perform updates independently, waiting for a random time interval between updates. A random per-cell mask is applied to update vectors, setting update values to zero with a predefined probability (0.5 during training). This introduces a stochastic element, deviating from the typical synchronized update of all cells. (f) Living Cell Masking: Empty cells (those without mature neighbors) are explicitly set to zero in all channels. The α channel is used to determine the *life* of a cell, and empty cells do not participate in computations or carry hidden state. In summary, the model employs a differentiable update rule with a stochastic element, allowing for individual cell updates and a representation of organism growth. The continuous cell states and the use of gradients enable the application of gradient-based optimization techniques for learning the desired behavior of the CA. Further to that, authors experiment learning to grow, when to stop, regenerate and chirality (combination of rotations, translations, and some conformational changes).

2.2.4 Self-Organizing Intelligent Matter NCA

Another strong research motivation was from Self-Organizing Intelligent Matter (SIM). This work introduces a grid-world implementation as an instance of the proposed framework. In the proposed system, the environment is constructed with elements, each residing a neural operation, such as matrix multiplication or a sequence of operators forming a mini neural network. Elements interact through underlying rules and direct communication of neural states. The SIM lacks a built-in notion of agents and instead focuses on the environment. A neural network is placed at each point in an $m \times m$ grid, updating activations, weights, and hyperparameters over time. The networks can perform actions influencing neighboring cells, enabling operations such as replication, knowledge transfer, and programming for specialized functions. To achieve the desired system, the authors propose steps, including the use of standard recurrent neural networks (RNN) and the evolution of weights. Communication between cells is facilitated through a signal layer mechanism, and a form of physics is introduced with energy and chemical fields, influencing cell behavior. The system exhibits

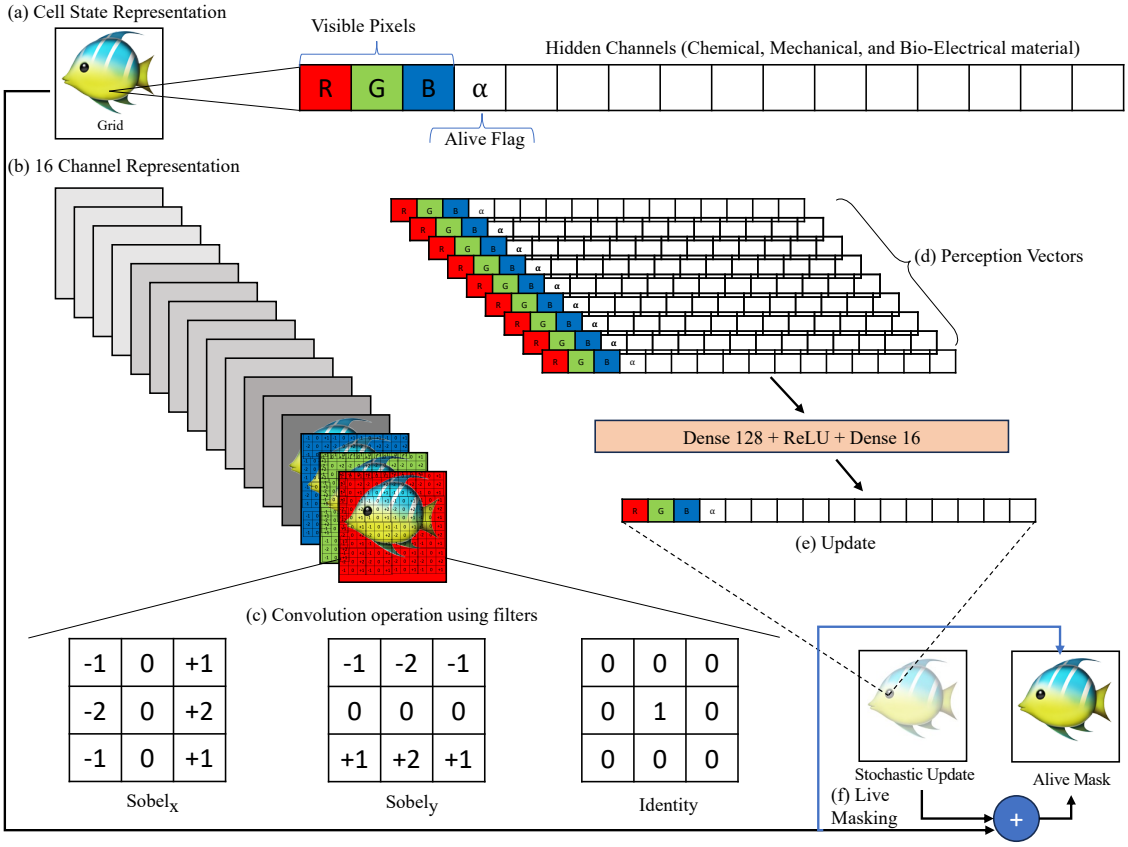


Figure 2.4: A detailed diagram of Growing NCA from [30]. Please refer text in Subsection 2.2.1 for more information.

self-replication and propagation of information through complex interactions. Overall, exploring various aspects such as grid sizes ranging from 100×100 to 400×400. Each grid location encompasses variables including energy, chemicals, enzymes, and signals conveying information about different variables (act as different layers). RNN variables, involving hidden layer activations, weights, and biases, are also integral to the system. Neural network operations involve classic recurrent updates to activations, with weights evolving during specific actions, such as copy and mutate events. The energy-chemical dynamics within a cell are elucidated, where energy is confined between 0 and a maximum value, and chemicals undergo transformations based on enzyme-mediated reactions. Cell actions, including copy, move, energy flow, chemical flow, and enzyme production, control the cell's behavior and interactions with neighbors. Additional features, such as energy protection and signals, facilitate communication and the formation of larger aggregates. The entire system, executed on a single GPU, is detailed, with the maximum system size of 400×400 ($160k$ networks) determined by GPU memory constraints. A simple demonstration of the framework is shown in Figure 2.5.

2.2.5 Self-Classifying MNIST

Self-Classifying MNIST in the same line of work of Growing NCA answers the questions that we discussed in Chapter 1 about cells knowing what to grow and where to grow.

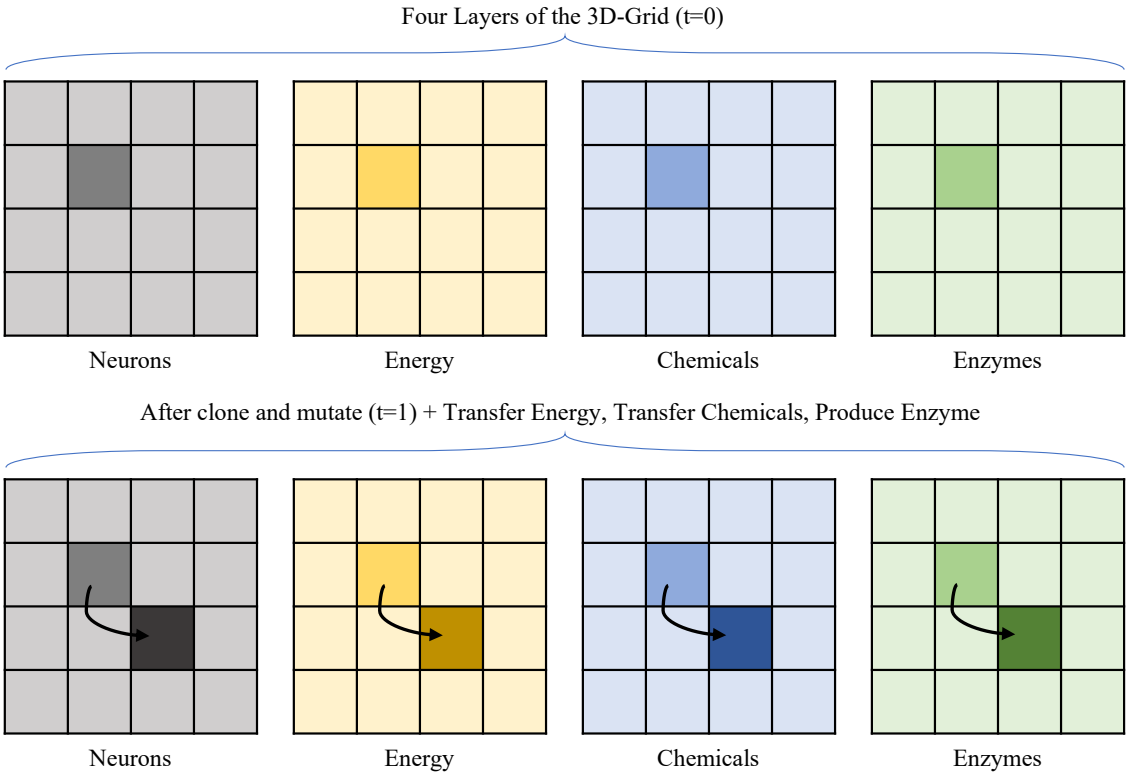


Figure 2.5: A simple illustration of the working of SIM model introduced in [33]

For example, cells interact and builds tissues, embryo and organs as they already know what exactly is needed to be developed and proportion of growth required to develop that particular organ. As already discussed, biologists agree on sharing global information among cells by local message passing. This work [31] addresses a significant question in biology: how do cells collaborate to create and regenerate complex multicellular anatomies? The model, characterized by parameter efficiency and end-to-end differentiability, introduces a novel approach to modeling the regulation of anatomical homeostasis. The study extends the application of Growing NCA to machine learning by exploring its potential in classification tasks, particularly the self-classifying MNIST task. This task involves arranging a population of agents on a grid to collectively form digits through local communication, simulating biological processes where cell collectives determine and classify their large-scale morphology. The analogy draws on developmental and regenerative biology questions related to how cell groups decide on the correctness of organ or tissue patterns. The article introduces the self-classifying MNIST task, where agents, analogous to cells, aim to correctly output the label of a digit based on their structural configuration in a manner inspired by biological remodeling structures within larger bodies. The task leverages the concept of alive and dead cells, representing whether a cell contributes to the digit's shape,

and introduces a visual representation of the labels through color mapping. The study not only explores the computational potential of Growing NCA but also draws parallels to fundamental questions in biology concerning cellular cooperation and self-awareness of anatomical structures. Implementation wise, this model is exactly similar but have a little modifications in size of hidden layers, which is 19 mutable state channels. An example snapshot of the simulator is shown in Figure 2.6.

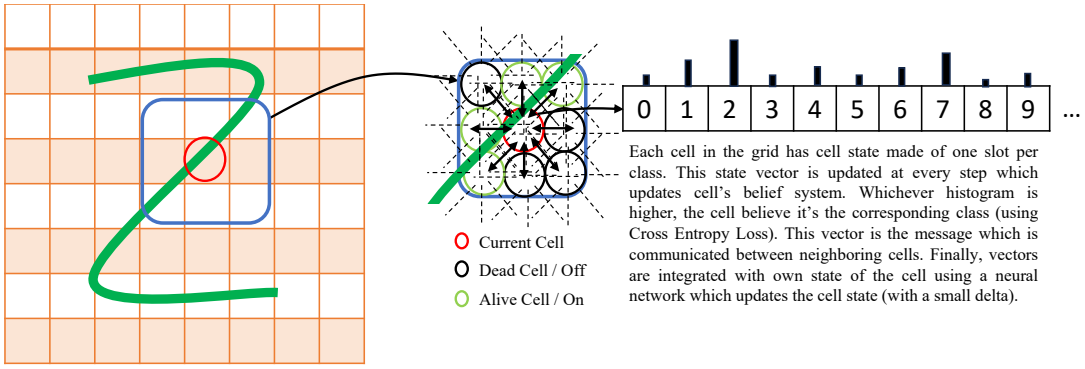


Figure 2.6: Working of Self-Classifying MNIST. It uses convolution operator which is learned only once per step as it was used in [30]. The difference with Growing NCA is two folds, primary being the algorithm, recurrent neural network with a residual convolution as operator. Secondary in previous work the latent vector comprised of RGB and Alpha values however here in [31] it has class labels (10 for MNIST) where dead and alive cells stays static.

2.2.6 Biomaker Cellular Automata

Biomaker CA, despite its over-engineered nature of the framework, suggests a comprehensive system for generating diverse environments, including the design and evolution of complex lifeforms. Environments consist of materials like Earth, Air, Agent, Immovable, and Sun, each serving a specific function. Conditions necessitate continuous nutrient harvesting from both Earth and Air for cell survival to become a plant-like organism, promoting complex operations such as specialization, reproduction, and new cell spawning. Reproduction introduces variation through mutation (with 0.20 probability of updating parameter, it mutates with Gaussian noise. Also each parameter has 20% chance of update), ensuring constant behavioral changes in Biomaker CA. Since this is highly experimental work, we refer to manual and corresponding code provided in [44].

2.2.7 Self-Replication in Neural Cellular Automata

We explore [45] version of the NCA. This work proposes even though the asynchronous NCA model was not explicitly trained for producing genetically different descendants, it was observed that progeny shows a genetic drift from ancestors. For example, (a) observing bacteria with two nuclei at the time of self-replication with transplant, (b) replication with an egg (for example fish laying an egg and then producing a fish from it), (c) upon looking at multiple generations (training steps) organism (for example fish) starts to loose stripes for some and have more than expected for others. The usefulness that author hypothesise for this particular case, biological mutations like these are difficult to get for

artificial systems. (d) One very important observation from this work, also provides a support to our research hypothesis, strong relationships between Genetic Mean Squared Error (MSE) and Phenotypic MSE (directly proportional) which for their case each of the MSE is exponential curve showing near to exponential drift in progeny's phenotype and genotype compared with their ancestors. All points (a)-(d) are illustrated in Figure 2.7.

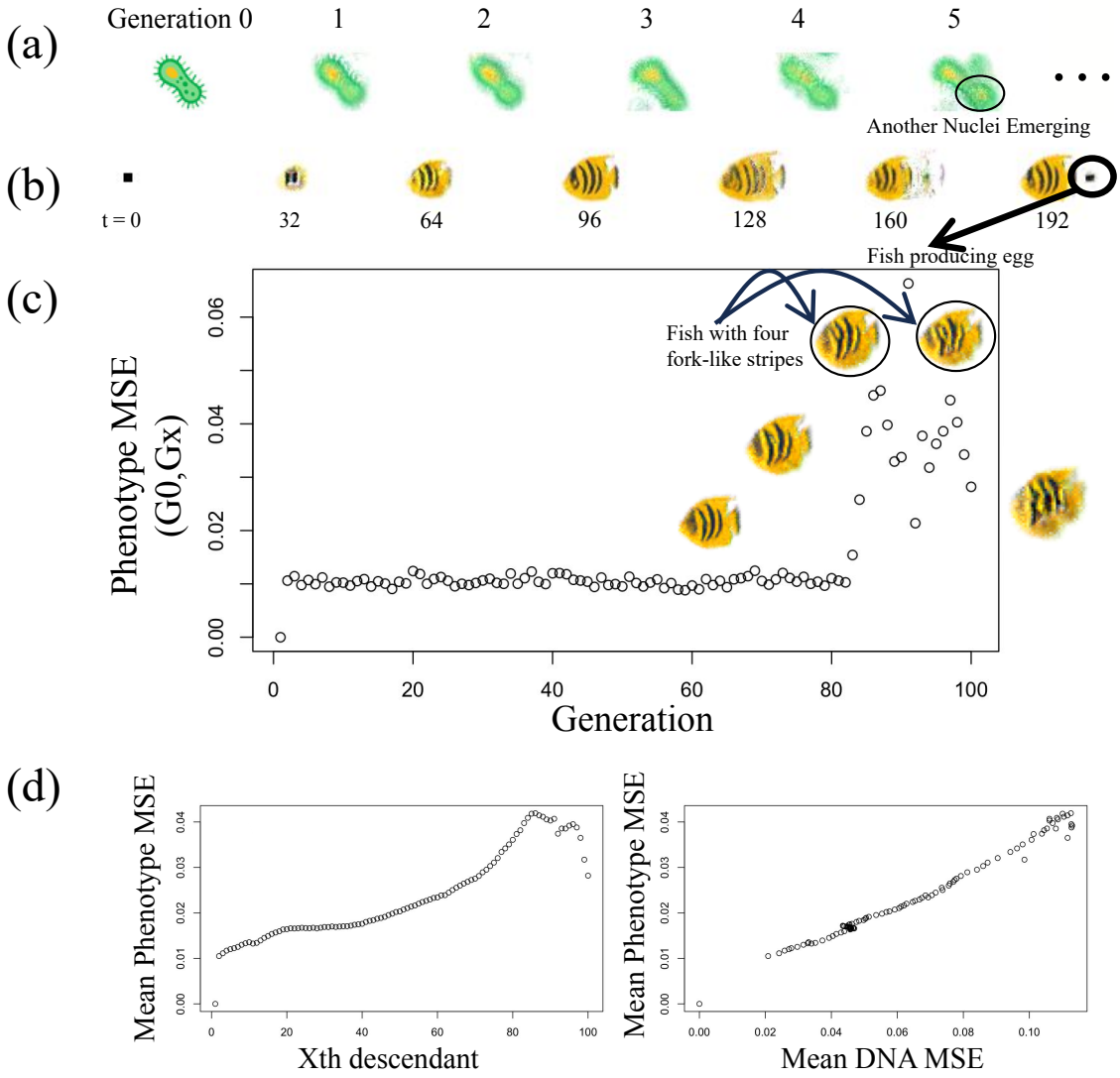


Figure 2.7: A snapshot of (a) Self-replication with mutations (b) Replication with egg, (c) Mutations over generations and (d) Genetic MSE and Phenotypic MSE shows relationship between GD and PD.

2.2.8 Diversity Measuring Tools

Coarse grained frequency analysis provides cell-level insights for speciation and genetics for CA. Not only it can help to visualise computations and capture emerging attractors in a system (for example large grid) [41], but it also helps in measuring diversity of

multi-state cellular automata (for example, Multi-neighbor CA or Neural CA). Statistical tools based on entropy, mean and variance introduced in [15] autonomously quantifies the diversity in phenotypic patterns over time which also hints about the existence, extinction and emergence of new species (cells that show identical phenotypic traits in the world). Specifically, these tools perform (1) Cell count and then plot their frequency over evolutionary steps, (2) Entropy based, (3) Gross cell variance and (4) Global variance. All of these four concepts provide measure of uniqueness of the cells with respect to either median, local neighborhood or global expected mean neighborhood. We formalise these tools and propose modifications on top of it which we further discuss in Chapter 3. Finally we also got inspirations to measure GD in our framework from [33] and [25]. Authors of SIM proposed an approach, though not robust, however, suggests to plot randomly picked gene sequences in a RGB channel plot. Similarly, McCaskill & Packard's paper suggest Genotypic Hash Coloring (GHC). Proposed GHC in [25] works on squashing the input gene sequence to a 24-bit sequence using pseudo-random hash function, and then splits the 24-bit sequence to 3 parts each for R,G and B channels. We also hypothesise that GHC should work as better tool than SIM's approach in [33] because picking n random weights from a large gene sequence would make a very small change in diversity visualisation of the genes. Finally we also study [46] for the latest developments in GD measuring tools. This work is titled as *Open-Endedness in Genelife* and focuses on innovation that arise because of long term dynamics of genetic and spatial interactions.

2.2.9 Measuring Complexity and Evolution

Measuring (or quantifying) complexity in evolutionary systems helps us understand how these systems grow and change over time. It's different from evolutionary fitness, which is all about how well individuals survive and reproduce. Researchers have been studying variety of tools to measure complexity [7], some of them are as follows.

- Kolmogorov Complexity: This measures how short the simplest description of a pattern can be, helping us see how complex the pattern really is.
- Lyapunov Exponents: These measure how quickly small changes can lead to big differences, which is useful for spotting chaotic behavior in systems.
- Entropy Measures:
  - Shannon Entropy: This quantifies how unpredictable or random a system's states are.
  - Approximate Entropy (ApEn) and Sample Entropy (SampEn): These measure the regularity and complexity of data over time.
- Fractal Dimension: This measures the complexity of patterns that repeat at different scales, often seen in nature and complex systems.
- Lempel-Ziv Complexity: This looks at how many different patterns or substrings a sequence has, which is useful for analyzing sequences from developmental systems and its growth.
- Chaitin's Omega: A specific measure that gives insight into the randomness and complexity of a system (Entropy measure).

2.2.10 Novelty Search and Quality Diversity

Novelty search [13] is a way of evolving systems that focuses on finding new and unique behaviors rather than just trying to optimize for a specific goal. This approach encourages exploration and creativity, leading to a variety of interesting solutions. Quality diversity (QD) algorithms [19] take this idea further by ensuring that the new and unique solutions are also high-quality. They aim to create a diverse range of solutions, each excelling in different ways. For example, evolution of robot behaviors [27]. Instead of evolving robots to complete a single task, like walking as fast as possible, novelty search would encourage robots to develop different kinds of movements. Some robots might evolve to walk on two legs, others might roll, and some might even hop or slither. Quality diversity algorithms would then ensure that each type of movement is not only unique but also effective in its own way. This leads to a wide variety of robot behaviors, each adapted to different environments and challenges.

2.2.11 Phenotype-Genotype Relationships

We refer three research work to provide evidences for genotype-phenotype relationships, phenotype-genotype relationships and their plasticity. As discussed in [27] they propose an experimental approach using Random Boolean Networks (RBNs) that can adjust their connections (genotype) based on environment stimuli (phenotype). In biology, the capacity of a genotype to produce different phenotypes depending on the environment in which it is located is defined as phenotypic plasticity, developmental plasticity if differences emerge during development. While mutations can contribute to diversity by altering gene networks, they are not essential for phenotypic plasticity, which is more influenced by network dynamics. However, mutations play a role in genetic accommodation, a process that occurs after the selection of a phenotypic variant with a genetic component. It can also facilitate individuals of future generations in reaching the same phenotype more efficiently.

Other works including [14] and [18] where former tries to capture emerging phenotypic traits and predict it using genomic compositions via a parameter λd. λd measures attractor length to the trajectory length for an emerging *zygote*. In developmental systems, one may argue to have a behaviour which is stable structure or state, however emerging growth or complexity is an important criteria for agents (cells or organisms) to have dynamical phenotypes to adapt environment. overall λd highlights the phenotypic changes over time. The usage of such parameter lies in the control of phenotypic growth to reach certain complexity and evolve a stable organism. Overall this work uses an evolvable genetic information (like a rule-lookup-table where depending on different neighborhoods it has corresponding state - 0, 1 or 2) which is allowed to evolve in Evo-Devo, and then this development is quantified using λd. Finally we also study [29]. It shows tradeoff between diversity between Phenotype and Genotype. If we try to make phenotype more diverse, it is speculated to lose diversity in genotype. If we try to make genotype diverse, it is speculated to lose diversity in phenotype.

2.3 Summary

We studied three different components of the related work and alternatives. First, framework which works as environment for our agents or organisms to grow. We presented three different frameworks namely discrete CA, continuous CA and Neural CA with their

corresponding flavours for different purposes. We explored NCA in much more details, as this is core to our research. Further to that, we highlighted some tools that could measure emerging growth and complexity in such environment. Specifically, we discussed about the criteria of measuring PD and GD using statistics and gene sequencing. Finally, we understood the relationship and strong resemblance between genetic code and phenotypic traits addition to phenotypic plasticity.

Chapter 3

Proposed Methodology

In this chapter we will understand the details about the concepts and algorithms we used for the development of the proposed framework (which we have been calling it as *substrate*) and corresponding tools to analyse Genotypic Diversity (GD) and Phenotypic Diversity (PD). Firstly, we formulate the environmental framework (formally we call it as Neural CA) that we propose with this research. This framework is a grid based system where the agents evolve to represent cells in the grid (recall *cells* vs *agents* in 1.4, we will often use these terminologies in the rest of the chapter) using self-replication and mutation. Secondly, diversity measurement tools are proposed that could work within our proposed 2D grid based framework. These tools work as evaluation criteria for the evolved grid on the verge of speciation and genealogies. We discuss more details about the theoretical analysis of these tools in later chapters. However it is crucial to note at this stage to understand the importance of these algorithms, measuring the evolved system provides insights that how many species survive, die, extinct, rebirth and adapts in the environment therefore becoming self-aware or *intelligent*. Overall, quantifying different genealogies and species over time using GD and PD tools help us to capture the emerging intelligence within such system where agents try to self-organise and self-maintain (autopoiesis) to survive for longer generations in the environment.

3.1 Overview

Our 2D grid based substrate is built on top of the ideas of Cellular Automata (CA). Addition to that, a slit of neural operations are assigned with each cell. Therefor, we call it as, our Neural CA (NCA) resembles an environment where agents can interact with each other (sense neighborhood) and result different dynamics over time. In other words, our NCA is made of different nature-like compositions (for example, chemical, enzyme, energy or any other activations) along with a neural head (agent) for each cell. And we let it grow with help of self-replication with mutation which is explained in details in the next section. Further, this growth is quantified (measured) using different tools that we propose in later sections as well. We quantify this growth for cells and agents separately using entropy and variance over time for *cell* diversity quantification where as gene sequencing methods for *agent* diversity quantification. Simply put, the proposed methodology has a 2D NCA made of two channels and a layer of neural agents, where one channel of the NCA is Alpha channel which determines the livelihood of the cells and a hidden composite channel can be analogous to any of the energy, matter or enzymes. We then let agents evolve over large

time steps. Finally, we perform a coarse grained analysis to determine various PD and GD level results from it which we discuss in later chapters.

3.2 Neural Cellular Automata as Framework

Proposed NCA model consist of two channels, where primary channel which we call as α channel is dedicated for alive agents where as the second channel persist hidden information (which can be in the form if energy, matter or enzyme) as latent variables for each agent. It is worth noting that both the channels are stacked together and hence participate as depth-wise composition of an agent. In simpler words, each agent (or cell) that is alive in the α channel posses hidden information (chemistry) in the second channel which is intrinsic to an agent. Further, each cell of the grid is assigned with an agent that has a neural operator (artificial neural network - ANN) that senses the neighborhood. We discuss architecture of the neural network and corresponding hyper-parameters in Chapter 4. Because environment may have some entropy already in the beginning, we seed few agents randomly (uniformly, as show in Figure 3.2) in the environment with some initial probability along with the stacked cells, called *founders*. This way, we initialise all such cells to unity and corresponding latent variables have some random storage of matter or chemistry (which can also be unity) in the other channel. Rest all cells and agents stays dead which are not initialised, hence not contributing by any means. While these agents (neural network weights) are initialised randomly, corresponding cells are meticulously initialised with unity (or any value greater than α) and therefor always alive in the environment. This means, a cell is said to be alive only when it posses the agency as ANN and some cell value in NCA. Because of inheritance, it may happen that agency of a cell has become non zero but it is at the next generation when the inherited cell will become alive. The methodology so far is summarised with Figure 3.1. The Figure shows two grids (or channels). The meta data at the top of the figure shows generation number, alive cells at the beginning and their corresponding neural agencies. It also shows variance of all cells in the grid and all agencies of the grid. We discuss all these in later sections in great details.

The next important design is for simulation. To run the proposed framework, such that it updates the states and hence global behaviour of the CA is changed for time steps, our approach to update CA follows *clone* and *mutate*. Governed by the principles of neural inheritance and perturbation, *update* iteratively processes each cell's neighborhood, considering the influence of neighboring cells with alpha values surpassing a specified threshold. In instances where neural inheritance occurs, an agent is replaced by that of a randomly chosen neighboring alive cell. To introduce variability, neural network weights also undergo perturbations. The *update* logic effectively works between cells and agents, by local interactions within the simulated environment. Following points enlist stepwise methods and details the *update* function of the proposed NCA:

1. To populate an initially empty vector with the neighborhood of the current cell, the methodology employs nested loops iterating over a 3×3 neighborhood surrounding the current cell in the 2D grid. For each pair of dx and dy representing the relative coordinates of neighboring cells, new indices ni and nj are calculated using modulo operations to ensure periodic boundary conditions. The method updates the vector, by assigning values from the corresponding depthwise positions. The resulting neighborhood vector comprehensively captures the values of neighboring cells across

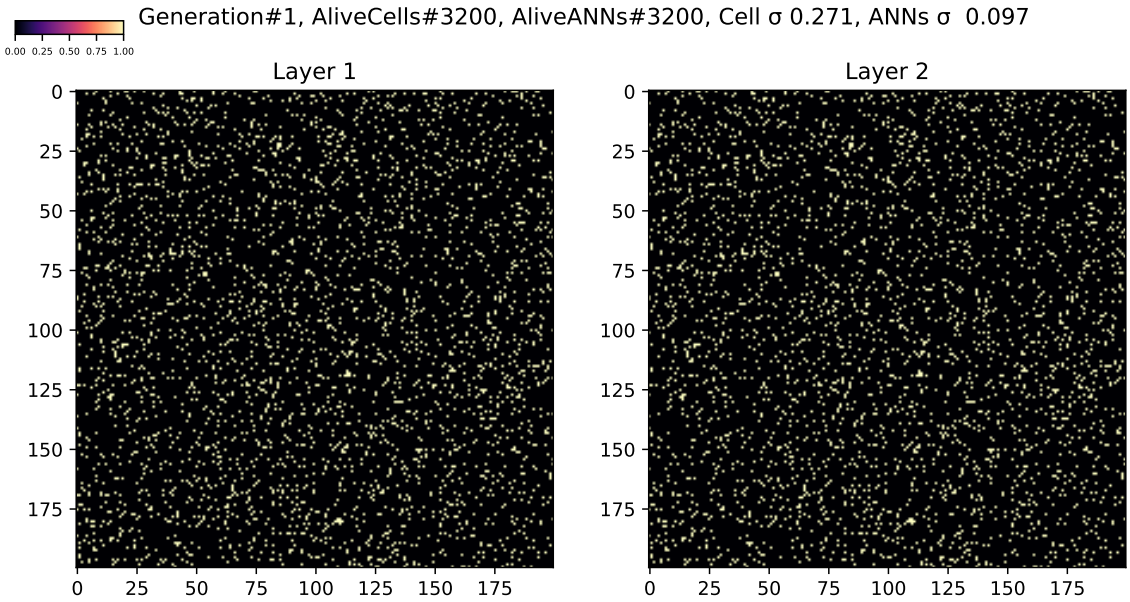


Figure 3.1: Snapshot of the proposed and initialised NCA framework. Left side grid shows α channel and right side grid shows corresponding matter (chemistry) of the alive cells. Text at top provides meta-data information of the NCA at every step which we will discuss more in implementation details (Chapter 4).

all channels, providing a representation of the local environment around the current cell.

$$N_{i,j} = \begin{bmatrix} \left(C_{i-1,j-1}^{(t,1)}, \ldots, C_{i-1,j-1}^{(t,d)}\right), & \left(C_{i,j-1}^{(t,1)}, \ldots, C_{i,j-1}^{(t,d)}\right), & \left(C_{i+1,j-1}^{(t,1)}, \ldots, C_{i+1,j-1}^{(t,d)}\right) \\ \left(C_{i-1,j}^{(t,1)}, \ldots, C_{i-1,j}^{(t,d)}\right), & \left(C_{i,j}^{(t,1)}, \ldots, C_{i,j}^{(t,d)}\right), & \left(C_{i+1,j}^{(t,1)}, \ldots, C_{i+1,j}^{(t,d)}\right) \\ \left(C_{i-1,j+1}^{(t,1)}, \ldots, C_{i-1,j+1}^{(t,d)}\right), & \left(C_{i,j+1}^{(t,1)}, \ldots, C_{i,j+1}^{(t,d)}\right), & \left(C_{i+1,j+1}^{(t,1)}, \ldots, C_{i+1,j+1}^{(t,d)}\right) \end{bmatrix}$$

The above $N_{i,j}$ represents the values of neighboring cells surrounding the cell at position (i, j) and time t in a 3×3 grid with d channels. Specifically, we use Moore neighborhood and wrap around condition as mentioned in Figure 2.1 (a) and (e) respectively.

2. If any of the $N_{i,j}$ cell has greater value than α, process $N_{i,j}$ with agency of the current cell. In simpler words, these alive (may also have some non alive cells) neighborhoods are processed using the neural agents assigned with the current cell (the cell at position (i, j) and time t depthwise). Else, $N_{i,j}$ cell has no greater value than α, the agent dies making its cell value to null.

3. Further, with some probability we reach towards the methodology to *clone* and *mutate* (we use uniform distribution for mutation on contrary to normal because of non-biasness of the perturbed value, as it is in Figure 3.2). We first check for at least one neighbor that is alive, then with some probability we let it enter the inheritance method. At this stage we store the positions of the agents that are alive. Secondly, we pick one random neighbor out of all living neighborhoods. With these steps, finally, we clone and mutate (uniform perturbation) the chosen agent to the

agency of the current cell. An illustration of all these steps is presented in Figures 3.3 and 3.6.

4. We also allow a life budget mechanism for each cell. In this method, we store a counter for each cell of the grid. The counter increments as generation goes by. If specified reachable counter hits, the agent dies (and hence the cell). This way we are cleaning all dead cells and agents on the basis of budget counter and threshold. However, we also ensure at the time of cleaning all dead cells such that the *just inherited* cells do not get wiped, just because they don't have a living α value.

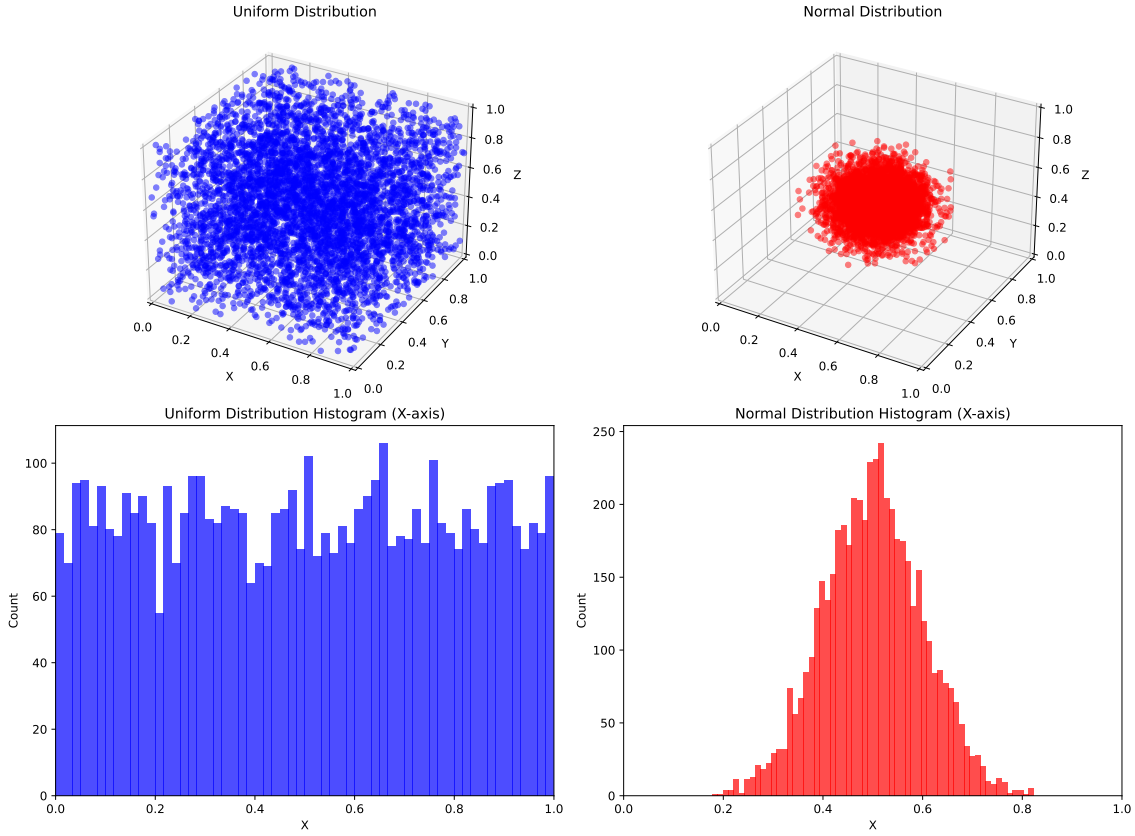


Figure 3.2: Shows a random run of 5000 particles to distribute in a 3D space *uniformly* (Left) vs *normally* (Right).

From the Figure 3.3 we can also deduce *fate of a living cell* and *fate of a dead cell*. Following points provide a detailed pathways for *fate of a living cell* also shown in Figure 3.4. While scanning we reached to a living cell, then ("→" resembles next step)

1. Neighborhood (Nh.) Alive == True →Calculate output using Nh. →Inheritance probability passed →store current index →pick a random living cell →if current index ANN (agent) non empty, which is yes →Deepcopy same ANN material.

2. Nh. Alive == True →Calculate output using Nh. →Inheritance probability NOT passed →cell value is the output and ANN remains unchanged.

3. It dies only when budget exhausts or the agency results a lower α.

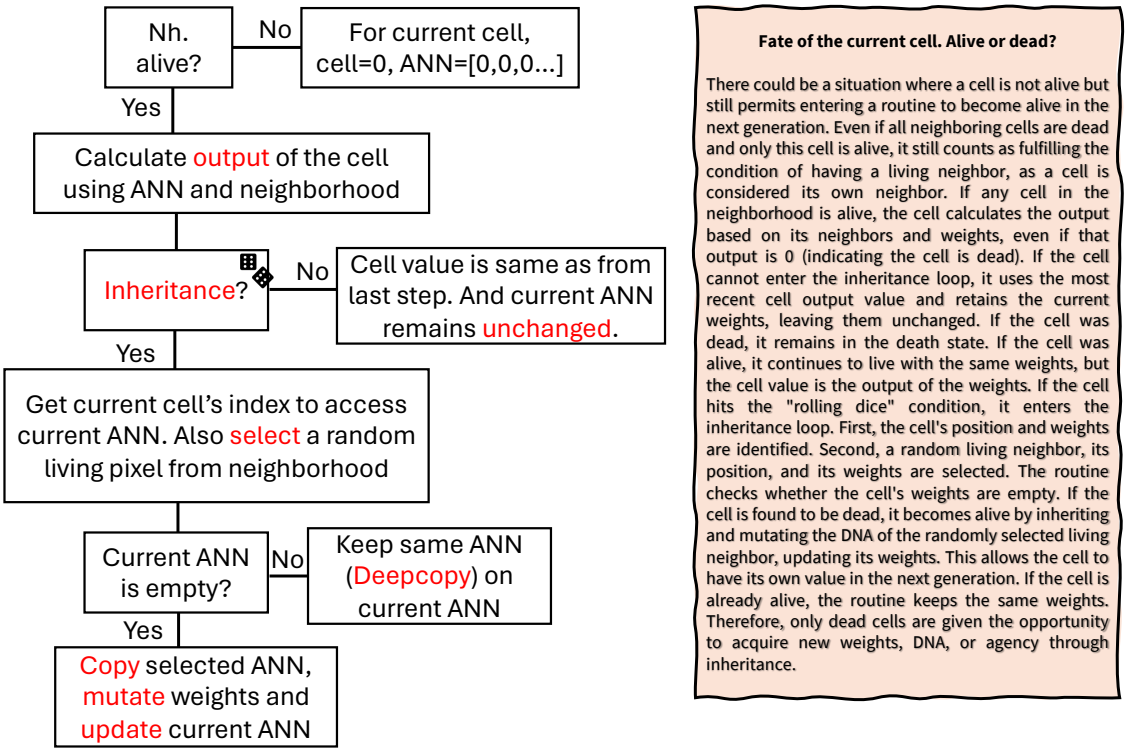


Figure 3.3: Fate of a living and dead cell! This flow diagram illustrates our proposed methodology of NCA framework how an agent can become alive or continue to dead. The comment box provides the complete lifecycle of a cell.

Following points provide a detailed pathways for *fate of a dead cell* also shown in Figure 3.5. While scanning we reached to a dead cell, then

1. Nh. Alive == True →Calculate output using Nh. →Inheritance probability passed →store current index →pick a random living cell →if current index ANN non empty, which is false (because current ANN is actually empty) →copy and mutate weights from random living cell's ANN.

2. Nh. Alive == True →Calculate output using Nh. →Inheritance probability NOT passed →output is 0 (used from previous step) and ANN remains unchanged which is dead in this case.

3. Nh. Alive == False →output is 0 and ANN weights are zeroed.

4. It takes birth (given chance to take birth) only when Nh. has living cell(s).

These steps are performed for each cell and their corresponding agent in the grid. To summarise the update methodology, it is based on core NCA (or CA) principles, where each cell state is determined by an agent that processes the neighborhood information. To update the NCA, we iterate each cell, considering its neighborhood, and calculates the output of the associated agent. If the neighborhood contains cells with alpha values above a certain threshold α, the agent output is calculated, and there is a chance for agent inheritance. Inheritance involves selecting a random live cell from the neighborhood, copying its neural network, and introducing perturbations to the weights. If no live pixels

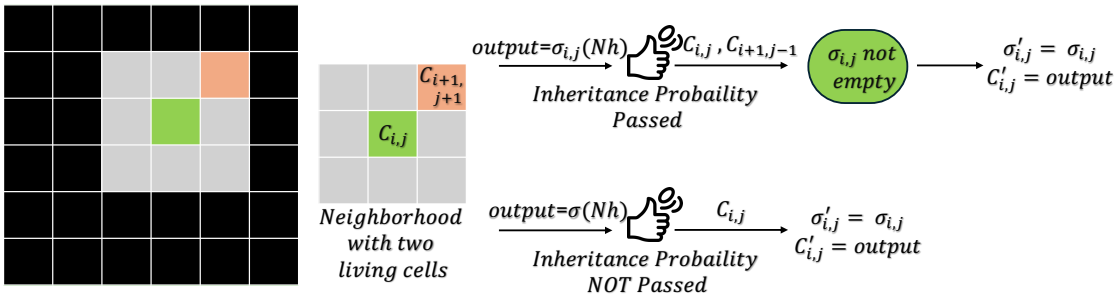


Figure 3.4: Fate of a living cell.

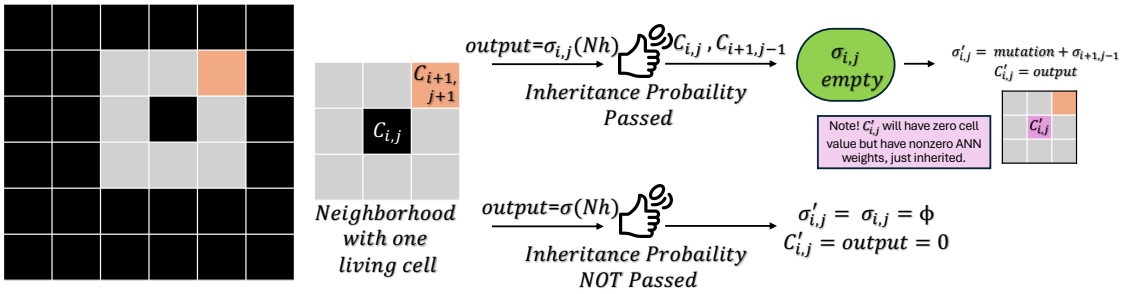


Figure 3.5: Fate of a dead cell.

are present in the neighborhood, the neural network weights are initialized to zero. The final CA grid is then updated based on the agent outputs, and the alpha values are thresholded to maintain stability.

The post-processing method starts once the scan of all agents is completed as part of the update run. Once the output of all agents are ready, we perform following steps chronologically.

1. We normalise all cell values including all layers to democratise the NCA cells. It is done using an activation function. We discuss about the implementation of these activation functions in next chapter.

2. Perform α check on the first channel on the basis of which an agent dies. A cell is called dead when its cell value, all hidden information and agency are nulled.

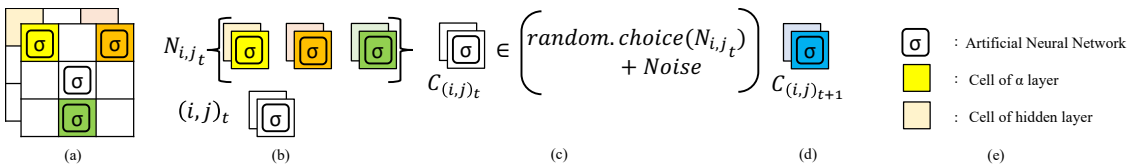


Figure 3.6: A simple illustration of the proposed NCA inheritance framework. (a) 3×3 grid of Moore neighbors of the current cell, showing cells (and agents) that are active while all white cells are dead. (b) List of all living neighbors at time t for the current cell at position (i, j). (c) Current cell undergoes inheritance and mutation using a Noise vector. (d) Cell at time $t + 1$ shows the resulting value from the agent (σ). (e) Representation for the rest of the diagram.

3. To make cells die, we also put an external condition of life budget. This budget ensures no cells grow beyond a specified threshold. Therefor we scan each cell, check its livelihood, if greater than alpha, increment budget counter with one, else make budget counter to zero for the respective cell. It is an optional condition, as we can always set infinite budget

4. We ensure while checking the α and making cells die, that just inherited cells do not die by storing their indices. Because a cell may have become dead because of the α condition, but since some cells are just inherited, they may also fall in the same trap.

The idea is to build an ecosystem where agents thrive, in which self-replication from alive neighbors is the only way to live longer for the current cell or agent. In other words, *if my neighbor(s) is/are alive, then I have a higher chance to live longer*. This neighborhood interaction is a fundamental concept in CA, where the behavior of a cell depends on the states of its surrounding cells. During neural network inheritance, we introduce perturbations to the weights of the inherited ones, which allows for the exploration of different behaviors based on the influence of neighbors, contributing to the system's adaptability and agent's longevity. In simpler words, in order to live longer, agents undergo self-replication and mutation which helps to adapt the environment. Conversely, The thresholding conceptually represents a stability control mechanism, preventing the system from diverging into chaotic or unstable states. Following are some key ideas that can be derived from our proposed framework methodology: (1) Agents with more number of living neighbors have more chance to live longer. (2) After continuous inheritance with mutation it is very likely that a different ancestor-descendant relationships emerge which may lead to a completely different specie of a descendant than its ancestor. (3) It is expected to see phenotypical changes (different cells) based only on genetical changes (different agents) over time. Also, it is worth noting, we are not making any cell-level changes explicitly using mutations, however the control is in the hands of agents (neural networks). (4) It is very less probable that a cell dies in growth phase because if a cell becomes alive, it is participating in the growth of potential neighbors which can make more cells alive. And hence the cells, can either be replaced with new specie or they continue to live. For now, through logical reasoning, the growth phase signifies a favorable environment where the presence of a living cell initiates a positive feedback loop, contributing to the survival and potential reproduction (using self-replication) of neighboring cells. We devise more empirical tools in further sections. (5) A single living cell has potential to self-replicate and thrive in the environment to produce its descendants and emergence of new species over time. (6) Over a significant generations, we estimate that new species would emerge, old ones die (or consuming the complete life-budget), some survive or adapts the environment. However, the ones that survives and adapts the environment, and still grows after their decay phase can be classified as *pseudo-intelligent* as they have *semi-learned* the dynamics of the system to stay alive for longer generations. (7) Developmental cells of growth phase can have similar phenotypical traits with different genealogies and genetic material of corresponding agents.

3.3 Phenotypic Diversity Tools

Continuing from the point number 4 in the last section (from last paragraph), let us analyse it a little more. In the growth phase, a cell becoming alive indicates a conducive

environment for growth. The logic is that a living cell contributes to the potential for more living cells by influencing its neighbors positively and hence contributes to the survival and potential reproduction of neighboring cells, leading to the overall growth of a local cluster of living cells. In order to prove it, first option could be, keeping track of the number of cells that become alive during the growth phase and monitor how often these cells survive or die in subsequent steps. This can be done by collecting statistics over multiple time steps. Secondly, analyzing the environment output over time to identify instances where cells that became alive during the growth phase continue to persist and contribute to the growth of neighboring cells. Thirdly, calculating the probability of survival for cells that become alive during the growth phase. This involves quantifying the ratio of living cells that persist over the total number of cells that became alive during this phase. This method further measures the growth rate of living cells over time. This can provide insights into how the positive feedback loop contributes to the expansion of connected components (attractors). Finally, some more statistical tests (using mean and variance) can be done to validate the logical and empirical outcomes, to determine if the observed survival rate during the growth phase is statistically different from what would be expected.

We use above mentioned ideas as first priority, besides the already studied motivation and related work that provides the evidences to measure PD (we refer Chapters 1 and 2). Following are the four major tools (in each subsection) that we propose in the line of previous work [15] to measure the PD and physical dynamics in the proposed framework[1].

3.3.1 Cellular Type Frequency Plot (CTFP)

This metric works by counting the number of cells in each state or 'color' at each time step. Each colour represents a different state depending on the precision of the cell values which are float values. And hence according to a precision, the cut-off is decided for the number of states. It's very straight-forward to think the rapid increase in the number of states with higher precision values. The concept revolves around monitoring variations in counts over time for distinct states, offering insights into the emergence of species. Essentially, the idea is genetic drifts occur over time in agents, leading to observable changes in counts by affecting cellular physical traits. These fluctuations in population sizes serve as indicators of alterations in the traits or behaviors of entities within the system, often referred to as genetic changes. In simpler terms, fluctuations in the population size of a state are linked to phenotypic diversity, by influence of genetic changes in species. The underlying assumption is that the shift in state (or physical trait) is directly associated with genetic variations and the process of *speciation*. Algorithm 1 calculates the frequency count of each cellular state in a 2D CA represented by grid G. The algorithm initializes an empty dictionary, *Dict*, to store the counts of each unique state. It then iterates through each cell in the CA, where W is the width and H is the height, extracting the state $C_{i,j}$ at position (i, j). For each state, the algorithm updates its count in the dictionary by incrementing the existing count or initializing it to 1 if the state is encountered for the first time. This process results in a comprehensive dictionary containing the frequency counts of all distinct cellular states in the grid, providing valuable information for analyzing population dynamics and observing PD in the system over time.

[1]We provide all mathematical examples for the corresponding Phenotypic Diversity tools in Appendix A

Algorithm 1 Cellular Type Frequency Plot (CTFP)

```
1: procedure FrequencyCount(G)
2:     Dict ← {}                          ▷ Initialize an empty dictionary for state-counts
3:     for i ← 1 to W do                                                    ▷ Width W
4:         for j ← 1 to H do                                                ▷ Height H
5:             C_{i,j} ← G[i][j]                          ▷ Cell state at position (i, j) in grid G
6:             Dict[C_{i,j}] ← Dict.get(C_{i,j}, 0) + 1          ▷ Update count for the cell state
7:         end for
8:     end for
9: end procedure
```

3.3.2 Global Entropy Plot (GEP)

GEP is a metric based on entropy, aiming to quantify the diversity of states in the CA. The methodology involves four steps: calculating spatial frequency for each state, determining the contribution of each state to overall entropy, summing these contributions, and finally, obtaining the global entropy at a given time.

1. Calculate Spatial Frequency: For each selected state, compute its spatial frequency, representing the proportion of cells in that state compared to the total number of cells in the model. For each state s_i in the set of selected states:

$$\rho(s_i) = \frac{N_i}{T_c}$$

   where $\rho(s_i)$ is the spatial frequency of state s_i, N_i is the count of cells in state s_i, and T_c is the total count of cells in the model for chosen states.

2. Contribution to Entropy: Determine the contribution of each state to overall entropy using the formula $Contribution(s_i) = -\rho(s_i) \cdot \log(\rho(s_i))$, where $\rho(s_i)$ is the spatial frequency of the state.

3. Summation of Contributions, λ: Sum the contributions from all selected states (or entities) to obtain the total contribution, capturing the collective impact on entropy. Sum the contributions from all selected states (λ):

$$\lambda = \sum_{i=0}^{|S|+1} -\rho(s_i) \cdot \log(\rho(s_i))$$

   where $|S|$ is the total number of selected states (remember selected states are not equal to total number of states) and addition of 1 is for no recognisable state.

4. Global Entropy, $H(t)$: Calculate the global entropy at the given time by normalizing the total contribution with the logarithm of the total number of possible states ($|\Sigma|$), indicating the diversity of states in the CA. Calculate the global entropy at the given time:

$$H(t) = \frac{1}{\log(|\Sigma|)} \cdot \sum_{i=0}^{|S|+1} -\rho(s_i) \cdot \log(\rho(s_i))$$

where $H(t)$ is the global entropy at time t and $|\Sigma|$ is the total number of possible states. $H(t)$ provides diversity, with higher values indicating greater diversity (randomness) and lower values indicating more uniformity (predictable) in the distribution of states in the CA.

It is important to note that, (1) Total number of selected states ($|S|$) is different than the total number of states that may exist in the system ($|\Sigma|$). (2) We calculate λ only based on the cells that are related to the states that we select. And we do not bother about rest of the cells that are not part of the states we picked. (3) If we choose only few states (out of total states), the goal is to assess the diversity and uncertainty within the selected states and their distribution, and that particular analysis is limited to those specific states and their corresponding cells. (4) Smaller entropy values suggest that the system is less diverse, with a more predictable or uniform distribution of states. (5) Larger entropy values indicate that the system is more diverse, with a less predictable and more random distribution of states. (6) Can be calculated at every time step, as well as in intervals. For example, entropy calculated at intervals may represent diversity in species after specific events or after few time steps.

3.3.3 Gross Cell Variance Plot (GCVP)

This metric utilizes variance over the grid to quantify diversity in the CA. It calculates the variance compared to the median state (S_{med}), representing the state occurring with a frequency at the middle of the distribution of cell states. Algorithm 2 involves one main step which is explained here. First, it determines the median cell state (S_{med}) that occurs with a frequency at the middle of the distribution of cell states within the CA model. Then, for each cell ($C_{i,j}$) in the CA, it calculates the gross cell variance (σ_{gross}) as follows:

1. Perform Kronecker δ function. It is defined as $\delta(S(C_{i,j}), S_{\text{med}}) = \begin{cases} 1 & S(C_{i,j}) = S_{\text{med}} \\ 0 & \text{else} \end{cases}$
2. Keep on adding the values to $\sigma^{sum}_{\text{gross}}$, however each value is normalised with total number of cells (T_c), for the states present in the CA.

Finally $\sigma^{t}_{\text{gross}}$ is returned, assigned with the value $\sqrt{\sigma^{sum}_{\text{gross}}}$ for time step t. This measure quantifies the variance in cell states from the median state, providing an overall assessment of diversity in the CA model. A higher σ_{gross} value indicates greater diversity, while a lower value indicates less diversity. For now, it can be derived from this formulation, for a larger PD and for the cells to perform *life-like* activities in the system (die-extinct-emerge-grow), we should welcome oscillating behaviour in GCVP. It can also be imagined that for beginning when NCA is in growth phase, it should try to reach maximum value to increase variation compared to the dead cells. However once all the cells are populated with living species, it should fall significantly. The real usage of σ_{gross} comes to play once it stabilises towards a certain range of levels, but oscillates within.

3.3.4 Cell Local Organisation Global Variance (CLOGV)

Measure the local organization of cell states by evaluating how different a particular local neighborhood is from the expected cell state. The expected cell state can be the average cell state in normal or average conditions. It is explained using Algorithm 3 (in

Algorithm 2 Gross Cell Variance Plot (GCVP)

procedure GROSSCELLVARIANCE(G) ▷ Grid G is input to the function
  $S_{\text{med}} \leftarrow FindMedian(\text{G})$ ▷ Find median state of the CA
  $\sigma^{sum}_{\text{gross}} \leftarrow 0$
  for $i \leftarrow 1$ to W **do** ▷ Iterate each cell in width (W) of the grid
    for $j \leftarrow 1$ to H **do** ▷ Iterate each cell in height (H) of the grid
      $\sigma_{\text{gross}}(C_{i,j}) \leftarrow \frac{1}{T_c}\left(1 - \delta(S(C_{i,j}), S_{\text{med}})\right)$ ▷ Explained in 3.3.3
      $\sigma^{sum}_{\text{gross}} \leftarrow \sigma^{sum}_{\text{gross}} + \sigma_{\text{gross}}(C_{i,j})$
    end for
  end for
  $\sigma^{t}_{\text{gross}} \leftarrow \sqrt{\sigma^{sum}_{\text{gross}}}$ ▷ Gross cell variance returned for time step t
end procedure
procedure FINDMEDIAN(G)
  $G_f \leftarrow []$ ▷ Initialise empty list for flattened grid
  for $i \leftarrow 1$ to W **do**
    for $j \leftarrow 1$ to H **do**
      Append $S_{i,j}$ to G_f ▷ Put each cell in a list to flatten it
    end for
  end for
  $G_f \leftarrow \text{InsertionSort}(G_f)$ ▷ Apply insertion sort to the flattened list
  $mid_i \leftarrow \lfloor \frac{\text{len}(G_f)}{2} \rfloor + 1$ ▷ Index of the middle element
  $S_{\text{med}} \leftarrow G_f[mid_i]$ ▷ The middle element
  return S_{med}
end procedure
function INSERTIONSORT(arr) ▷ Standard Insertion sort algorithm for reference
  for $i \leftarrow 1$ to len(arr) **do**
    key $\leftarrow$ arr[i]
    $j \leftarrow i - 1$
    while $j \geq 0$ and arr[j] $>$ key **do**
      arr[$j + 1$] $\leftarrow$ arr[j]
      $j \leftarrow j - 1$
    end while
    arr[$j + 1$] $\leftarrow$ key
  end for
  return arr
end function

the continuation of previous sections). This expected distribution is determined by the normalized frequency of cell states, providing insights into the distinctiveness of specific neighborhoods in terms of cell state frequencies and local structure. The importance of this metric lies into measurement of the local diversity and variation in cell states within a specific neighborhood using μ_{loc} and σ_{loc}. This local neighborhood can be tweaked with a parameter. For example, 3×3 grid is a neighbor of 9 cells, that contributes to toal count of cells in local neighborhood, can be seen in Step 2 of 3. The metric aims to assess how different a particular neighborhood is from what is considered a expected neighborhood. Overall, it helps to gain insights into how specific neighborhoods within the CA, exhibit distinctiveness in terms of cell state frequencies and local structure using μ_{glob} and σ_{glob}.

Algorithm 3 Cell Local Organisation Global Variance (CLOGV)

procedure GLOBALVARIANCE(G) ▷ Grid G
  Calculate Normalized Frequency of Cell States ▷ Step 1
  for each cell state S_i in G **do**
    $\rho_{\text{norm}}(S_i) \leftarrow \frac{\rho(S_i)-\rho(S_{\min})}{\rho(S_{\max})-\rho(S_{\min})}$ ▷ Normalized Frequency
  end for
  Calculate Local Organisation in 3×3neighborhood(ν_M) ▷ Step 2
  for each cell $C_{i,j}$ in G **do**
    $\mu_{\text{loc}}(C_{i,j}) \leftarrow \frac{1}{9} \sum_{c \in \nu_M(C_{i,j})} \rho_{\text{norm}}(S_c)$ ▷ Local Mean
    $\sigma_{\text{loc}}(C_{i,j}) \leftarrow \left(\frac{1}{9} \sum_{c \in \nu_M(C_{i,j})} \left(\rho_{\text{norm}}(S_c) - \mu_{\text{loc}}(C_{i,j})\right)^2 \right)^{1/2}$ ▷ Local Variance
  end for
  Calculate Global Mean and Variance ▷ Step 3
  $\mu_{\text{glob}} \leftarrow \frac{1}{T_c} \sum_{C_{i,j} \in \text{G}} \sigma_{\text{loc}}(C_{i,j})$ ▷ Global Mean
  $\sigma_{\text{glob}} \leftarrow \left(\frac{1}{T_c} \sum_{C_{i,j} \in \text{G}} \left(\sigma_{\text{loc}}(C_{i,j}) - \mu_{\text{glob}}\right)^2 \right)^{1/2}$ ▷ Global Variance
end procedure

The value of σ_{glob} serves as an indicator of the overall diversity and variation in cell states within the entire CA. A higher σ_{glob} suggests that there is significant variability in local organizations among different neighborhoods within the CA, indicating a diverse distribution of cell states. This can be attributed to neighborhoods exhibiting distinctiveness in terms of cell state frequencies and local structure. On the other hand, a smaller σ_{glob} implies a more uniform or consistent distribution of cell states across the entire grid. In such cases, neighborhoods tend to share similar characteristics, and the local organizations exhibit less variation. Therefore, the σ_{glob} value in the CLOGV plot provides a quantitative measure of the global diversity in cell states, with higher values signifying greater heterogeneity (and hence diverse species) and lower values indicating more uniformity (predictable behaviour) in the distribution of local organizations within the CA model.

3.4 Genotypic Diversity Tools

In this section, we will analyse three tools to measure and visualise Genotypic Diversity (GD) in the evolved system. We provided a brief overview and some of the related work

in chapters 1 and 2 for GD tools respectively, also elaborated in [25], [46]. Following subsections describe detailed methodology for each of the proposed GD tools[2].

3.4.1 Genotypic Hash Coloring (GHC)

This tool provides a visual representation of the evolution of agents' (neural networks) genes (weights) over time in a simulated scenario. As NCA grows, agents undergo inheritance and mutations, if the neighbor(s) are alive. As we have already seen in 2.2.1 how inheritance works with mutation, it allows to distinguish between ancestor agent (the one that is alive in neighborhood and selected for cloning) and the descendant agent (the one that inherits or receives the copy of genes from the ancestor). In order to capture a meaningful ancestor-descendant relationships, we propose GHC methodology. The method is very simple, as it reads, and explained below. We take the genes of the agents at every time step of evolution or genetic data as the starting point for this tool.

1. For each time step, the genes (weights of neural network) are compressed (hashed) using a function simulating gene sequencing, mapping them to a 24-bit sequence. Compression, in metaphorical sense, is analogous to few of the biological processes. For example, during evolution, genetic information undergoes selective pressures, and advantageous traits are retained while less beneficial or redundant sequences may be lost over time. This process could be loosely compared to compression, as the genetic information is refined and optimized for the organism's survival. Additionally in genomics, for instance, in DNA sequencing, advanced algorithms and techniques are employed to compress and store large amounts of genomic data.

2. The 24-bit sequence is further split into three 8-bit RGB values. Splitting this sequence into three 8-bit RGB values can be compared to dissecting the genetic information into specific traits or characteristics, each represented by one of the RGB components. Consider each RGB value as a representation of different genetic attributes, such as physical traits, biochemical functionalities, or regulatory elements (enzymes). The red (R), green (G), and blue (B) components could symbolize distinct genetic dimensions, like structural genes, regulatory regions, or hereditary sequences.

3. The resulting RGB values are assigned to the corresponding cells in a grid, forming a visual representation of the genes (weights) of the agents (neural networks) for each time step. The assignment of resulting RGB values (gene sequences) to corresponding cells in a grid in GHC can be compared to creating a genetic portrait or map that visually represents the collective traits and characteristics of a agents' genes (weights) over time. Each cell in the grid symbolizes an individual, and the RGB values assigned to it depict the genetic features carried by that particular member (agent) of the population.

The GHC tool's visualization can provide insights into the diversity, convergence, or divergence of gene sequences (and hence agents) over time. If the colors are too diverse, it may indicate a wide range of gene sequences, potentially representing distinct species. Similar colors or longer trajectories of the same color may signify convergence or stability

[2]We provide all examples and supporting material for the corresponding Genotypic Diversity tools in Appendix A

in the evolved agents (resulting strong ancestor-descendant relationships), suggesting a consistent genetic pattern. Longer gradients of colors over extended time steps can indicate sustained genetic alterations (via mutations, adaptations, and variations). Finally, from the methodology, it can be derived that for a diverse agency there should be significant shift in colors where as for a strong agent relationships (indicating very less alterations) there should be longer trajectories. It can also be drawn that conditions may arrive where species born, grow, survive, die, and extinct. In our system, it is very less probable that a certain specie invade over others, because of mutation. Overall, GHC serves as a valuable tool for interpreting the genetic evolution of agents and can be likened to observing genetic diversity and trends in real-world biological systems.

3.4.2 Random Weight Selection Plot (RWSP)

This tool facilitates a dynamic visualization of the genetic evolution of agents with the help of their respective gene sequences. At each time step, three specific genetic traits are chosen from a pool of genes, by selecting random indices, and these indices remain consistent throughout the evolution. The resulting animation provides a grid representation, with each cell displaying the selected genetic features through color-coded values. By maintaining the focus on these specific indices over successive time steps, the tool allows for a detailed examination of the impact of these specific genes on the overall adaptation and behavior of the simulated agents. One can consider this tool as a genetic microscope which adjusts its focus only on three specific genetic features chosen randomly at once, across successive generations. It's akin to spotlighting particular genes in a real-world species, studying their influence on the entire population's evolution. The evolving grid of color-coded values mirrors the changing genetic landscape, offering a focused view into how these chosen genetic traits steer the course of adaptation over time. One can argue it's very similar to GHC; however, GHC comprehensively visualizes the evolving genetic landscape by hashing and color-coding all genes, the RWSP zooms in on a few specific genes across generations. The latter provides a more focused examination, akin to observing the impact of specific genes in a controlled genetic experiment. This selective approach in the RWSP allows for an in-depth analysis of the dynamics of particular genetic traits. Overall, GHC and RWSP both have their own advantages, offering comprehensive insights into overall genetic diversity and focused analyses on specific gene dynamics, respectively.

3.4.3 Unique Color Count Plot

This combined plot summarises the information from GHC and RWSP both. It just count the number of unique colors at every step. The idea is to show count plot for both the tools in a single plot such that it becomes easy to visualise the count of species increasing-decreasing over time. Ideally, RWSP is supposed to perform poor compared to GHC. We discuss more about the analysis of these plots in later chapters of the thesis.

3.5 Proposed Alternatives

One of the methods that we additionally propose, to track the long term trajectories of ancestor-descendant relationships and genealogies, is called as Ancestral Descendant History Encoding (ADHE). It works as extended version for GHC. This concept is a complex idea related to the representation of genetic sequences in a high-dimensional

space. An *ancestor* in this context is a genetic precursor to a specific sequence. In other words, it's a genetic entity that existed in the past and gave rise to the current sequence. This representation is used to visualize genetic relationships between sequences. The idea is to assign colors to sequences in such a way that similar sequences are represented by similar colors. This ensures that sequences with a common ancestor are assigned colors that are close in the color space. Self Organizing Feature Maps are designed to capture the underlying structure in complex, high-dimensional data and present it in a lower-dimensional space while preserving the relationships between data points (but not PCA). Based on our proposed framework, there can be two kinds of ancestors. First, Last Common Ancestor (LCA) or Most Recent Common Ancestor (MRCoA) which can help us tracking the *haplotypes* [8], [12] and hence the early signs of the parents that have first entry in the population history. The primary purpose of this approach is to analyze genetic relations. It allows for the categorization of sequences into groups based on their common ancestry. This categorization helps in understanding how different gene sequences are connected evolutionarily. Second, Most Recent Clonal Ancestor (MRClA), which links a gene to the immediate clonal ancestor of a clone emerged in the process of self-replication with mutation. It is not the scope of this work to implement this approach for now.

3.5.1 Clustering Neural Weights Approach (CNWA)

CNWA is an additional tool proposed with this work to analyze the behavior of agents by clustering their genes over time. This algorithm utilizes K-Means clustering to categorize the genes of agents into distinct groups, representing different species. The initial assignment of cluster colors helps in tracking the consistency of species across time steps. The tool then employs Principal Component Analysis (PCA) to reduce the dimensionality of the large genetic data and generate visualizations. The resulting plots provide insights into the clustering patterns, showing how different species evolve and interact over time. Additional visualizations, such as area plots and stacked bar plots, offer further insights into species proportions and their changes over successive time steps, enriching the analysis of genetic diversity and dynamics within the population. It can be well explained using Algorithm 4. To understand significance of this process, firstly, standardization of features using the mean and scaling to unit variance is like normalizing genetic traits within a population. This step ensures that the genetic information is comparable and operates on a consistent scale. The K-means clustering process can be associated with the formation of species or groups in a population based on shared genetic characteristics. The algorithm iteratively refines these clusters, mimicking the process of natural selection favoring distinct genetics that enhance the fitness of each "species" (clusters). Moreover, the Principal Component Analysis (PCA) step is akin to identifying the essential genetic features that distinguish one species from another. It's as if we're pinpointing the key genetic traits responsible for the unique characteristics of each of the species. Finally, the visualisations provides evolving genetic landscape of agents where observing the shifts in clusters and the prominence of certain genetic traits over time is comparable to tracking the genetic diversity and adaptation of species in a changing environment respectively.

Algorithm 4 Clustering Neural Weights Approach (CNWA) with k-means and PCA

procedure ClusteringWeights(W) ▷ W: List of weights
  $k \leftarrow$ number of clusters ▷ Set the number of clusters
  $c \leftarrow$ dictionary to store cluster colors
  for $i \leftarrow 0$ **to** length(W) **do**
    $w_p \leftarrow W[i]$
    ▷ Standardize the features by removing the mean and scaling to unit variance
    $n_d \leftarrow \frac{w_p - \text{mean}(w_p)}{\text{std}(w_p)}$
    ▷ Initialize cluster centroids randomly for KMeans
    $centroids \leftarrow$ Randomly select k data points from n_d
    repeat ▷ Assign each data point to the nearest centroid
      for $j \leftarrow 0$ **to** length(n_d) **do**
        $c_j \leftarrow \text{argmin}_{\text{centroid } c_k} \text{ distance}(n_d[j], c_k)$
      end for
      ▷ Update cluster centroids based on the mean of assigned data points
      for $k \leftarrow 0$ **to** k **do**
        $c_k \leftarrow \frac{1}{\text{count}(c_k)} \sum_j n_d[j]$ for all $n_d[j]$ where $c_j = c_k$
      end for
    until Convergence
    ▷ Principal Component Analysis (PCA) for dimensionality reduction
    $X \leftarrow n_d^T \cdot n_d$ ▷ Compute the covariance matrix
    $\lambda, V \leftarrow$ Eigen decomposition of X ▷ Compute eigenvalues and eigenvectors
    $P \leftarrow$ Top-2 eigenvectors of X ▷ Select the top-2 eigenvectors
    $r_d \leftarrow n_d \cdot P$ ▷ Project data onto the principal components
    Create scatter, proportionate bar, and area plots for cluster points
  end for
end procedure

3.6 Research Design Considerations

We look back into the study of genetics and biology to put forth the methodologies proposed for this study. Within this section, we explore the decisions made during the planning and structuring of our research. Commencing with the mappings of *phenotype-genotype* and *genotype-phenotype*, we aim to gain insights into the underlying patterns and trends that characterize genetic divergence across spatio-temporal landscapes, as well as to understand the ways in which genetic variations manifest in observable phenotypic traits. It is very important to note that similar research works has been prominent to the genetics and computational biology. In Chapter B, we provide a biologist view of the proposed methodologies and the associated terminologies[3].

[3]Additional notes on *A Biologist View - Triangulation Study* is provided in Appendix B

Chapter 4

Implementation and Experimental Setup

Because of the experimental nature of this research, this chapter contains multiple sections and subsections that provide enough implementation details for the proposed methodology and some of the key aspects of the associated algorithms by which one can easily reproduce the results. To be specific, first section outlines the building blocks of the implemented methods, where in its subsections we will discuss implementation of NCA model, then implementation of measuring tools for PD and GD, and modularisation of the overall working pipeline. In the next section, we setup different experiments along with the proposed hypotheses. Finally, the system requirements and environment configuration for the experiments to run is provided in the last section. Since, the proposed methods include working with multi-dimensional data and large number of neural networks, the implementation utilises expensive operations. Therefor a curated resource manager is also plugged in the proposed NCA framework which is carefully designed in the hope to outperform existing libraries together with parallel processing for NCA framework, discussed in miscellaneous details later in this chapter. Most of the implementation details and code are provided in this chapter. Rest of the implementation details can be found at `https://github.com/s4nyam/Self-Replicating-NCA`.

4.1 NCA and Corresponding Tools

Our proposed NCA framework is built on top of simple CA architecture. Each pixel corresponds to a cell and an agent in the grid. Moreover, since the NCA is multi-channel, we perform depthwise scanning of each cell via corresponding agent. In other words, NCA model is made up of cells in two layers along with each cell is associated with a neural network that could sense and process neighborhoods. Further, we let the NCA to evolve only on the basis of the output of neural network. We have already seen related concepts in 2.2.1, for example self-replication, mutation and self-organization of the proposed NCA. We use neural networks for agents and pixels for cells interchangeably throughout the chapter. Further qualitative and quantitative studies are done via various proposed PD and GD tools. PD tools focus only on the statistical analysis of the growing NCA cells, where as GD tools analyses genealogies and ancestor-descendant trajectories for the evolved NCA agents. Let us formalise the further implementation details in the following sub-sections.

4.1.1 Neural Cellular Automata

The proposed NCA implementation consist of some constants and variables that were initialised before running the simulation. For simplicity, we keep the parameters with very small values which can help us further debug and study rest of the implementation. However it is very easy to scale these variables and parameters to a larger values which we further discuss in following sections as part of experimental setup. Firstly, Height and Width of the CA are initialised, for this subsection, we keep *width=3* and *height=3* therefor we have *9* pixels in total initialised in the CA grid. Hence grid size is a tuple of (*width, height*). Next, we set initial population that is going to be seeded at very beginning of the simulation. We call them *founders* of the population as these cells or agents are going to reproduce and perform growth in the NCA in future. Initial cells are set alive using probability value, (for example, *init_prob=0.12*) which is further processed to calculate minimum number of pixels that are going to be initialised depending on the grid (works as percentage of pixels that are going to be alive initially based on the grid size). Therefor, $min\_pixels = max(0, width \times height \times init\_prob)$. For our case so far, it is calculated as *1.08* which will make at least one cell alive and there's 8% chance that another cell will become alive or not (making at most two cells alive in the NCA grid at very beginning). Then we initialise number of channels for the CA as *2* where one works as α and another as hidden chemistry. Further, we set α (*to 0.6*) as threshold that regulates the livelihood of an individual. We discuss the usage of α in more details in later discussions. As we built the rest of the code, we realised it is also wise to initialise two more variables related inheritance and mutation in beginning. Particularly, we initialise a probability (percentage chances) that neighboring cells will be inherited by the current cell (inheritance of weights of the neighboring neural network, also called as genes for our discussion). We call it inheritance probability (and set it to *0.2* for this discussion). Additionally, we initialise parameter perturbation probability that specifies the chances that a gene of a particular gene is going to be perturbed or mutated. It can be interpreted as, if the size of gene pool is n then the number of genes that are going to be mutated is size of gene pool multiplied by the probability ($0.2 \times n$). Finally, we initialise time steps (for this case, *3*) which tells the number of evolutionary steps (generations) that the NCA is going to be evolved.

After initializing the parameters, let's delve into the core NCA framework. Before proceeding further, it's beneficial to grasp the fundamentals of Artificial Neural Networks (ANNs). While we won't delve extensively into ANNs, we provide essential information to facilitate a deeper understanding of NCA and the remaining implementation details. Once the discussion of the ANN employed in this research is complete, we will utilize it unchanged for the subsequent chapters. Proposed architecture of simple ANN consists of two fully connected (FC) layers separated by a sigmoid activation function. The input size is determined by depthwise neighborhood for two-layered NCA (as 9×2), and the output size of the second layer is same as the number of channels in NCA. It might be confusing to understand the interplay of layers, however we enforce our proposed architecture with two layered ANN that takes input from two grid NCA. The weights and biases of ANN are initialized to zero (which implies agents start with zero brain). The forward pass involves linear transformations and the application of the sigmoid activation function. In short, input to the ANN is number of neighboring cells in Moore neighborhood (which is *9*) multiplied by number of channels in the NCA (which we fixed as *2*). We set the hidden size of the ANN as *2*. Output from the ANN is two values that adjusts to the

corresponding Alpha layer and hidden layer of the NCA. Summary of the proposed simple ANN is provided below:

```
-----------------------------------------------------------------
Calculating ANN Parameters
-----------------------------------------------------------------
input_size_fc1 = 9 * NUM_LAYERS
output_size_fc1 = NUM_LAYERS
weight_params_fc1 = input_size_fc1 * output_size_fc1
bias_params_fc1 = output_size_fc1

input_size_fc2 = NUM_LAYERS
output_size_fc2 = NUM_LAYERS
weight_params_fc2 = input_size_fc2 * output_size_fc2
bias_params_fc2 = output_size_fc2

weight_params = weight_params_fc1 + weight_params_fc2
bias_params = bias_params_fc1 + bias_params_fc2
total_params = weight_params + bias_params
-----------------------------------------------------------------

-----------------------------------------------------------------
Model Summary
-----------------------------------------------------------------
Input Size: 18 (Depthwise Neighborhood)
FC Layer 1 Weight Parameters: 18 * 2 = 36
FC Layer 1 Bias Parameters: 2
FC Layer 2 Weight Parameters: 2 * 2 = 4
FC Layer 2 Bias Parameters: 2
Total Number of Parameters: 44
Layer 1: fc1, Input Size: 18, Output Size: 2
Layer 2: fc2, Input Size: 2, Output Size: 2
-----------------------------------------------------------------
```

While we focus our study only for *sigmoid* and *tanh* activation, we proposed two other activations ($ReLU$, and $LeakyReLU$) that can be experimented in future work. It would be interesting to see the comparisons of ReLU and LeakyReLU for the growing NCA as activations are *0* for negative neural network outputs and minuscule activations values, respectively. All four activations are compared in Figure 4.1 and 4.2.

Now that we have understood the architecture of the proposed ANN, we shift our focus to the implementation of rest of the NCA framework. We start with a completely dead CA grid that neither has any active cell or agent. Then, place the *founders* randomly into the α channel and then corresponding energy is also initialised with a random multiplier in second channel (it can be done with multiple options including (1) initialise all energy with one (2) initialise all with same random value, or (3) initialise all with different uniformly random values), along with the initialisation of corresponding neural networks with uniformly random weights. However for simplicity, the implementation initialises all founders to *1*. In simpler words, all cells that were randomly initialised in the α channel, their associated

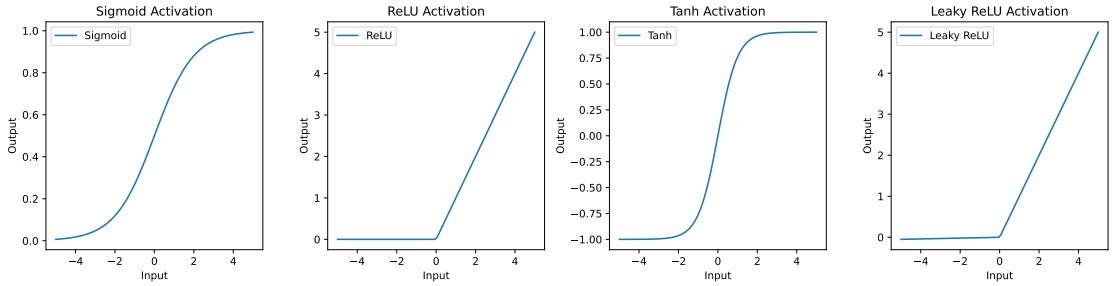


Figure 4.1: Comparing four activations. Sigmoid, ReLU, TanH and LeakyReLU respectively.

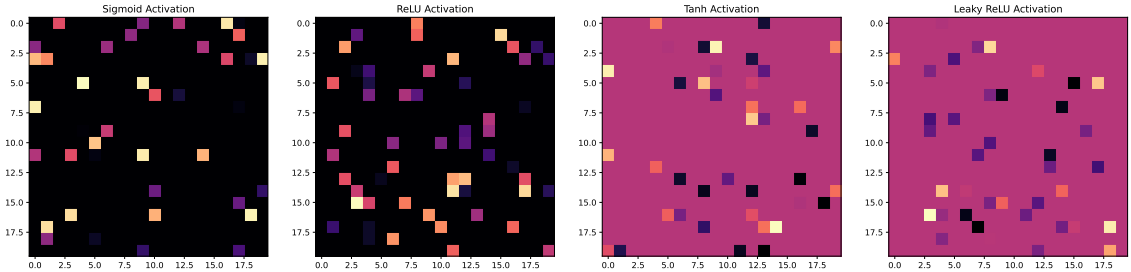


Figure 4.2: Effect of activations on grid values. The grids of size 20 are initialised with 10% cells here. These cells are in the ranges of activation functions. First, *Sigmoid* has a range of 0 to 1 and hence a desired α threshold of 0.5 can be suitable. Second, *ReLU* activation has range 0 to ∞. Third, *Tanh* activation has a range -1 to 1 which makes suitable to put a α threshold of 0. Fourth, LeakyReLu has range of -∞ to ∞. An important observation to note here is that *Sigmoid* and *ReLU* only have positive range. While *Tanh* has a range of -1 to 1, we decide an α of 0, this also ensures color inversion problem of matplotlib colormaps when plotted negative values.

chemistry and agency are also initialised randomly. Once we place the founders in the α channel, we scan complete α channel again and check for the cells that are alive ($> \alpha$), initialise weights. Rest all cells remain dead. Once the CA is initialised, it undergoes multiple runs equal to provided time steps. For each time step, the algorithm iterates the CA grid, scan it and perform update CA function. The update CA function takes two inputs, the grid itself and the list of neural networks. While we enlist important functionalities of the update CA logic in the following text, we release and refer the associated code repository provided in Section 1.5 (especially, *updateCA*() function) in order to understand exact meaningful interpretations of the variables.

1. Initialization:

    - A new CA grid (`new_ca_grid`) is created to store updated values.
    - A nested function, `process_neighborhood`, is defined to handle computations for each cell's neighborhood.
    - Note! A temporary ANN list is initialized to store updated ANNs. While previous gen's ANNs list is used for other update requirements. In short, for each generation we initialize empty list that stores updated ANNs while we also preserve last gen's ANNs.

2. Processing Neighborhoods:

   - For each cell, a 3×3 neighborhood is extracted for each channel of the CA. This yields, $2 \times 3 \times 3$ array of values.
   - The neighborhood is flattened and used as input for the associated neural network (`ca_nn_list[idx]`).
   - The neural network processes the input, producing an output that represents updated values for the pixel.

3. Inheritance and Mutation Mechanism:

   - If the alpha value of any cell in the neighborhood is greater than a threshold (α), the neural network inherits the weights from a randomly chosen living agent corresponding to the selected cell in the neighborhood. This process is chanced with an inheritance probability. Hence first check is done to ensure there exist living agents in the neighborhood, so as to calculate the neural network output. And second condition ensures, not every cell in hand undergoes inheritance. Following code snippet formalises this process:

     ```
     if((neighborhood > ALPHA).any()):
         output = ca_nn_list[idx](neighborhood)
         if(random.random() < INHERTIANCE_PROBABILITY):
     ```

   - The calculation of *output* is carefully done right after checking the living neighbors, to ensure agents do not return unexpected garbage values due to memory redundancy.
   - After checking two conditions, we now find and list the neighboring cells with alpha values greater than α, which provides a concise indices of the living cells. Now we perform a final check for livelihood of the pixels (if this passes, that means there is at least one neighbor that is greater than α).
   - Pick a random living neighbor and copy the weights. For instance,

     ```
     selected_pixel = random.choice(high_alpha_pixels)
     ni, nj = (i + selected_pixel[0]) % WIDTH, (j + selected_pixel[1])
                                                     % HEIGHT
     ca_nn_list[idx] = copy.deepcopy(ca_nn_list[ni * WIDTH + nj])
     ```

   - The inheritance involves copying the neural network from the selected neighboring pixel and perturbing specific amount of its weights. We do this by parameter perturbation probability. For instance,

     ```
     for name, param in ca_nn_list[idx].named_parameters():
       if 'weight' in name:
           if random.random() < parameter_perturbation_probability:
               param.data += torch.randn_like(param.data) *
                                           random.uniform(-1, 1)
     ```

   - If the alpha value in any of the neighbors is not greater than α, the weights of the neural network associated with the current pixel are initialized to zero. Because even if the current pixel has higher α value it participates in neighborhood. In simpler words, current cell is also neighborhood of itself if its alive.

4. Updating the CA Grid:

   - The function updates `new_ca_grid` with computed values for each channel of the CA.
   - A sigmoid or tanh transformation is applied to the updated values (of the complete updated CA grid) to ensure they fall within the range [0, 1] or [-1, 1] respectively. Since, we are dealing with real numbers, it was difficult to track livelihood of cells on the basis of a fixed α value. Therefor to fix an α value and ensure all the cells that have values from 0 to α dies or -1 to α respectively, and rest stays alive, we perform this sigmoid or tanh transformation.
   - Finally, values below a certain threshold (α) are set to 0, simulating cell death. Check if any value in alpha channel is less than α at the current position, If any value is less than α, set values in all channels at the current position to 0.
   - We also kill cells on the basis of external life budget. We maintain a numpy grid that increments the corresponding cell value as generation passes by. As soon it reaches the threshold or life-budget, a separate routine is called to ensure these cells die. We collect all indices of the cells that are just inherited to prevent premature killing of those cells. We just don't kill those cells that are below α and have non-zero genetic material or neural network weights.
   - Please note that the sigmoid function, denoted as $\sigma(x)$ or sigmoid(x), is defined as:
     $$\sigma(x) = \frac{1}{1 + e^{-x}}$$
     Its properties include:
     - **Domain:** The sigmoid function is defined for all real numbers, i.e., domain$(\sigma) = (-\infty, \infty)$.
     - **Range:** The range of the sigmoid function is between 0 and 1, i.e., range$(\sigma) = (0, 1)$. As x approaches positive infinity, $\sigma(x)$ approaches 1, and as x approaches negative infinity, $\sigma(x)$ approaches 0.
   - Moreover, $\sigma(0) = 0.5$. In our case where cells stay dead for most of the time in the beginning, it can clearly be seen that all zeros will be replaced with 0.5 and hence the α threshold has to be meticulously decided at least above 0.5 to eliminate dead cells.
   - Additionally, the hyperbolic tangent function, denoted as $\tanh(x)$ or $\tanh(x)$, is defined as:
     $$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$
     Its properties include:
     - **Domain:** The hyperbolic tangent function is defined for all real numbers, i.e., domain$(\tanh) = (-\infty, \infty)$.
     - **Range:** The range of the hyperbolic tangent function is between -1 and 1, i.e., range$(\tanh) = (-1, 1)$. As x approaches positive infinity, $\tanh(x)$ approaches 1, and as x approaches negative infinity, $\tanh(x)$ approaches -1.
   - Furthermore, $\tanh(0) = 0$. Given the same context where cells stay dead for most of the time initially, it can clearly be seen that all zeros will remain zeros. Therefore, the α threshold must be carefully determined to ensure the accurate behavior of the system.

How are we doing mutation?

1. Prepare a mask that one hots random positions based on parameter perturbation probability.

2. $rand_like(param.data)$: This function generates a tensor with the same shape as parameter data, filled with random numbers from a uniform distribution over the range [0, 1).

3. First a random mask is prepared using uniform distribution and then comparison is made to one hot it, such that values below parameter perturbation probability are 1 and rest 0.

4. Add random noise to the parameter tensor. It generates random numbers from a uniform distribution with the same shape as parameter data. The noise is added element-wise to the parameter tensor, but only where the corresponding element of the mask is True.

5. To prepare mask we use uniform distribution and for adding noise / perturbation we also use uniform distribution.

6. Note! ANNs are mutated only once in their lifetime. If budget of life span is high, we might see less diversity however if the budget is small then we should see more diversity (more new ANNs and more mutations.)

7. When we are mutating, the parameter space is a list of 40 elements (while it should be possible to mutate all named parameters, we are mutating only 'weight' parameters not the biases) – idea is weights are learnable parameters and hence only those are mutated.

4.1.2 Phenotypic Diversity Tools

In this subsection, we will discuss the implementation of all Phenotypic Diversity (PD) tools. Essentially, we have already discussed enough theoretical and algorithmic side of these tools in Chapter 3 that can be easily reproduced and implemented. We provide a small implementation summary of these tools as follows:

1. Cellular Type Frequency Plot (CTFP) simply plots the frequencies (or counts) of different states. The idea is to first round the grid values in a certain precision (say *1*) and hence we get all living cell states as *0.6, 0.7, 0.8, 0.9 and 1.* In addition to these 5 states, we also have one dead state as *0.* By collecting the active total count of each cell state for a time step, we get a frequency dictionary. Collection of all such dictionaries for all time steps is prepared and then plotted (Frequency vs Step Number) for all 6 states.

2. Global Entropy Plot (GEP) calculates entropy values at each time step based on the spatial frequency distribution of values. First, calculate the spatial frequency ($\rho = \frac{count}{T_c}$), then individual contributions of each state as,

   ```
   (-each_rho) * math.log(each_rho)
   ```

perform the summation of contributions and finally normalise the resulting value with the total number of possible states as,

```
H_t = (1/math.log(total_states)) * Σ
```

where Σ is summation of individual contributions. This results global entropy. This is provided in stepwise manner in Section 3.3.2.

3. Gross Cell Variance Plot (GCVP) can be calculated in very few steps. Determine the median cell state, then for each cell in the CA, calculate $\sigma_{gross} = \sigma_{gross} + (1 - \delta)$ where δ is *kroncker_delta* using the formulae mentioned in Algorithm 2.

4. Cell Local Organisation Global Variance (CLOGV) is implemented as same as presented in Algorithm 3. First calculate normalized frequencies of each state as,

```
normalized_count = (count - min_count) / (max_count - min_count)
```

Secondly, local statistics as,

```
for each_neighbor in neighbors_of_current_cell:
    sum_of_frequencies_of_cell = sum_of_frequencies_of_cell +
                frequency_of(each_neighbor,frequency_dict_temp)
mu_loc_current_cell = sum_of_frequencies_of_cell/9
sigma_loc_current_cell = math.sqrt(sum_of_squared_frequencies)/3
```

And then finally calculate μ_{glob} and σ_{glob} is computed. We refer code on GitHub.

(Please refer 6.1 for quick reference to full forms).

Abbreviation	**Full Form**
PD	Phenotypic Diversity
GD	Genotypic Diversity
CTFP	Cellular Type Frequency Plot
GEP	Global Entropy Plot
GCVP	Gross Cell Value Plot
CLOGV	Cell Local Organisation Global Variance
RWSP	Random Weight Selection Plot
GHC	Genotypic Hash Coloring
RWSP	Random Weight Selection Plot
CNWA	Clustering Neural Weights Approach

Table 4.1: List of abbreviations recently used

4.1.3 Genotypic Diversity Tools

GD tools comprises of three implementations. Firstly, Genotypic Hash Coloring (GHC) compresses the weights (of any size) of the neural network to 24-bit binary sequence using hash function. The purpose of this function is to take a weight sequence as input and hash using hash function. The resulting hash values are then converted to 24-bit integers by taking the modulus % 2. This effectively maps each hash value to either 0 or 1. One can argue here, the % 2 operation may result in hash collisions (different inputs mapping to the same hash value), as it reduces the range of hash values to just two possibilities (0 or 1). However because of the use case we have used such functionality. The *hash()* function in Python is deterministic within a single execution of a program, meaning that for a specific object, it consistently produces the same hash value throughout the execution. If the *hash(value)* is called multiple times for the same value within a program run, it will yield the same result each time. However, it's important to note that the *hash()* function is not designed to be *pseudo-random* or *cryptographically* secure. While it swiftly generates hash values for objects to aid in quick comparisons, it lacks the characteristics of unpredictability typically associated with *cryptographic* hash functions. The primary goal of the *hash()* function is efficient mapping of objects to hash values rather than ensuring *cryptographic* strength or *pseudo-randomness*. This makes it suitable for general-purpose hashing within a single program execution but may not be appropriate for applications requiring cryptographic-level security (which we are not concerned about for this project). Finally, split the 24-bit sequence to three 8-bit sequences and plot them over a grid using a stacked tuple *(R,G,B)*. Please note that all these 8-bit binary string is converted to corresponding integer values (which ranges from *0-255* for *R,G,B*).

Secondly, Random Weight Selection Plot (RWSP) is a proxy to GHC. Rather than calculating hash, we directly plot three random weights from the pool of *44* weights of a single ANN. However the indices of those randomly chosen genes remains same throughout the simulation. In other words, first initialise three random indices or positions, then keep it static throughout the process so that a fair chance is given to same genes to get picked in an agent's gene sequence. This concept is very prominent in biology as *codon*, where we try to identify *Heterozygous*, *Homozygous* and *Codominance* in organisms. This further helps to study *Homologous Structures* and *Homology*. Overall this method is very much similar to *selective or targeted gene analysis* or *gene-centric evolution studies*.

A combined plot for GHC and RWSP is also visualised for unique color count. First, it counts all the unique colors by comparing the RGB tuple (a list of 3 values) and results the number of unique such lists. Second, we subtract one from the count to ensure it does not count void (black) color which is all dead. Finally, it results unique color count for generations and plot a line curve corresponding to the count and generations. We refer the Supplementary material on GitHub, End Deliverables in Section 1.5, and Google Colab Notebook `https://bit.ly/neuralCA` for further implementations.

4.1.4 Hashing Test

Considering huge amount of data, it is highly possible to result collisions for GHC and hence same color for different genotype (neural network weight parameters). To investigate more, we did a hashing test to understand the collisions. We started asking questions, Why we see GHC has more diversity than RWSP? Why GHC returns same color, because if inheritance and mutation works GHC should return different colors because of different

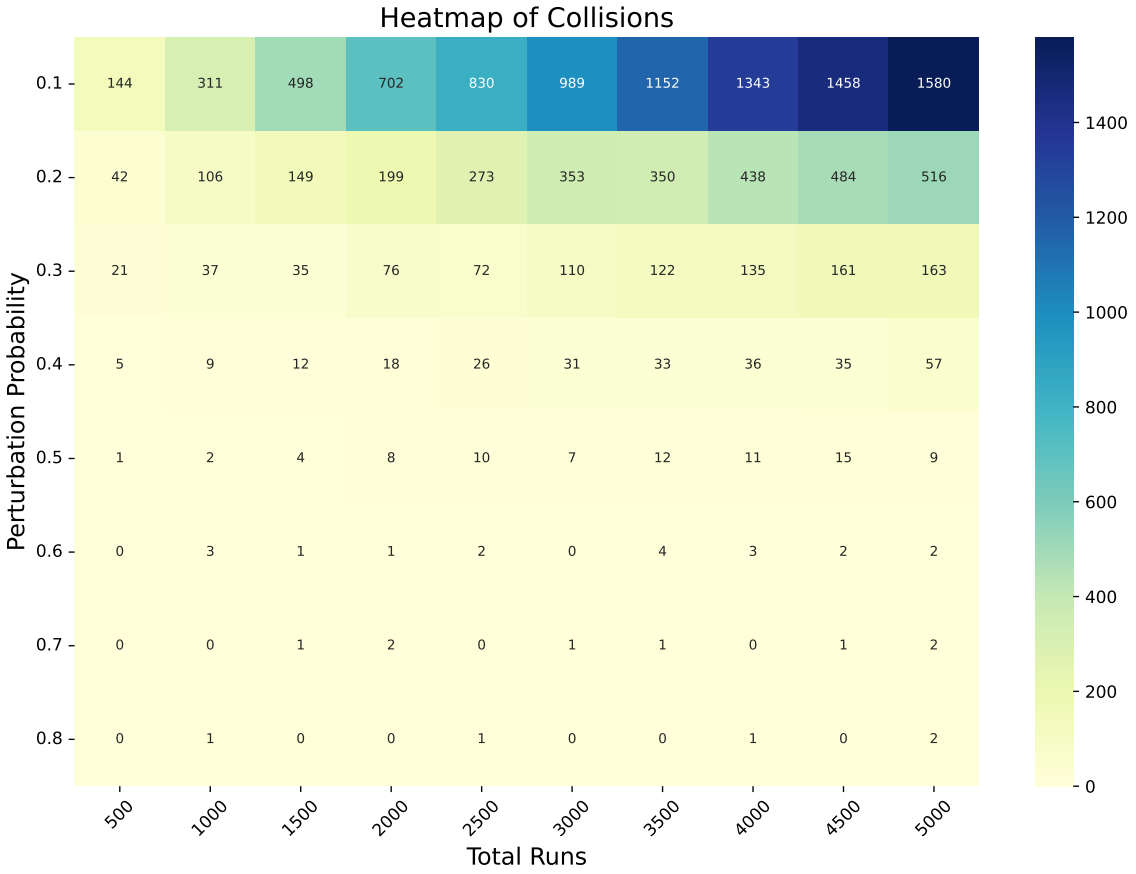


Figure 4.3: Hashing test example plot for different runs vs perturbation probability. Higher perturbation implies less collisions, smaller perturbation more collisions.

genetic material? In RWSP life is simple, we just pick three positions in gene pool and tracks it. High chances they stay same for long generations and hence we see same color on board. BUT GHC hashes complete genetic material. In our working pipeline we never inherit complete genetic material as it is, which means there are mutations working in action for those inheritance. If inheritance has worked and mutations has happened how GHC returns same color for two ANNs with different genetic material? One idea to answer these questions is to try inbuilt hash function which is used in GHC. As we are using $hash()$ function to squash 44-valued gene sequence to 24-bit binary sequence then splitting it to RGB values. The idea is, try out this hash function on some dummy data and check multiple outputs. Check whether we get collisions for small changes in the 44 parameter space list as we do in mutations for the ANNs while inheritance. In short, simulate this function for some random data but with similar mutations as we are doing in proposed NCA framework. The question to put here is how a 44 valued sequence with slightly different parameters results same has value? In implementation, self-replication (inheritance) happens only with 20% mutation (for example). If I am corrupting 20% (for example) values while duplication by adding or subtracting a small value between -1 to 1 (as part of mutation), how come its resulting same hash value? The steps to perform hashing test is as follows.

1. Generate random list of 44 elements (Our use case has 44 parameters). Call it as list 1.
2. Prepare another list 2 which is copied and mutated version of list 1. And 20% elements are mutated.
3. Mutations here is same as that we have in NCA. The 44 elements were initialized values between -1 to 1 uniformly. Mutation also has addition of value from -1 to 1 uniformly.
4. Now that we have list1 and list2, perform hashing by $hash()$. Idea is for most of the cases the inherited gene sequence list1 and mutated gene sequence list 2 should collide at least once if this is the case.
5. We run this test and provide parameter perturbation vs runs plot in Figure 4.3. Indeed there are collisions. Corresponding code for this test can also be found in the same GitHub repository provided in Section 1.5.

4.1.5 Working Pipeline

The implementation is modularised into three working pipelines. The very first is in Google Colab that is capable of running small experiments with small grid sizes and smaller agents (ANNs). This pipeline works well for debugging and experimenting new tools and frameworks. We use this code base for more of our implementation playground. While this is great for smaller runs and experiments, it cannot evolve large and complex grids because of the online resource limitation. Second, to run larger experiments we prepare another code base that is capable to evolve large NCA simulations with complex ANNs over expensive resources. We use SLURM to run our second implementation on high powered resources available in-house and cloud (community GPU Clusters). It includes the part that can only evolve NCA and exports the *pkl* weight files which can either be further be evolved in future in continuation or analysed using third working pipeline. The third includes all rest of the analytical tools that can further be run on Amazon Sagemaker instance. The necessity of using AWS is because of loading these large number of weights in memory, we need *Memory Optimised* workloads, not the *Accelerated Computing*. To summarise, first pipeline is hosted on Google Colab for experimentation, second on NVIDIA GPU community clusters and third one sagemaker instances[1].

4.2 Experimental Setup

We experiment two different setups for the NCA on the basis of starting variables (initial parameters) for the NCA testing (Colab specific) and for large experiments (SLURM specific). Table 4.4 summarises experimental setup for small experiments on Google Colab and Simula eX3. And Table 4.5 summarises the batch of ten large scale experiments that are let run on expensive computational resources on Simula eX3. Please refer the Table 4.3 for full form of abbreviations used in Tables 4.4 and 4.5. All large and small runs experiments can be downloaded from Archive - `https://archive.org/details/`

[1]We use free tier version of the Google Colab environment at `https://colab.google`, eX3 from Simula by Research Council Norway at `https://www.ex3.simula.no/` and AWS instance at `https://aws.amazon.com/sagemaker/pricing/`

Table 4.2: Combined System Configurations

Component	**Google Colab**	**eX3 - Simula**	**AWS Sagemaker**
Processor	Xeon 1-core	Milanq 64-core	Intel 48-core
RAM	12.72 GB	1.5 TB	384 GB
Storage	107.7 GB SSD	115 TB beegfs SSD	500 GB SSD
GPU	NVIDIA Tesla T4 (1 GPU)	NVIDIA Volta A100 (8 GPUs)	NA
GPU Memory	14.41 GB	80 GB (each)	NA
OS	Ubuntu 18.04 LTS	Ubuntu 20.04.6 LTS	Amazon Linux 2
Python Version	3.x	3.x	3.x
Stack	PyTorch	PyTorch	Conda Torch
CUDA	12.0	12.0	NA
Price	NA	NA	US$ 3.629 per hour

Table 4.3: Full forms of corresponding short forms used in other tables.

short form	full form
exp	Experiment Number
W	Width of NCA
H	Height of NCA
init	Initial seeded agents percentage
ppp	Parameter Perturbation Probability
ip	Inheritance Probability
b	Budget Life Threshold
α	Alpha Value Threshold
ch	Channels in Total for NCA
gens	Number of Generations for NCA to evolve
act	Activation on Board (applied to squash)
ttr	Expected Time to Run

`@evolutionary_lenia`. While the implementation for each of the framework and tools are discussed in this chapter from scratch, as part of our exploratory work, we came up with several other implementations of the NCA frameworks that are already in the literature. Precisely, we consider SIM's [33] experimental approach they mention in various sections of their work and methods towards the ideal frameworks. SIM, an over-engineered approach towards open endedness and self-organising self-maintaining growing NCA, it formalises numerous experimental ideas which turned out to be motivation for the testing of our NCA framework. For example, inheritance with mutation.

4.3 System Requirements

There are total three types of system environments we have used for this research project. As explained previously, we have three different working pipelines which utilises different environment configurations. Table 4.2 abstracts all system configurations used in this research work as per requirements. Please also visit the GitHub repository for recent system requirements that are updated at `https://github.com/s4nyam/Self-Replicating-NCA` at the section of "Configuration on Simula eX3".

Table 4.4: Experimental Setup for Small Runs. This table shows example runs that could be interesting. There were total of 1680 Experiments performed.

exp.	W	H	*init*	*ppp*	*ip*	b	α	*ch*	*gens*	*act*	$ttr(m)$
1.	50	50	0.1	0.2	0.2	4	0.5	2	300	*sigmoid*	70.1
2.	50	50	0.02	0.02	0.05	16	0.5	3	300	*sigmoid*	89.8
3.	50	50	0.20	0.02	0.05	8	0.5	2	300	*sigmoid*	70
4.	50	50	0.08	0.02	0.4	8	0	3	300	*tanh*	74.2
5.	50	50	0.12	0.02	0.4	8	0	3	300	*sigmoid*	78.4
6.	50	50	0.5	0.02	0.3	8	0	3	300	*sigmoid*	50.7
7.	50	50	0.5	0.02	0.3	4	0	3	300	*tanh*	66.0
8.	50	50	0.12	0.02	0.3	16	0.5	3	300	*sigmoid*	62.5
9.	50	50	0.5	0.02	0.5	16	0.5	3	300	*sigmoid*	64.3
10.	50	50	0.02	0.02	0.4	16	0	2	300	*tanh*	70.8
11.	50	50	0.12	0.02	0.5	16	0	3	300	*tanh*	78.2
12.	50	50	0.08	0.02	0.5	16	0	2	300	*tanh*	65.6
13.	50	50	0.12	0.2	0.075	16	0	3	300	*sigmoid*	64.7
14.	50	50	0.5	0.2	0.2	4	0.5	2	300	*sigmoid*	63.6
15.	50	50	0.02	0.2	0.2	8	0	2	300	*tanh*	71.2
16.	50	50	0.2	0.2	0.5	16	0.5	3	300	*tanh*	72.0
17.	50	50	0.08	0.2	0.5	8	0.5	3	300	*sigmoid*	58.8
18.	50	50	0.12	0.8	0.1	8	0.5	3	300	*sigmoid*	52.5
19.	50	50	0.2	0.8	0.2	16	0	3	300	*tanh*	64.1
20.	50	50	0.2	0.5	0.2	8	0.5	2	300	*sigmoid*	70.0
21.	50	50	0.08	0.5	0.1	16	0.5	2	300	*sigmoid*	73.1

4.3.1 Effective usage of the GPU and memory I/O

Firstly, parallelisation is key when dealing with large-scale computations, and *GPUs* excel in parallel processing. We utilize parallel programming framework (CUDA) to distribute the computation of individual cells and time steps across GPU cores concurrently. Additionally, we take advantage of batch processing to feed multiple instances of our CA to the GPU simultaneously, further enhancing parallelism. Finally, we carefully reduce I/O and GPU-CPU communications to ensure efficient data transfer between the CPU and GPU by minimizing unnecessary copies.

Table 4.5: Experimental Setup for Large Runs. This table shows the experiments that are tried for long generations handpicked from small runs of Table 4.4. Please also note that all the experiment numbers are hyperlinked with corresponding experimental data that redirects to the download link.

exp.	W	H	*init*	*ppp*	*ip*	b	α	*ch*	*gens*	*act*	$ttr(h)$
1.	200	200	0.02	0.02	0.1	4	0.5	2	1000	*sigmoid*	42.8
2.	200	200	0.02	0.02	0.1	4	0	2	1000	*tanh*	43.9
3.	200	200	0.02	0.02	0.1	8	0.5	2	1000	*sigmoid*	44.2
4.	200	200	0.02	0.02	0.1	8	0	2	1000	*tanh*	40.0
5.	200	200	0.08	0.02	0.1	4	0.5	2	1000	*sigmoid*	40.9
6.	200	200	0.08	0.02	0.1	4	0	2	1000	*tanh*	38.6
7.	200	200	0.08	0.02	0.1	8	0.5	2	1000	*sigmoid*	43.1
8.	200	200	0.08	0.02	0.1	8	0	2	1000	*tanh*	42.3
9.	200	200	0.02	0.02	0.5	4	0.5	2	1000	*sigmoid*	39.2
10.	200	200	0.02	0.02	0.5	4	0	2	1000	*tanh*	43.3
11.	200	200	0.02	0.02	0.5	8	0.5	2	1000	*sigmoid*	40.7
12.	200	200	0.02	0.02	0.5	8	0	2	1000	*tanh*	44.5
13.	200	200	0.08	0.02	0.5	4	0.5	2	1000	*sigmoid*	38.8
14.	200	200	0.08	0.02	0.5	4	0	2	1000	*tanh*	42.4
15.	200	200	0.08	0.02	0.5	8	0.5	2	1000	*sigmoid*	40.1
16.	200	200	0.08	0.02	0.5	8	0	2	1000	*tanh*	41.7
17.	200	200	0.02	0.02	0.1	∞	0.5	2	1000	*sigmoid*	42.2
18.	200	200	0.02	0.02	0.1	∞	0	2	1000	*tanh*	39.6
19.	200	200	0.08	0.02	0.1	∞	0.5	2	1000	*sigmoid*	43.0
20.	200	200	0.08	0.02	0.1	∞	0	2	1000	*tanh*	44.4
21.	200	200	0.02	0.02	0.05	∞	0.5	2	1000	*sigmoid*	40.5
22.	200	200	0.02	0.02	0.05	∞	0	2	1000	*tanh*	38.9
23.	200	200	0.08	0.02	0.05	∞	0.5	2	1000	*sigmoid*	43.8
24.	200	200	0.08	0.02	0.05	∞	0	2	1000	*tanh*	41.1

Chapter 5

Results

This project is highly experimental, it is essential to present all potential large run results to enhance the understanding of the proposed methods and experimental setup. In this chapter we refer Table 4.5 to show resulting plots and evolved NCA for all 24 experiments, then we analyse these plots and classify in different categorisations depending on the observed plots. The results, specifically contain four major plots that includes (1) Evolved NCA grid, (2) Cellular Type Frequency Plot (CTFP), (3) Combined Phenotypic Diversity (PD) Plot (GEP, GCVP and CLOGV) and Genotypic Diversity (GD) by count of Unique Color Plot (GHC and RWSP). We perform analytical and statistical significance based discussions on results with bigger interpretations and solutions to our research questions in chapter 6. Please also note the Table 4.5 is embedded with hyperlinks in experiment number column that redirects directly to the *tar* export files that can be downloaded and analysed independently.

5.1 Evolved Neural CA, PD and GD Tools

In this section we present plots for all large experiments. The Tables 5.1 (Experiment 1 to 4), 5.2 (Experiment 5 to 8), 5.3 (Experiment 9 to 12), 5.4 (Experiment 13 to 16), 5.5 (Experiment 17 to 20), and 5.6 (Experiment 21 to 24) shows evolved NCA grid, CTFP, PD tools and GD tools respectively. All these experiments are five-fold, that means the same experiment was ran five times and then averaged for validation. The plot shows the range-gap of all these five experiments as shades. Please note (1) Evolved NCA grid is picked randomly from all five experiments and hence shows any one grid among all five experiments, (2) CTFP plot is only averaged and not shows the range-gap for better interpretability of the plot. Again, the experiment number are hyperlinked to respective animations of evolution of NCA, RWSP and GHC videos on YouTube. This can also be viewed as playlist at `https://www.youtube.com/@growingnca/playlists`. We also show the evolved visuals of GD tools including RWSP and GHC in Tables 5.7 (Experiment 1 to 4), 5.8 (Experiment 5 to 8), 5.9 (Experiment 9 to 12), 5.10 (Experiment 13 to 16), 5.11 (Experiment 17 to 20), and 5.12 (Experiment 21 to 24). The visuals of RWSP and GHC tools (first generation and last generation) are picked randomly one from five-fold experiments. While five-fold experiments were done to average statistics, it is expected that visuals would perform more or less similar in all those experiments.

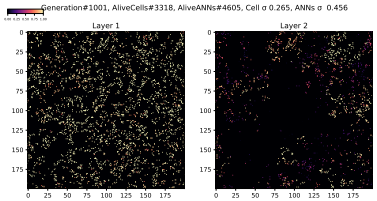
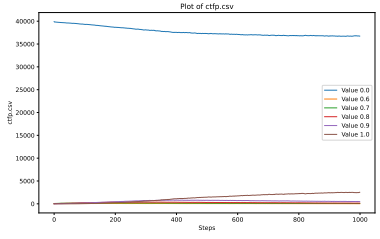
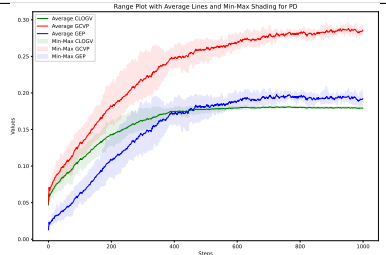
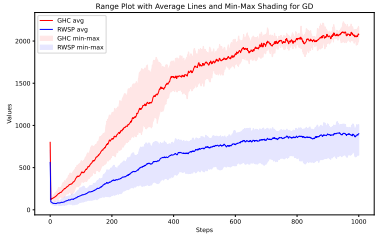
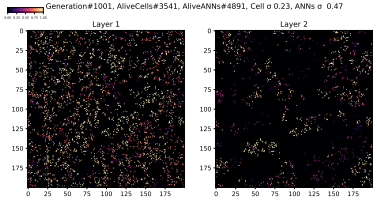
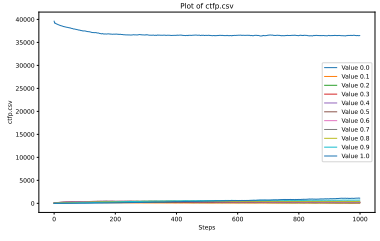
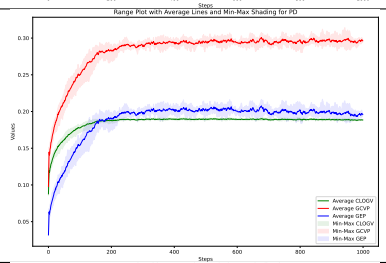
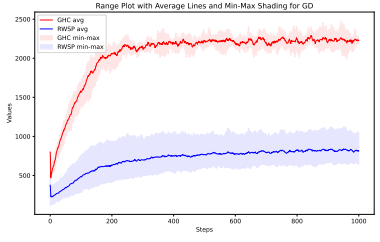
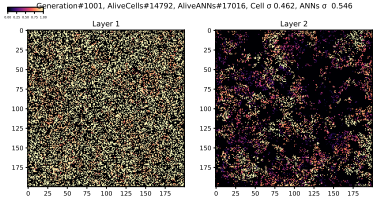
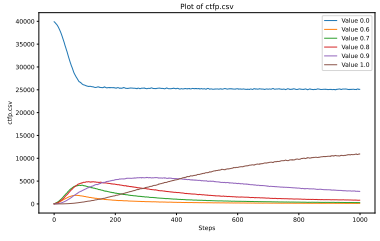
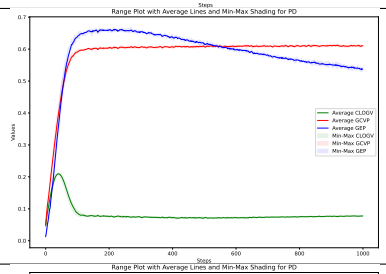
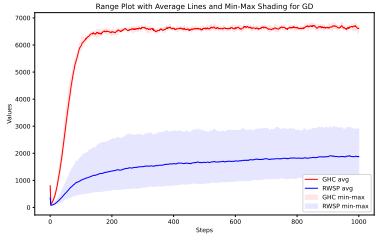
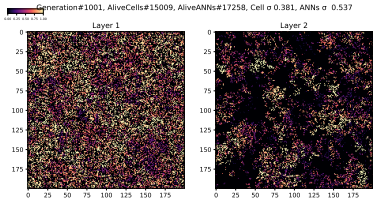
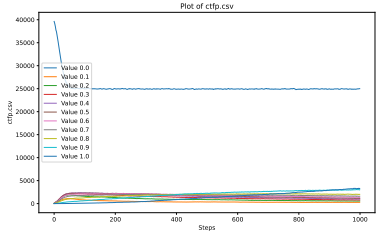
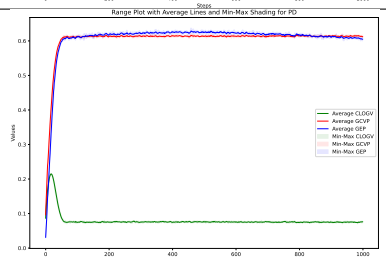
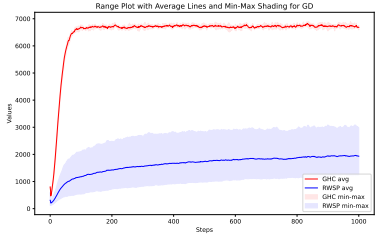
Exp.	NCA	CTFP	PD	GD
1				
2				
3				
4				

Table 5.1: NCA, Speciation and Diversity Plots from Experiments 1 to 4. For activation *tanh* the global entropy (and hence diversity) is still preserved for large generations which can be reflected in NCA biome. In third experiment it is clearly visible how one specie has dominated the biome as shown in CTFP and corresponding entropy falls. For fourth experiment almost cell variance and entropy gets equivalent representing median cells are constantly changing with respect to the current cell.

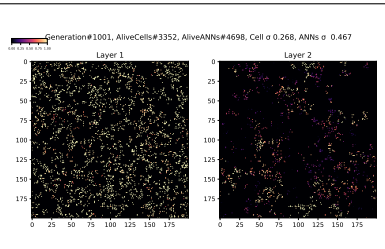
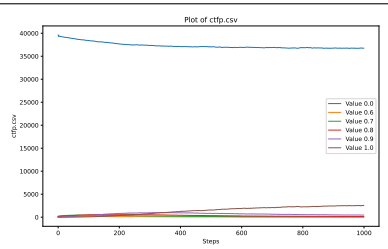
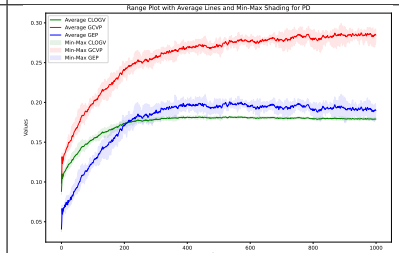
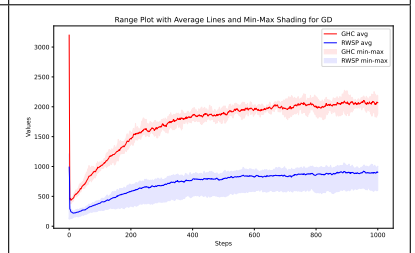
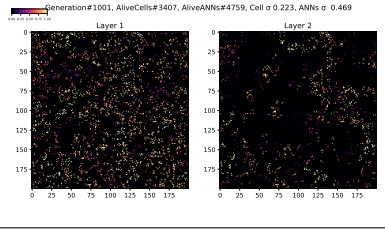
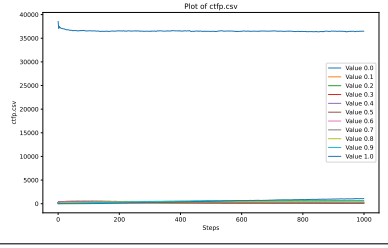
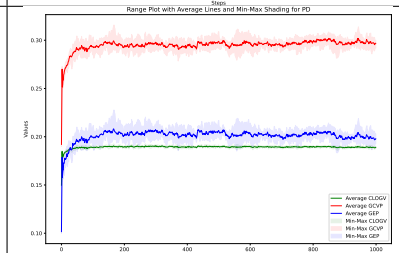
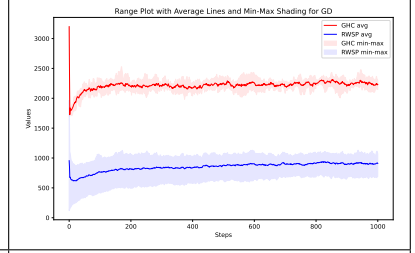
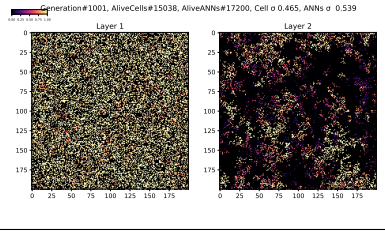
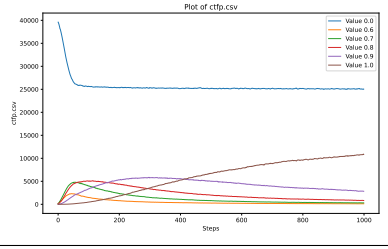
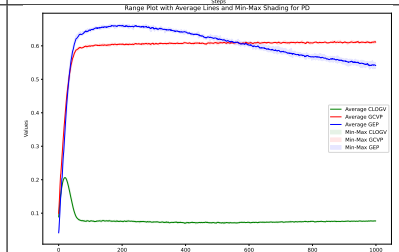
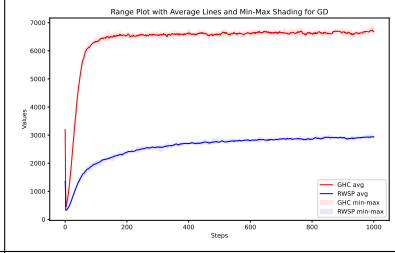
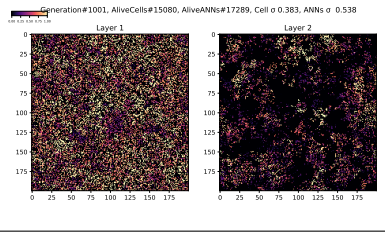
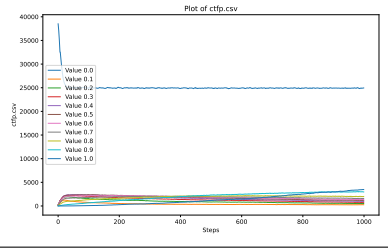
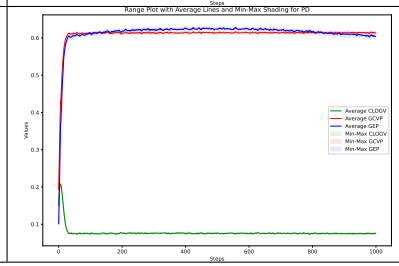
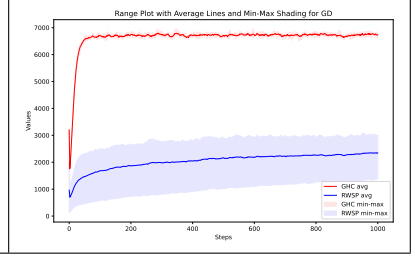
Exp.	NCA	CTFP	PD	GD
5				
6				
7				
8				

Table 5.2: NCA, Speciation and Diversity Plots from Experiments 5 to 8. It is interesting to see the second channel (energy and matter composition) in NCA. All CTFP plots resemble that no NCA was successfully able to cover fully by agents, and hence the empty space persists till last generation. PD results are almost similar to one to four experiments, but GD plots suggest that experiments seven and eight has preserved much higher diversity for larger generations as can be seen in GHC count plot.

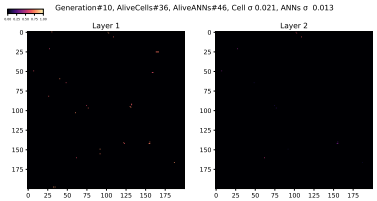
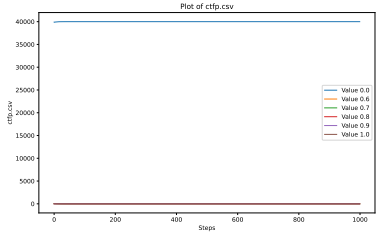
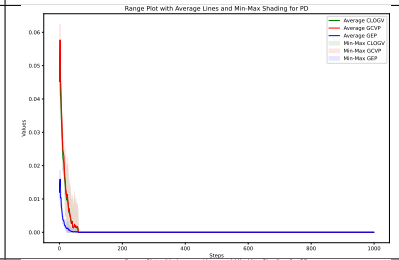
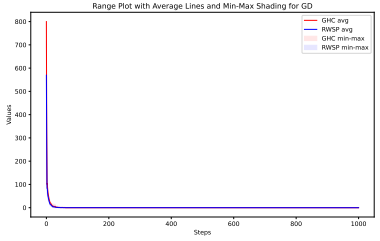
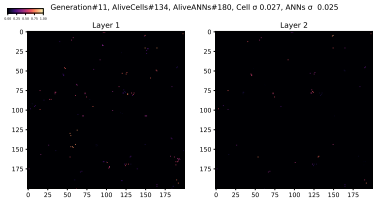
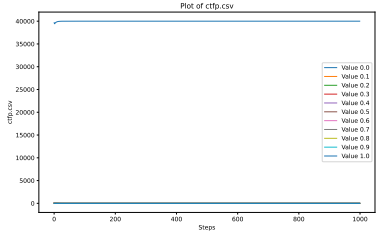
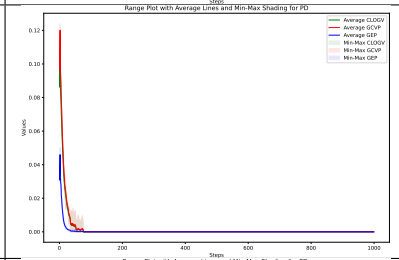
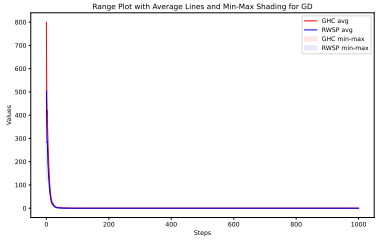
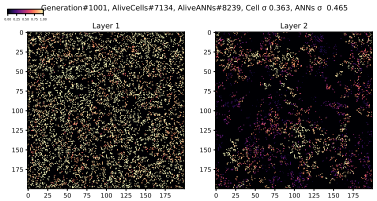
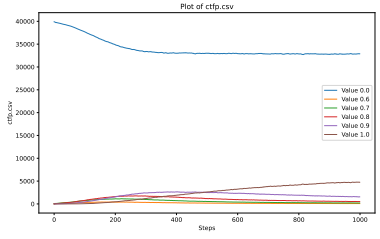
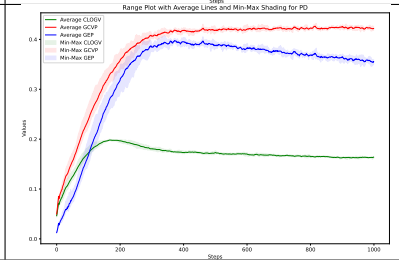
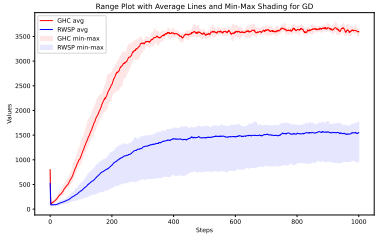
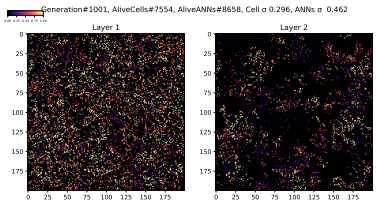
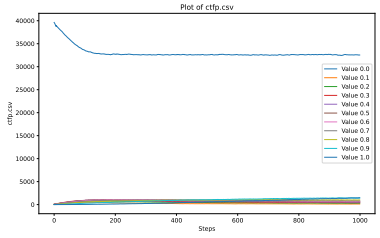
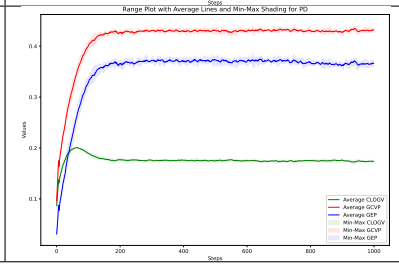
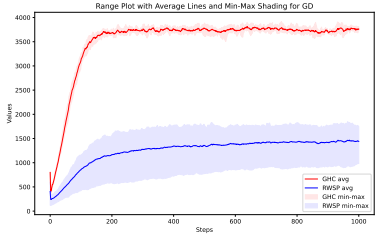
Exp.	NCA	CTFP	PD	GD
9D				
10D				
11				
12				

Table 5.3: NCA, Speciation and Diversity Plots from Experiments 9 to 12 (D ahead of experiment number resembles dead experiment). Experiments nine and ten are dead, that means NCA dies in early generations thus resulting all corresponding plots to zero. For eleven, it can cl;early be seen how GCVP gives a deception of higher diversity while GEP is down-trending. Conversely the case is different for twelfth experiment, where the median changes rapidly resulting higher contributions to diversity (the *tanh* activation has more species range).

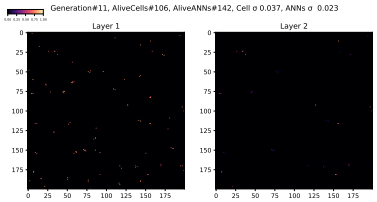
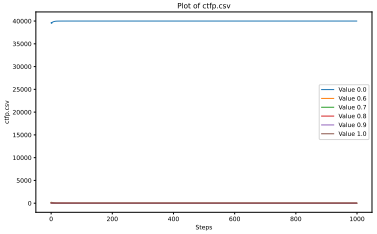
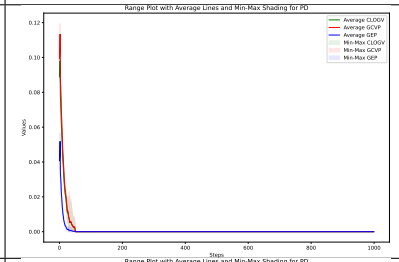
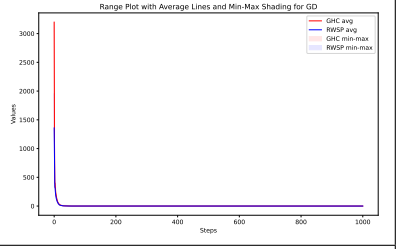
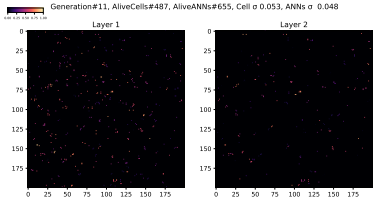
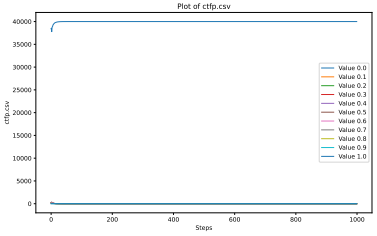
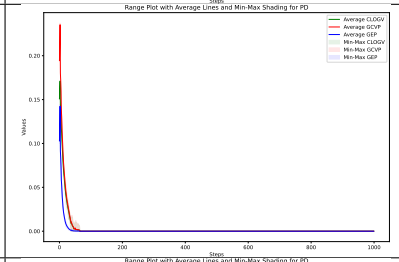
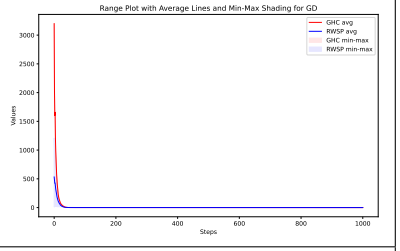
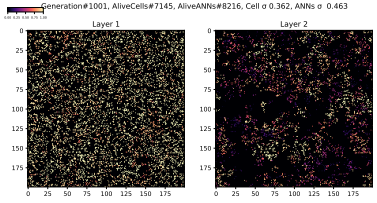
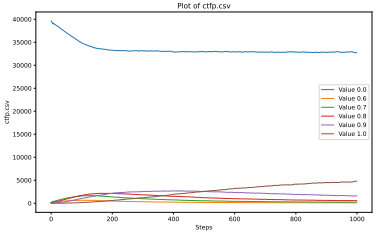
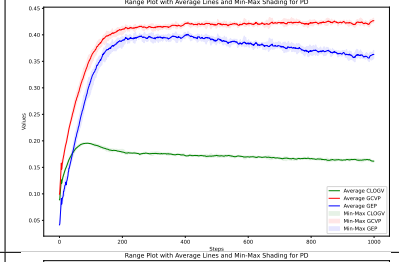
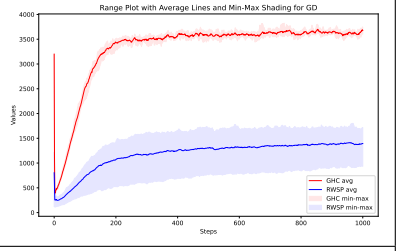
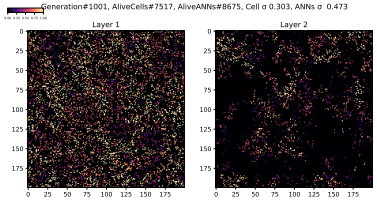
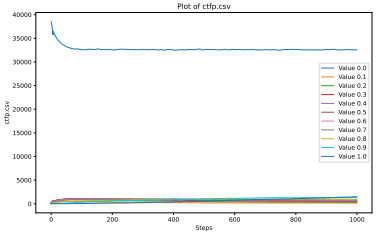
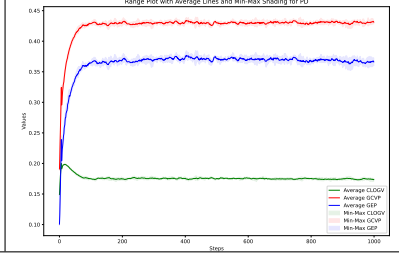
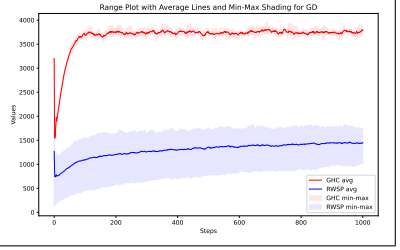
Exp.	NCA	CTFP	PD	GD
13D				
14D				
15				
16				

Table 5.4: NCA, Speciation and Diversity Plots from Experiments 13 to 16 (D ahead of experiment number resembles dead experiment). Thirteen and fourteen are dead experiments. PD and GD tools for fifteen and sixteen almost show similar dynamics as of eleventh and twelfth.

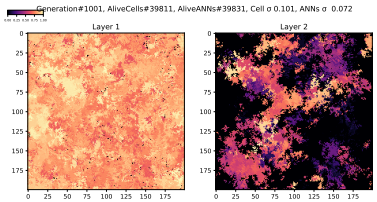
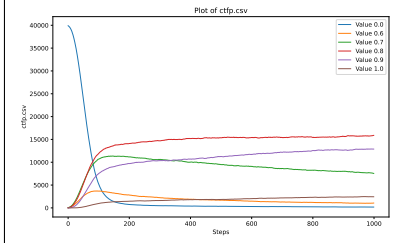
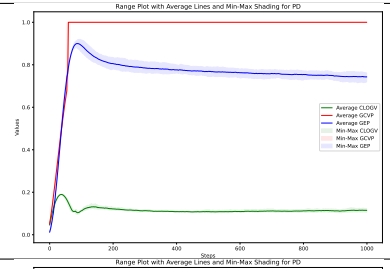
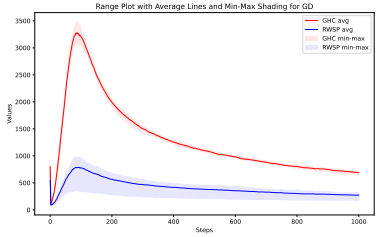
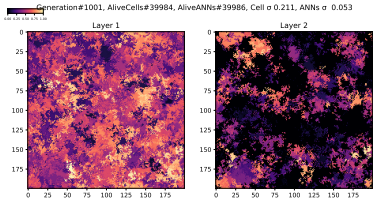
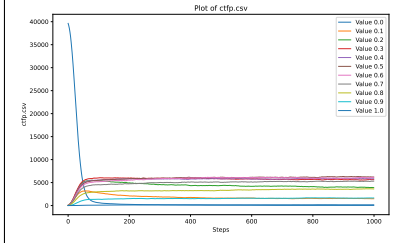
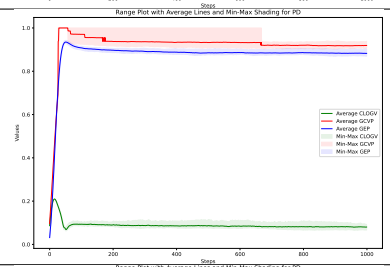
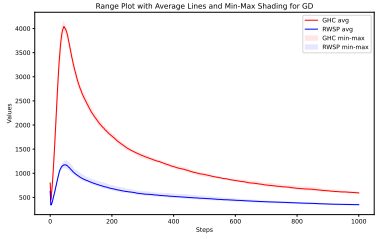
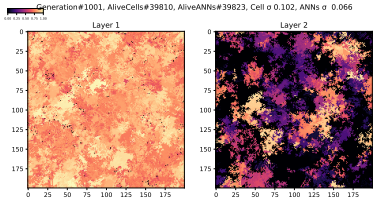
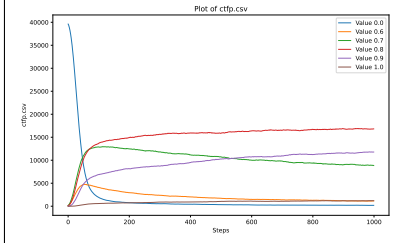
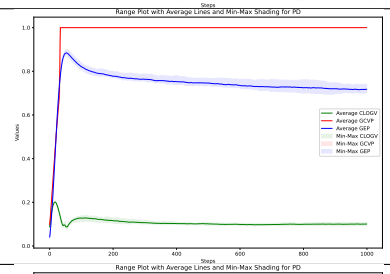
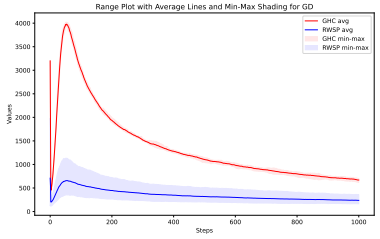
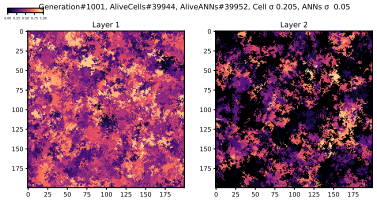
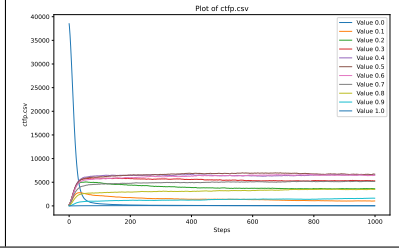
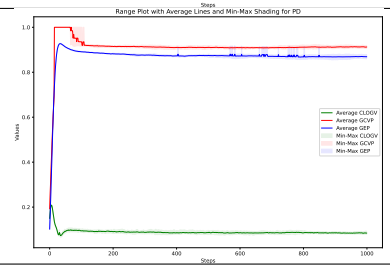
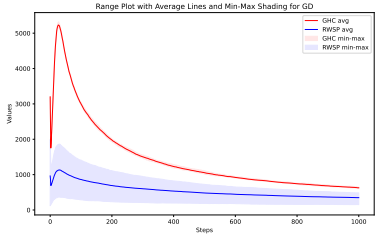
Exp.	NCA	CTFP	PD	GD
17				
18				
19				
20				

Table 5.5: NCA, Speciation and Diversity Plots from Experiments 17 to 20. We are letting these systems grow without external budget. This way they capture complete NCA biome for development and exploration. The difference is clearly visible between *sigmoid* (17, 19) and *tanh* (18, 20) activations. It can be seen very easily in CTFP plots that diversity is very well preserved for species as different. NCA second channel also looks interesting how agents have collaborated to create colonies. In PD tools, specifically GCVP, it stays to unity mostly because as soon the NCA biome is filled with cells, it is high chance (since the cells are not dying) that median and current cell that is compared stays different for almost all generations, and hence resulting zero when performed Kronecker Delta.

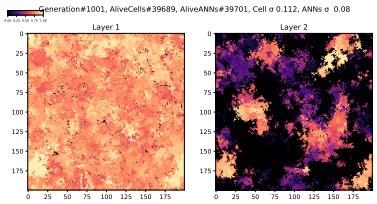
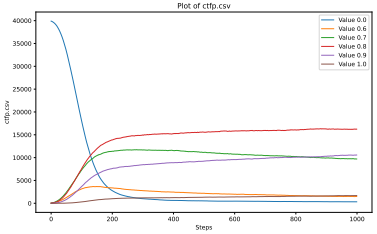
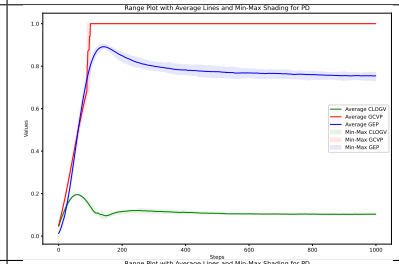
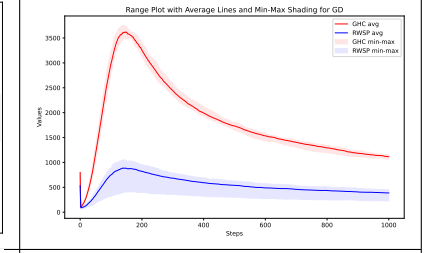
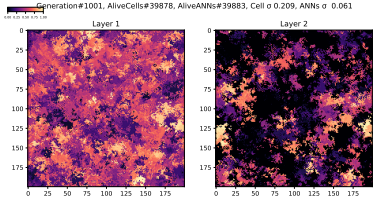
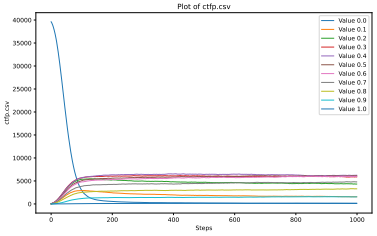
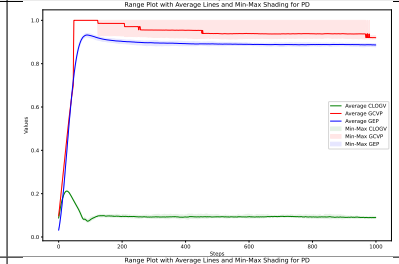
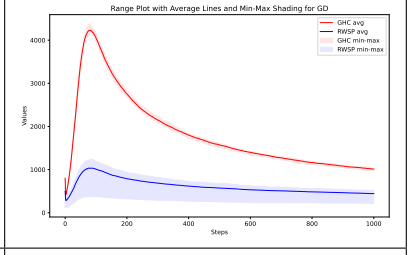
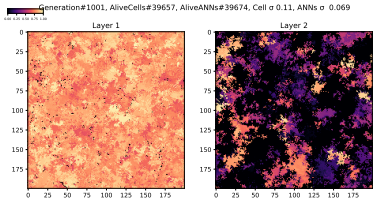
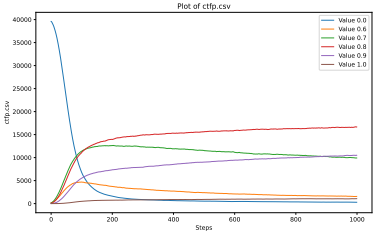
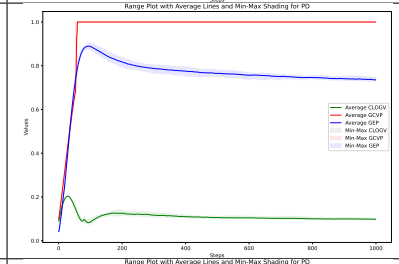
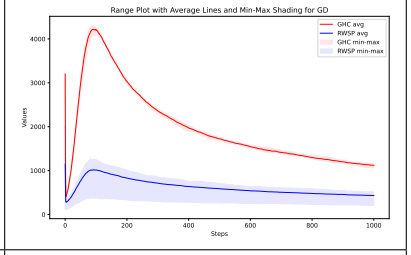
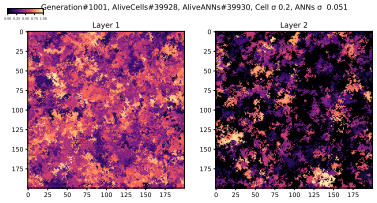
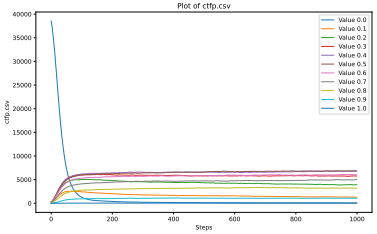
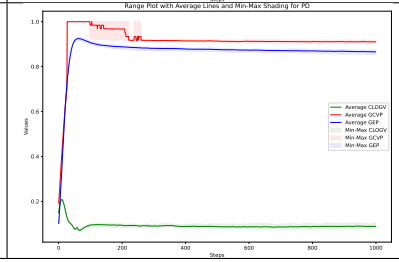
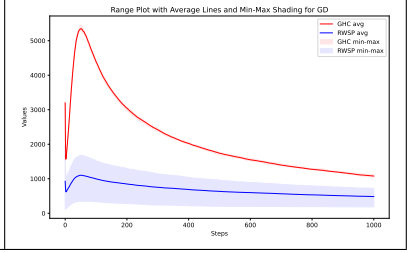
Exp.	NCA	CTFP	PD	GD
21				
22				
23				
24				

Table 5.6: NCA, Speciation and Diversity Plots from Experiments 21 to 24. In these experiments we se similar dynamics as we saw in experiments seventeen to twenty. The behaviour of GD tools is rapid growth and rapid fall (specially GHC). It can be explained by a fact that as NCA is allowed to grow without any external death, it happens very few times that any existing agent dies out of the α threshold. Therefor, we don't see very less new agents forming after the NCA is filled. Because there are no new replications and inheritance happening, no new genetic material is being formed and hence the uniqueness of the color count dampens after few generations when the NCA is full till no empty space.

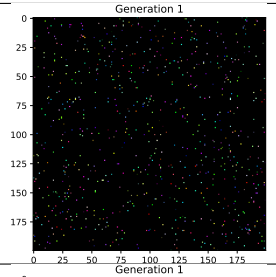
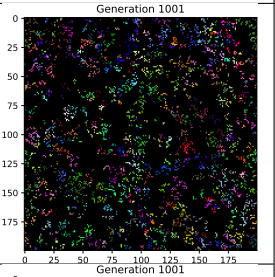
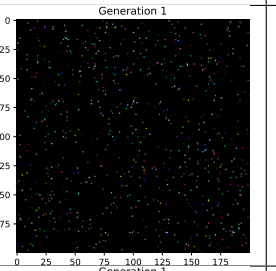
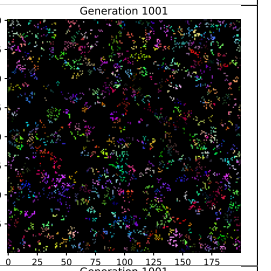
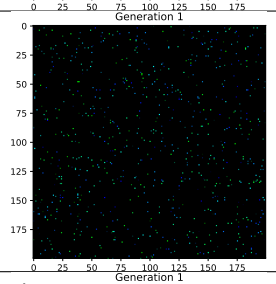
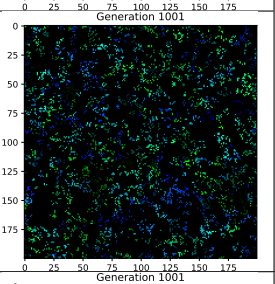
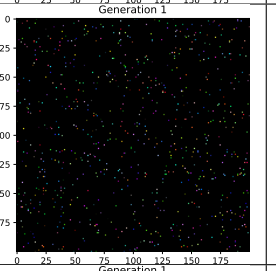
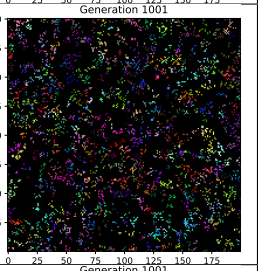
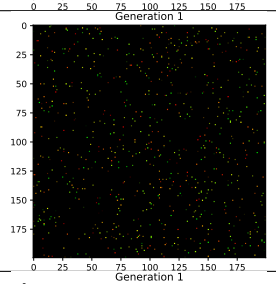
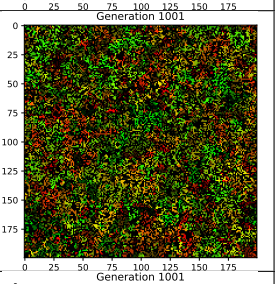
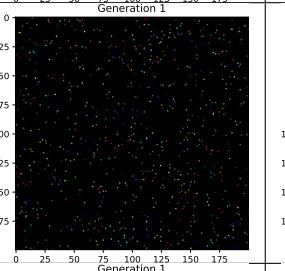
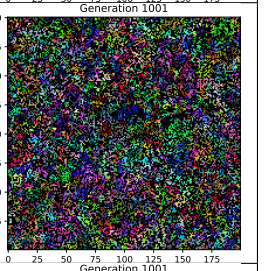
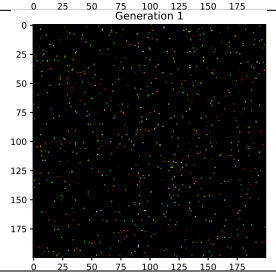
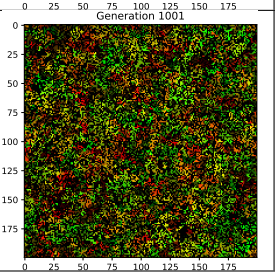
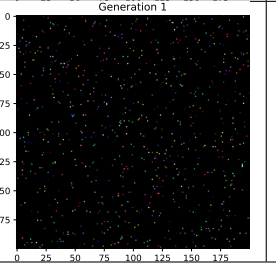
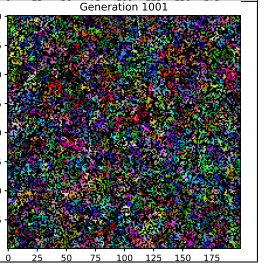
Exp.	RWSP step 1	RWSP step 1000	GHC step 1	GHC step 1000
1				
2				
3				
4				

Table 5.7: RWSP and GHC Visuals from Experiments 1 to 4. Experiment one is still able to preserve heterogeneity in RWSP and GHC, but for two, three and four, RWSP becomes homogeneous as no significant changes happening in the three gene trajectory while lot more heterogeneity can be seen in GHC.

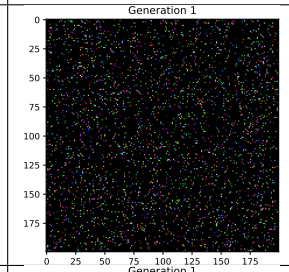
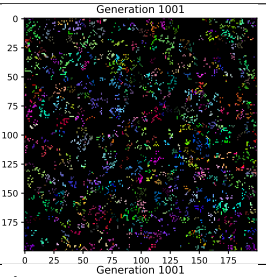
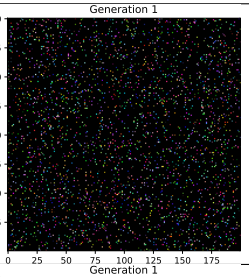
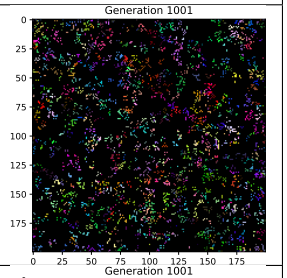
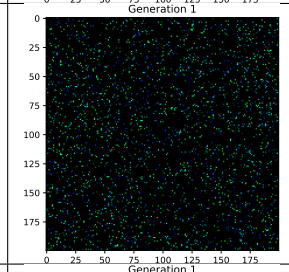
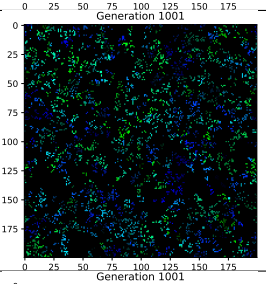
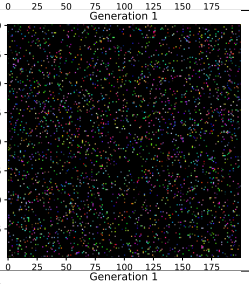
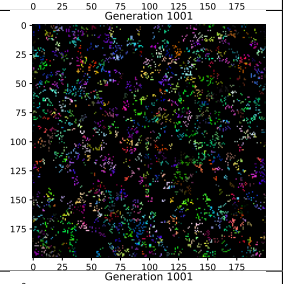
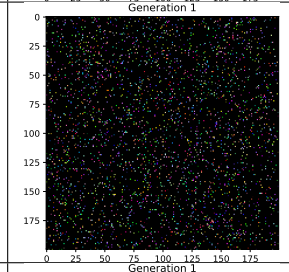
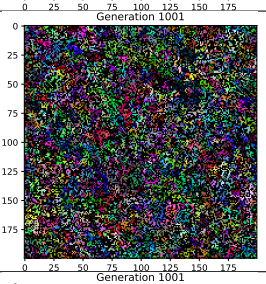
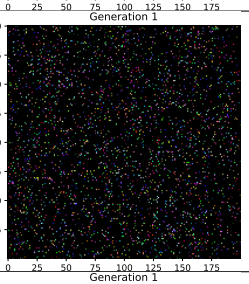
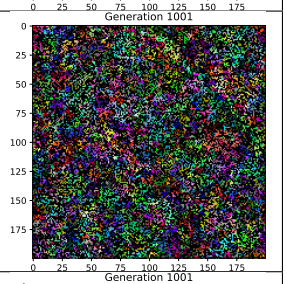
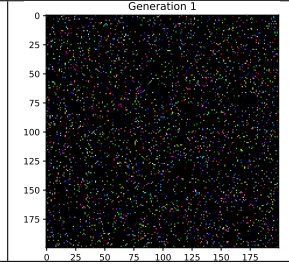
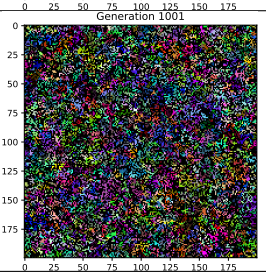
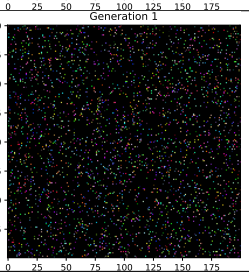
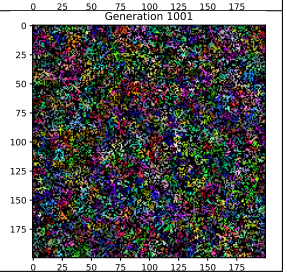
Exp.	RWSP step 1	RWSP step 1000	GHC step 1	GHC step 1000
5				
6				
7				
8				

Table 5.8: RWSP and GHC Visuals from Experiments 5 to 8. Experiments five, seven and eight are still able to preserve heterogeneity in RWSP and GHC, but for six, RWSP becomes homogeneous as no significant changes happening in the three gene trajectory while lot more heterogeneity can be seen in GHC.

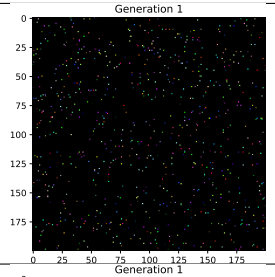
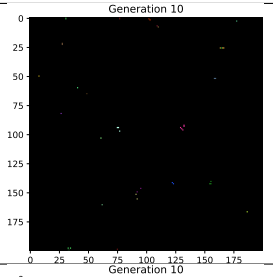
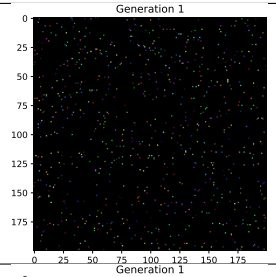
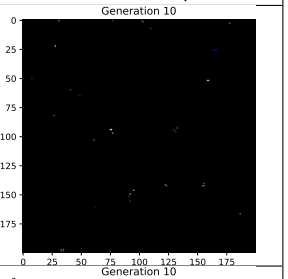
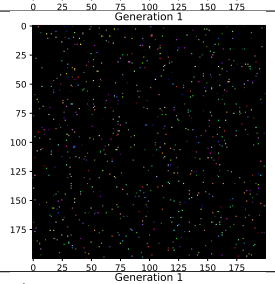
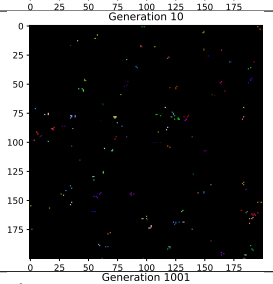
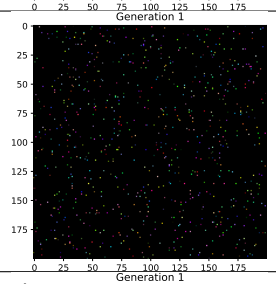
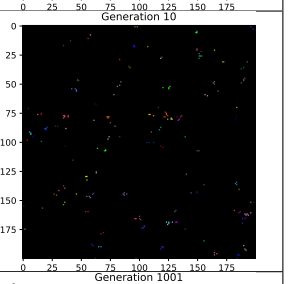
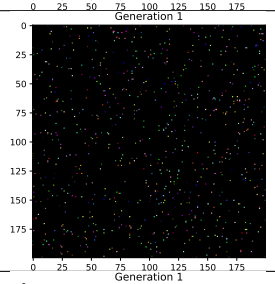
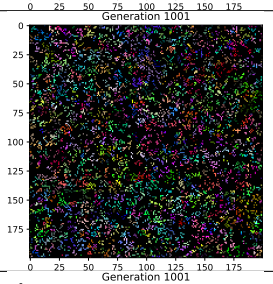
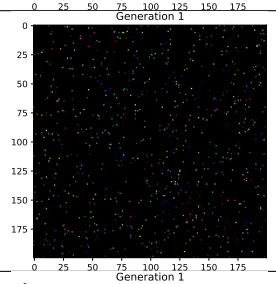
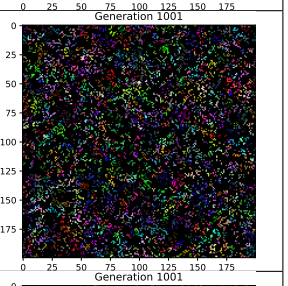
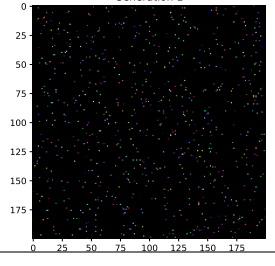
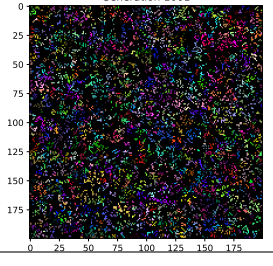
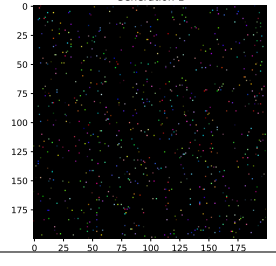
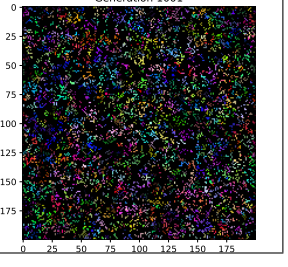
Exp.	RWSP step 1	RWSP step 10/1000	GHC step 1	GHC step 10/1000
9D				
10D				
11				
12				

Table 5.9: RWSP and GHC Visuals from Experiments 9 to 12. (D ahead of experiment number resembles dead experiment). Nine and tenth are dead experiments and hence to visualise tenth generation is shown. While for eleven and twelfth, last generation is shown. Both RWSP and GHC shows large heterogeneity.

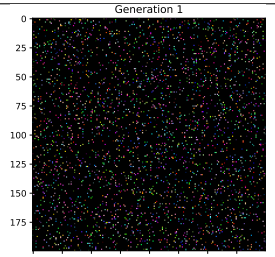
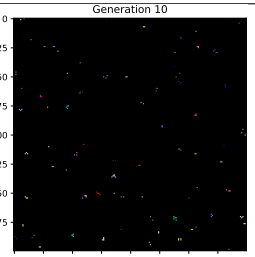
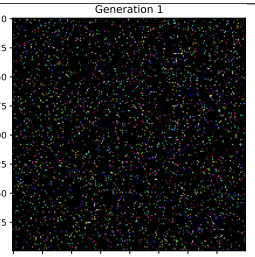
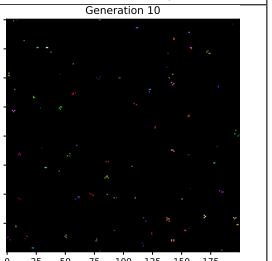
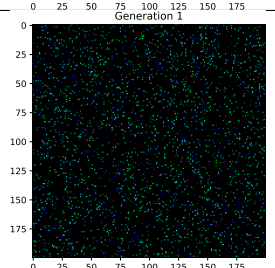
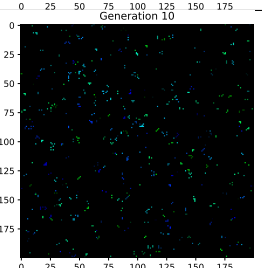
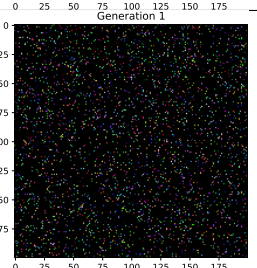
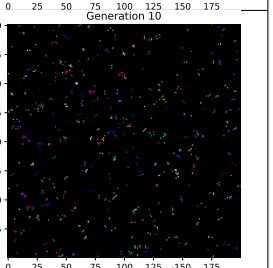
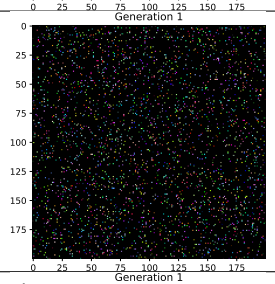
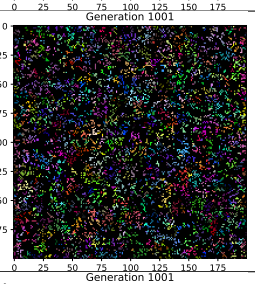
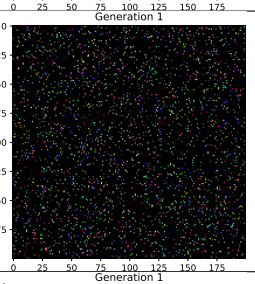
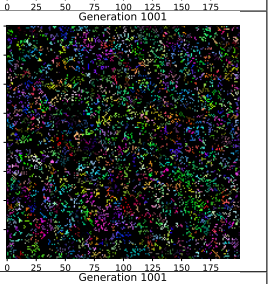
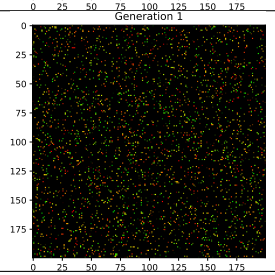
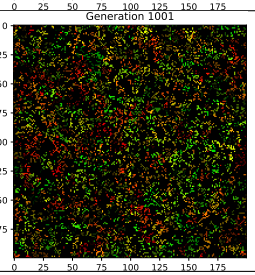
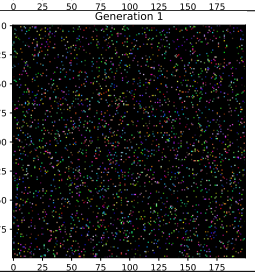
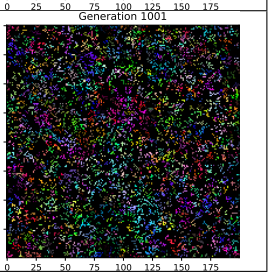
Exp.	RWSP step 1	RWSP step 10/1000	GHC step 1	GHC step 10/1000
13D				
14D				
15				
16				

Table 5.10: RWSP and GHC Visuals from Experiments 13 to 16. (D ahead of experiment number resembles dead experiment). For experiments fifteen, even after significant perturbation rate, RWSP shows high homogeneity, which validates the usability of GHC plot.

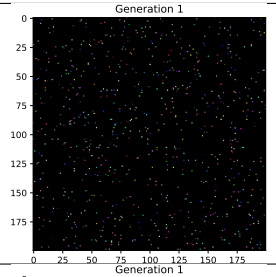
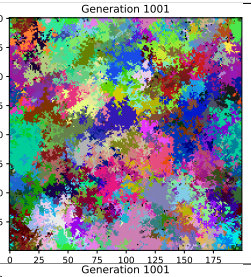
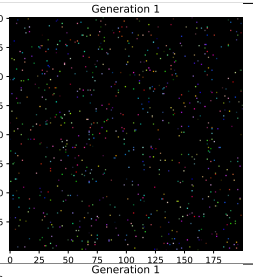
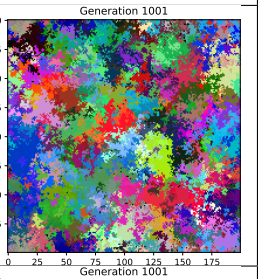
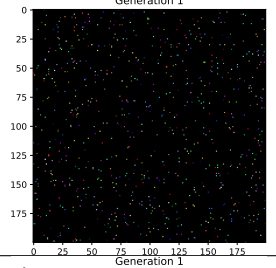
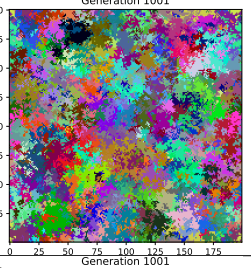
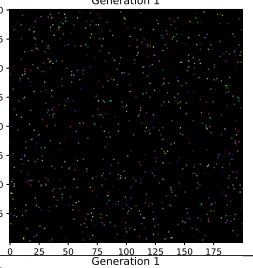
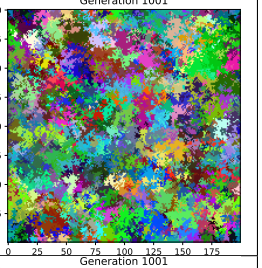
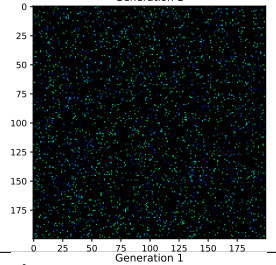
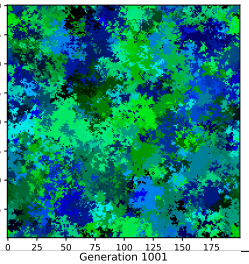
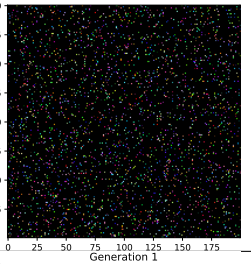
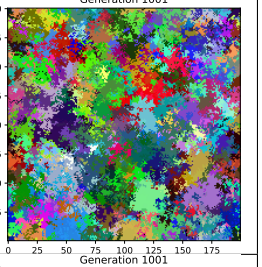
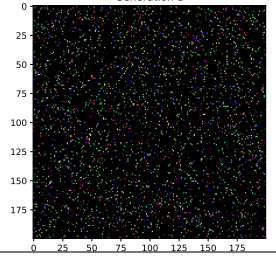
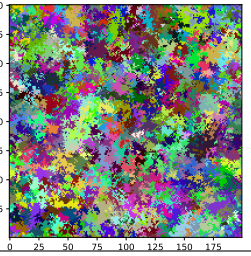
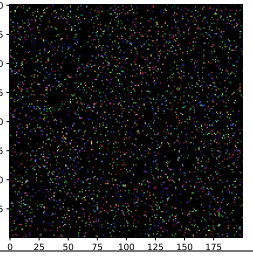
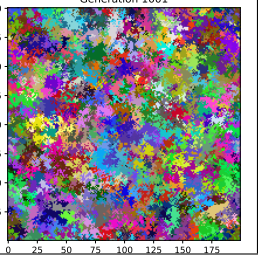
Exp.	RWSP step 1	RWSP step 1000	GHC step 1	GHC step 1000
17				
18				
19				
20				

Table 5.11: RWSP and GHC Visuals from Experiments 17 to 20. Experiments 17, 18 and 20 are best example of forming groups of same genotype as colonies result of self replication and inheritance with local interactions. RWSP plot shows bigger colonies as it only tracks three genes from the pool and it is high chance that those three genes are not mutated at the time of inheritance. Where are diversity can be easily seen in GHC.

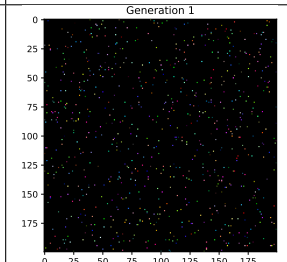
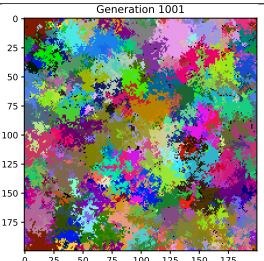
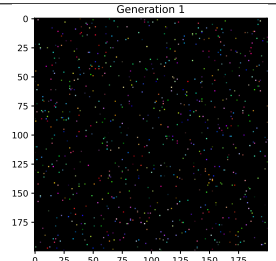
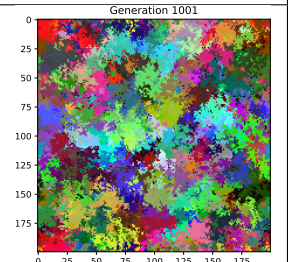
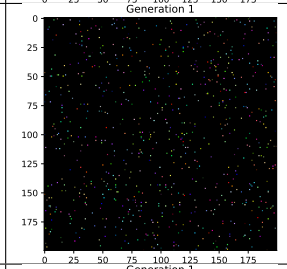
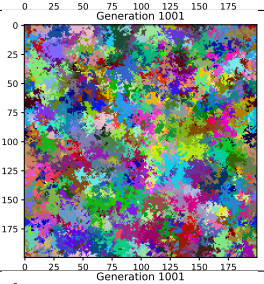
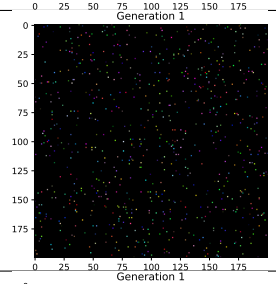
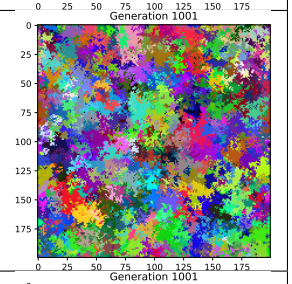
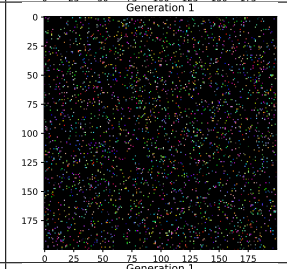
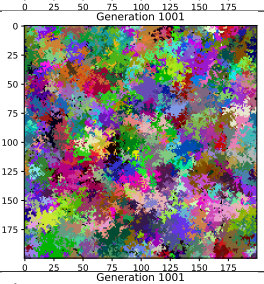
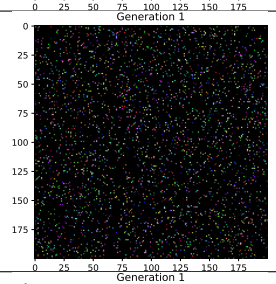
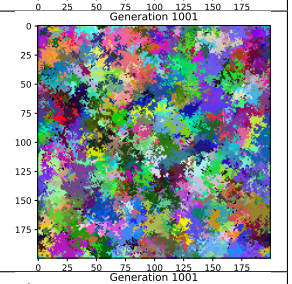
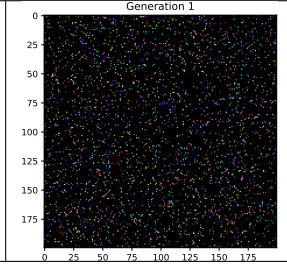
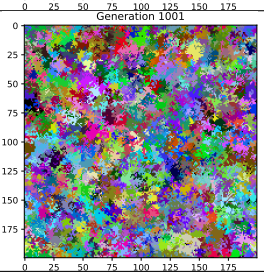
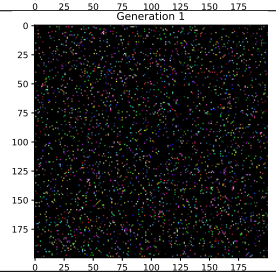
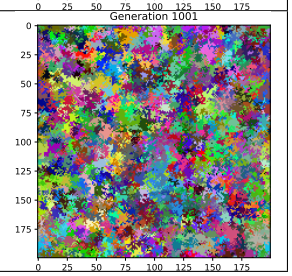
Exp.	RWSP step 1	RWSP step 1000	GHC step 1	GHC step 1000
21				
22				
23				
24				

Table 5.12: RWSP and GHC Visuals from Experiments 21 to 24. RWSP is lot more homogeneous until three fixed gene changes and hence bigger colonies, but GHC is very sensitive to mutations and this lot more heterogeneity.

Chapter 6

Discussion

In this chapter we discuss and relate results, research questions, how we reached to goal, how the proposed Neural CA actively simulates the life-like cellular behavior in a biome of cellular organisms. We also cherry-pick some of the results to prove and explain how the proposed complex system simulates species dying, surviving, extinction, invading. For example, how one specie only overlapped all over the evolved NCA which helps to understand how specific genes and traits enable them to reproduce and not allowing other species to grow. In a narrow sense, we promote explainability why some clusters tend to outline the communal growth for example, species trying to separate themselves from other genealogies and hence communities for their survival and hence a new species evolves completely. Finally we discuss interpretations and resemblances of the tool plots, and the Phenotype-Genotype dependency and predictability.

Overall we discuss proposed methods, results and novelty of this research. We also discuss future use cases of this research towards scientific research in computational biology. First, we discuss the classification of experiments and corresponding results. In this section we discuss small runs to identify *interesting* behaviour of the NCA over different experimental setup. We outline various categories on the basis of NCA behaviour that could be interesting to note before analysing large run experiments. Second, we discuss the large run results from the last chapter on the basis of categorisation of small runs. It is wise to note here that small run or large run experiments are also discussed on the verge of statistical interpretations in their respective sections with the help of hand-picked experimental results. Third, we outline the alternative method results which can be starting point for future research. Chapter 3 also contains proposed alternative methods. Fourth, we discuss the applications and future usability of this research beyond this thesis where it can yield meaningful results in real world setting. Finally, we place all rest of the discussions including metaphorical ideas in miscellaneous discussions in last section. We also keep referring tables from previous chapter for large run results.

6.1 Classification of Results

In this section we classify the results of small runs on the basis of their Phenotypic and Genotypic behaviours and strongly believe that the dynamics would show similar trends for large runs (as we also verify in chapter 5). Therefor all analysis that we do for small runs in this section, also applies on large runs. Here the NCA grid is relatively small that is evolved for few generations. In this section, we will use PD and GD tools that we

have devised in this thesis to study behaviour of different configurations of the evolved NCA. Specifically, we use CTFP, GEP (H_t), GCVP (σ_{gross}), CLOGV (σ_{glob}) and Unique Color Count Plot (from RWSP and GHC) tools to categorise experiments. This analysis help us to visualise the frequency based coarse grained analysis of the evolved NCA thus capturing the interestingness of the corresponding NCA. While we have configuration details presented in Appendix C for each of the results in this section, we only focus on the categorisation of similar behaviour of the NCAs. Please refer Appendix C for corresponding configuration for these experiments and results (Please refer 6.1 for quick reference to full forms).

Abbreviation	**Full Form**
PD	Phenotypic Diversity
GD	Genotypic Diversity
CTFP	Cellular Type Frequency Plot
GEP	Global Entropy Plot
GCVP	Gross Cell Value Plot
CLOGV	Cell Local Organisation Global Variance
RWSP	Random Weight Selection Plot
GHC	Genotypic Hash Coloring
RWSP	Random Weight Selection Plot
CNWA	Clustering Neural Weights Approach

Table 6.1: List of abbreviations recently used

6.1.1 Coexistence of Species

In Figure 6.2, species manage to survive together and still there is lot of empty space in the NCA to be consumed by the species. Plots show that species have managed to coexist together with empty space in an equilibrium manner. This also means that species emerge and die in a ratio such that it maintains diversity, their group count and empty space.

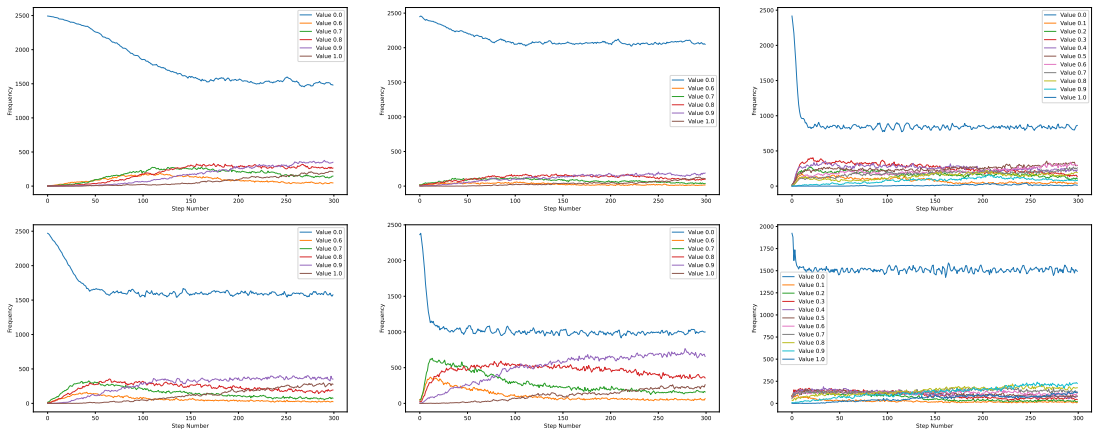
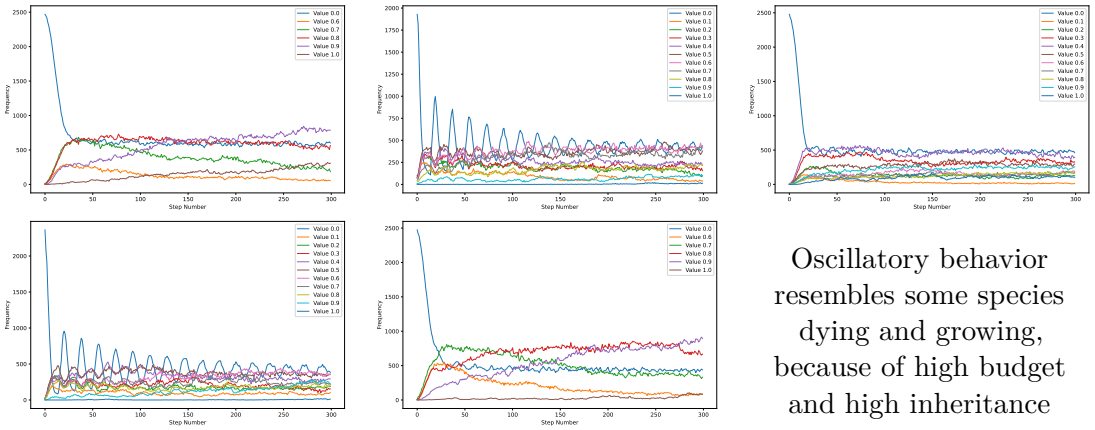


Figure 6.1: CTFP plots for coexistence of Species.

6.1.2 Species Consumes All Space to Coexist (Low PPP)

In Figure 6.2, it can also be thought as, species compete for space to coexist. This is because, there is less number of void in the NCA, and count of some species are more than null, that means the ones that has crossed nullity count are more successful in competence of space managing to propagate.

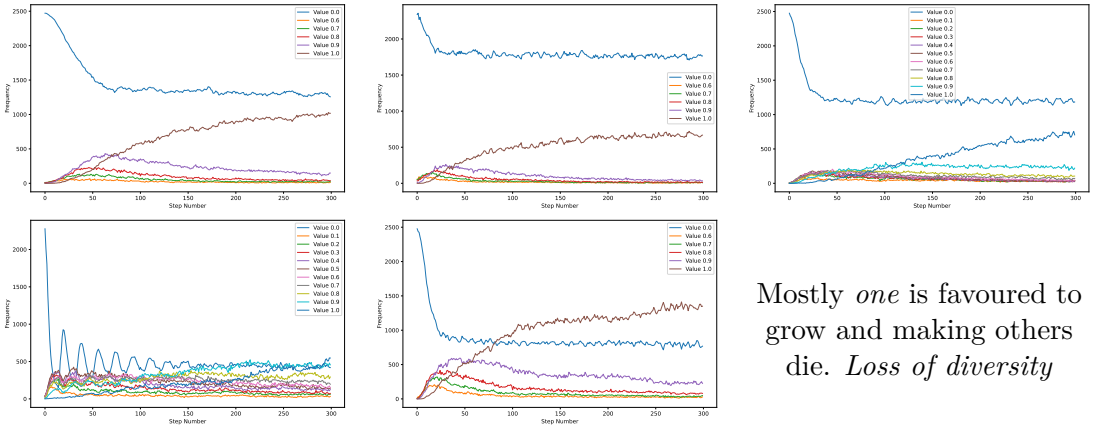


Oscillatory behavior resembles some species dying and growing, because of high budget and high inheritance

Figure 6.2: CTFP plots for species consumes all space to coexist.

6.1.3 One Specie Dominate

In Figure 6.3, the CTFP plots show all species are getting dominate by one specie.



Mostly *one* is favoured to grow and making others die. *Loss of diversity*

Figure 6.3: CTFP plots for one specie dominate.

6.1.4 High Parameter Perturbation Probability (PPP)

In Figure 6.4, the CTFP plots show the effect of high PPP on NCA grid. It is interesting to see how after few generations all species are pseudo-dead and only one specie comes to existence which for some cases also further crosses the empty space of the grid.

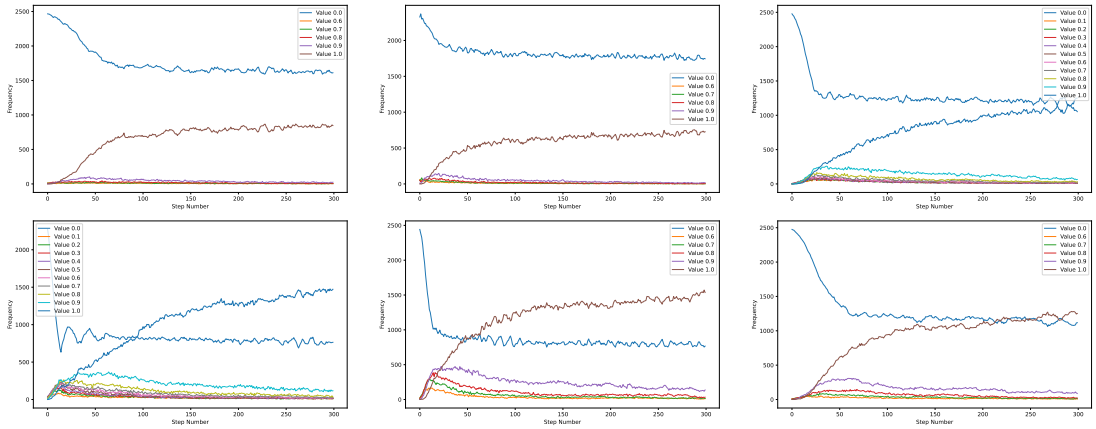


Figure 6.4: CTFP plots for high Parameter Perturbation Probability.

6.1.5 Effect of Low PPP on Phenotypic Diversity

In Figures 6.5 and 6.6, GEP shows a trend and continue to stabilize. This means entropy/randomness is prominent for further generations in terms of phenotype. It could also mean that diversity is not lost at those points in further generations. It can be observed with NCA as well. We checked each of the corresponding NCAs of the plots we showed and confirmed visually that diversity is not getting lost. This can also be confirmed with corresponding CTFP plots, which we confirmed as well. But to elaborate we are showing it in Figure 6.6. If the species have equal dominance (where only one specie is not overriding then we can claim such thing) then only we see such plot for GEP Thus verifying and explain GEP in the combined plot and our claims. See how it becomes explainable when we compare plots in both the figures. In simpler words, and concluding it, we are only seeing GEP right now and trying to show how it measures diversity of NCA. We claim, if the GEP plot stays stable or oscillates within a small range that means we are successfully able to keep the diversity in NCA species. Had it been GEP was down trending, we could have assumed that diversity is getting lost and it could also be verified in the corresponding CTFP plots. GCVP plot shows similar trend keeping average cell variance corresponding to median cell showing overall diversity remains stable compared with median. But that does not mean NCA is diverse enough. GCVP might not be considered a good measure, as even if it sees a different median, it will tell variation is enough and thus gives a deception of diversity. It could be useful for binary systems where CA is just 0-1 valued. We will see this in further plots. A high, down and then stable CLOGV suggests that local variation increases too much with growth phase, but then drops because the NCA is filled, and local neighborhoods are also filled. And then stabilizes because no new species are getting born and local neighborhoods are also filled with the existing species.

6.1.6 Effect of Medium PPP on Phenotypic Diversity

In Figure 6.7, GEP shows downtrend, which means diversity is getting lost. We need to find a balance of PPP such that we get diverse phenotype as we saw in PPP of 0.02 case in previous Figures. This can also be confirmed visually seeing NCA or CTFP plot. CTFP plots look like plots from Figure 6.3. We rather seek plots like in Figures 6.1 and 6.2. Because their corresponding variation plots would also look better. We still see GCVP

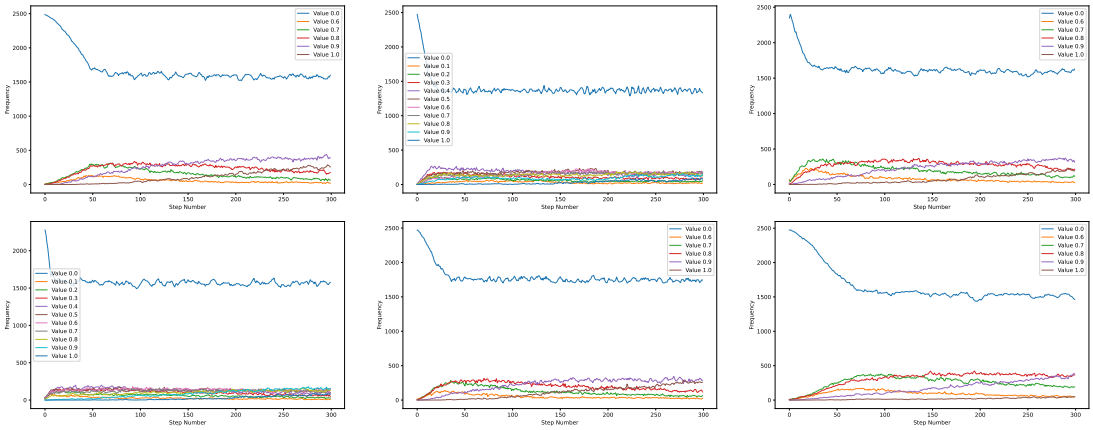


Figure 6.5: CTFP plots for effect of low PPP on Phenotypic Diversity.

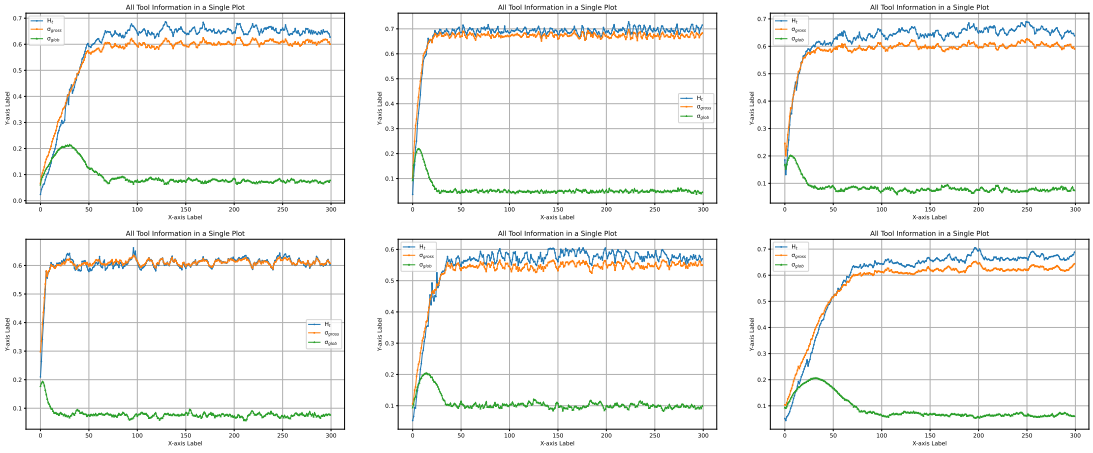


Figure 6.6: PD Tools 2,3 and 4 plots for effect of low PPP on Phenotypic Diversity.

showing better trends but it's a deception as it is considering median variation. Considering cells dying and reborn, it could be working fine, as we just saw in previous section, this could have performed well even if the cell values are 0-1 in CA. So this might not be a better tool to measure diversity of species.

6.1.7 Effect of Medium PPP and Slow Inheritance on Phenotypic Diversity

Most of the configurations with low budget and slow inheritance shows similar behavior in Figure 6.8. These NCAs are interesting that shows small glider like patterns dying and getting formed but does not over-populate to completely capture the NCA biome.

6.1.8 Effect of High PPP on Phenotypic Diversity

Interesting to see plots in Figure 6.9, where GEP is downtrend to GCVP because, it will be very less diverse along with that one specie would be dominant in CTFP. GEP should never be smaller to GCVP. All it means is that, a system which is highly diverse towards median (a specific cell value) but is very less diverse globally (or in terms of randomness

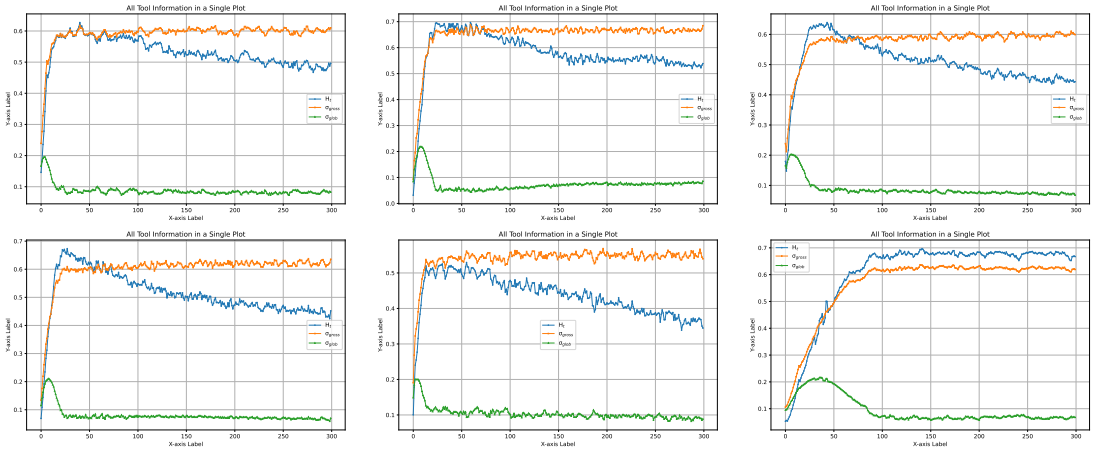


Figure 6.7: PD Tools 2,3 and 4 plots for effect of medium PPP on Phenotypic Diversity.

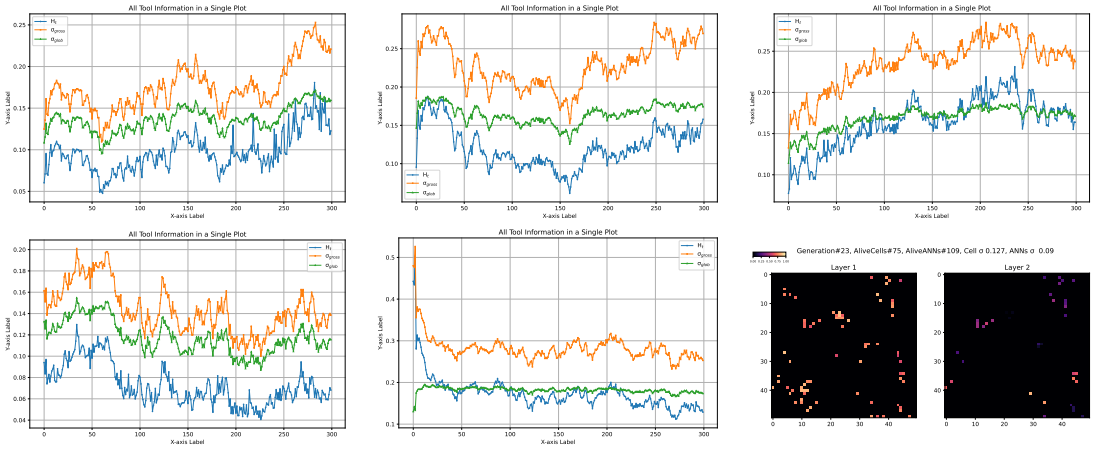


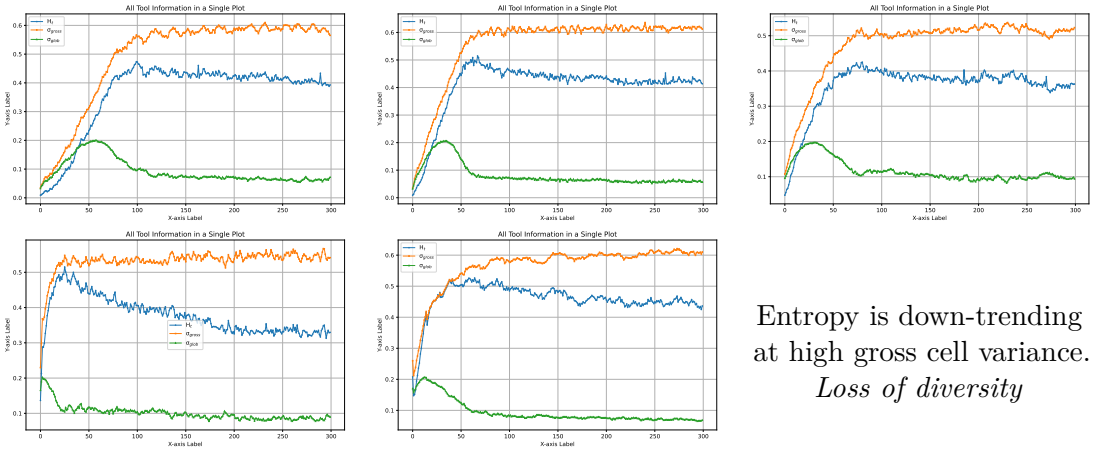
Figure 6.8: Tools 2, 3 and 4 to check effect of medium PPP and slow inheritance on Phenotypic Diversity. There is also one image of corresponding NCA plot corresponding first PD plot to show glider like behaviour.

and hence diversity of species!). When checked visually, the NCAs are concentrated with all cell values as one specie value.

In the following subsections we only see Genotypic Diversity Combined Plot and corresponding analysis. The unique count plot that counts the number of unique colors of RWSP and GHC over generations.

6.1.9 Effect of Inheritance Probability on Genotypic Diversity

In Figure 6.10, as inheritance probability increases, the gap between GHC and RWSP increase. Why? Keeping rest configuration same. Or we can say the diversity increases. Because as we increase the inheritance probability but keeping the PPP small, it still has higher chance of promoting mutation as whenever inheritance happens, PPP happens. We see balance of both in next subsections. But it can be considered with extreme cases where inheritance probability is low to high. The gap remains significant why? Because we have high inheritance which means weights are copied a lot of times, but perturbation of those



Entropy is down-trending at high gross cell variance. *Loss of diversity*

Figure 6.9: Tools 2, 3 and 4 plots for effect of high PPP on Phenotypic Diversity.

weights happens very small in quantity. So, RWSP is not able to capture many those mutations many times, while GHC is tool which captures overall Genotypic Diversity.

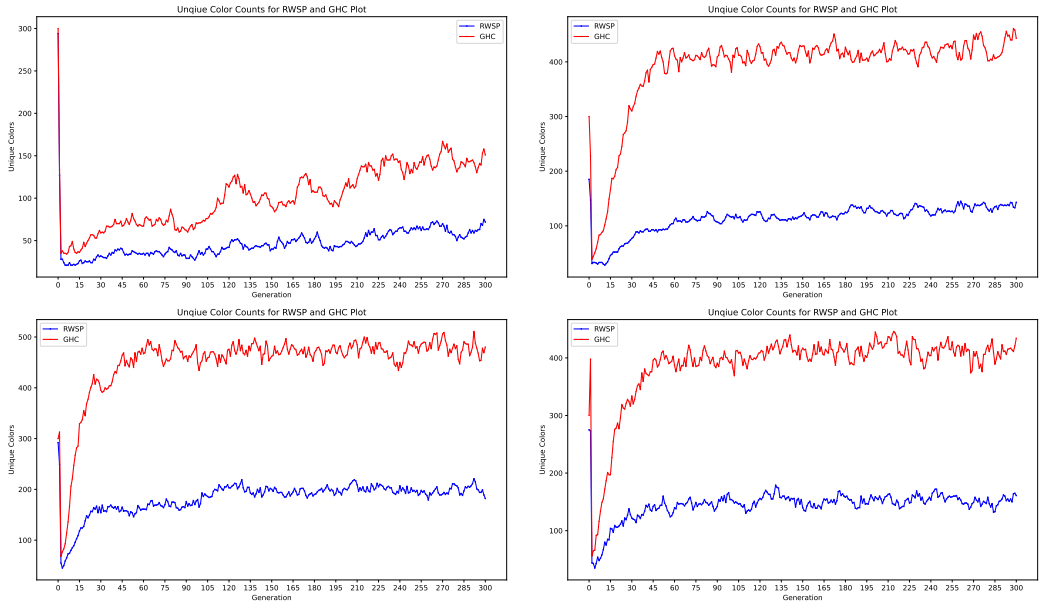


Figure 6.10: Unique Color Count Plots of RWSP and GHC for effect of inheritance probability on Genotypic Diversity (low to high as gap between RWSP and GHC increases).

6.1.10 Effect of PPP on Genotypic Diversity

In Figure 6.11, for a very high PPP, GHC and RWSP almost works same! Because, we have lot more perturbations happening, however the diversity has already reached with medium PPP. In other words, GHC is able to reach the peak diversity at medium PPP while RWSP at very high PPP (check y-axis that represents counts of the plots to compare).

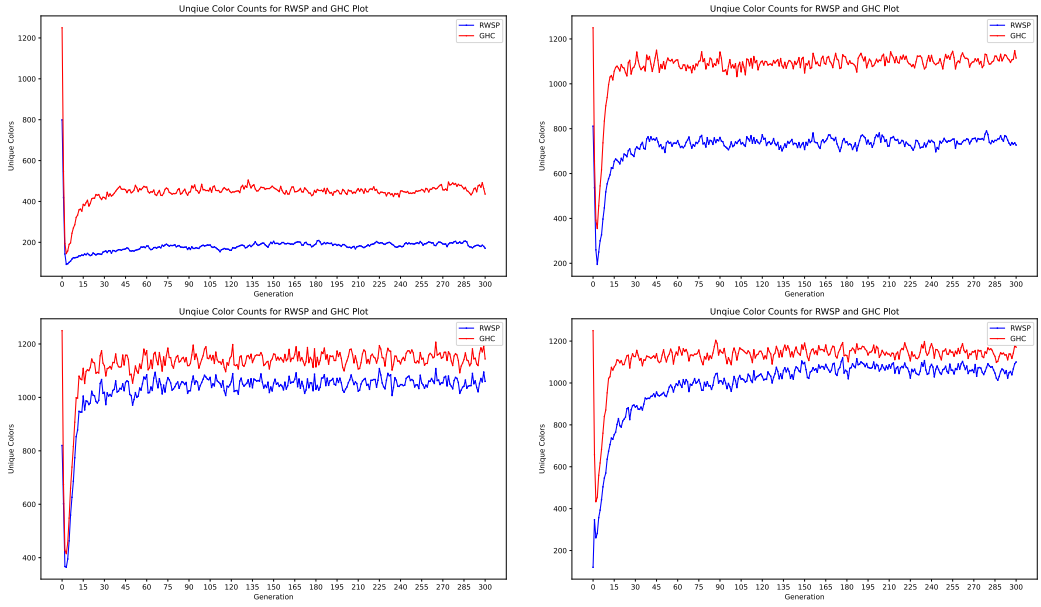


Figure 6.11: Unique Color Count Plots of RWSP and GHC for effect of PPP on Genotypic Diversity (low to high).

6.1.11 Effect of Budget and Inheritance on Genotypic Diversity

In Figure 6.12, we notice interesting behaviour. NCA looses interestingness with (low budget, low inheritance) and (high budget, high inheritance). Interestingness lies in (high budget, low inheritance) and (low budget, high inheritance). Figure 6.12 shows six figures that has low to high budget from left to right and low to high inheritance probability from top to down. Configurations with (low budget, high inheritance) and (high budget, low inheritance) have very high diversity and more unique colors. Configurations with (small budget, small inheritance) and (high budget, high inheritance) have relatively very low diversity and less unique colors.

6.1.12 Effect of High PPP, Initial Probability, High Budget and High Inheritance on Genotypic Diversity

Extreme case of high PPP. In Figure 6.13 plot High ppp is reflected by almost overlap of both the curves meaning diversity is preserved almost similar in both tools. Interestingly there exist a point in some cases where NCA start to lose diversity around middle generations but again increase it in further generations speedily. In other words, if we notice in earlier generations it has slow increase in diversity but with increasing generations, we say it has learned to maintain diversity. Learned to maintain diversity? If we start with some genome, and let NCA grow, most of the cells are inherited in future generations and have been undergone the mutation routine. We claim it can be understood that genomes are propagated in further generations which make them continue to produce diversity over time even if they are dying, its not like all genomes have died (even if there are some chances that all genomes are lost somehow), GD can still be preserved in higher budget as higher chance of inheritance taking place. Therefor we see high diversity with high budget.

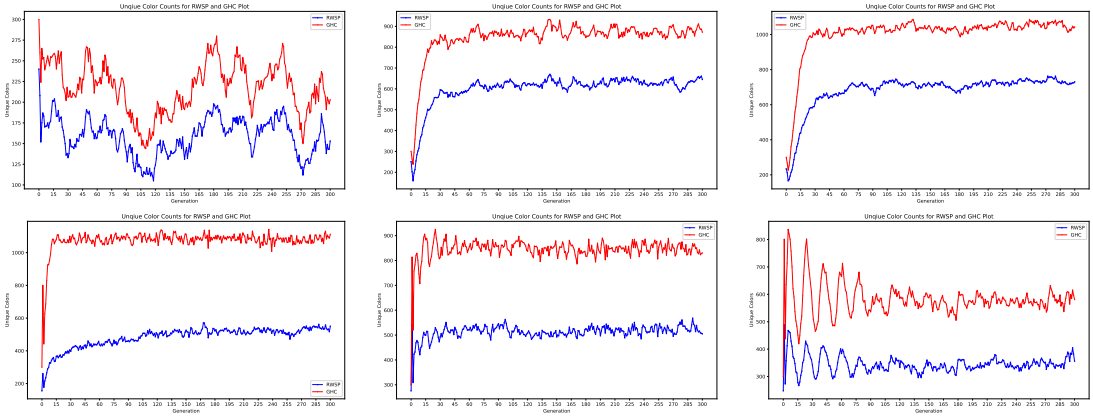


Figure 6.12: Unique Color Count Plots of RWSP and GHC for effect of Budget and Inheritance on Genotypic Diversity (low to high budget left to right, low to high inheritance top to down. So first plot is low-budget & low-inheritance while last plot is high-budget & high-inheritance).

We also see fallacy of RWSP and usefulness of GHC (GHC performs better than RWSP) in Figure 6.14.

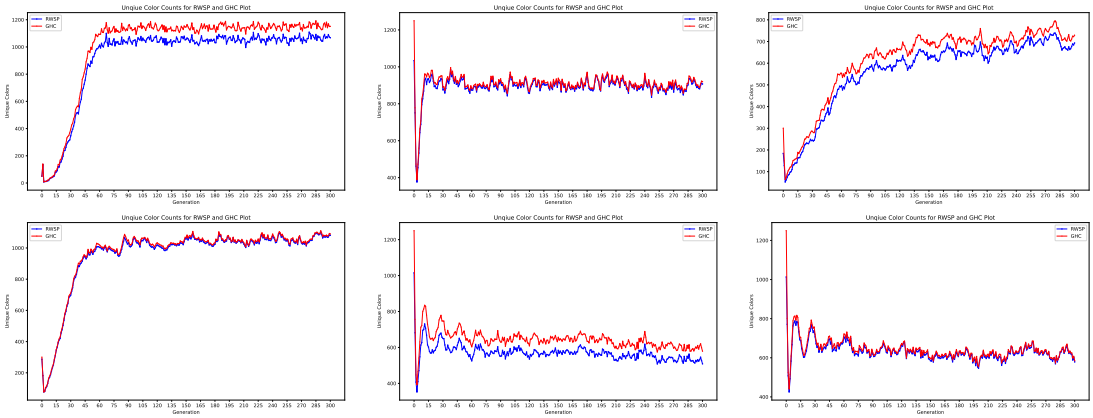


Figure 6.13: Unique Color Count Plots of RWSP and GHC for effect of High PPP, Initial Probability, High Budget and High Inheritance on Genotypic Diversity.

6.1.13 Tradeoff Visualised

Figure 6.15 shows how increasing PD NCA loses GD (first row) and increasing GD NCA loses PD (second row). We observed a tradeoff between Phenotype Diversity and Genotype Diversity. If we try to make phenotype more diverse, we lose diversity in genotype (For example low PPP configurations will yield diverse species in NCA but very less unique color counts in combined plot in GD). If we try to make genotype diverse, we lose diversity in phenotype (For example high PPP configurations will yield mostly one specie dominance (mostly value *one*) in NCA but more unique color counts in combined plot in GD). We refer [29] to support our observations of NCA as substrate with real-world examples. Bacteria have to make tough choices between being flexible with their traits and growing fast. They

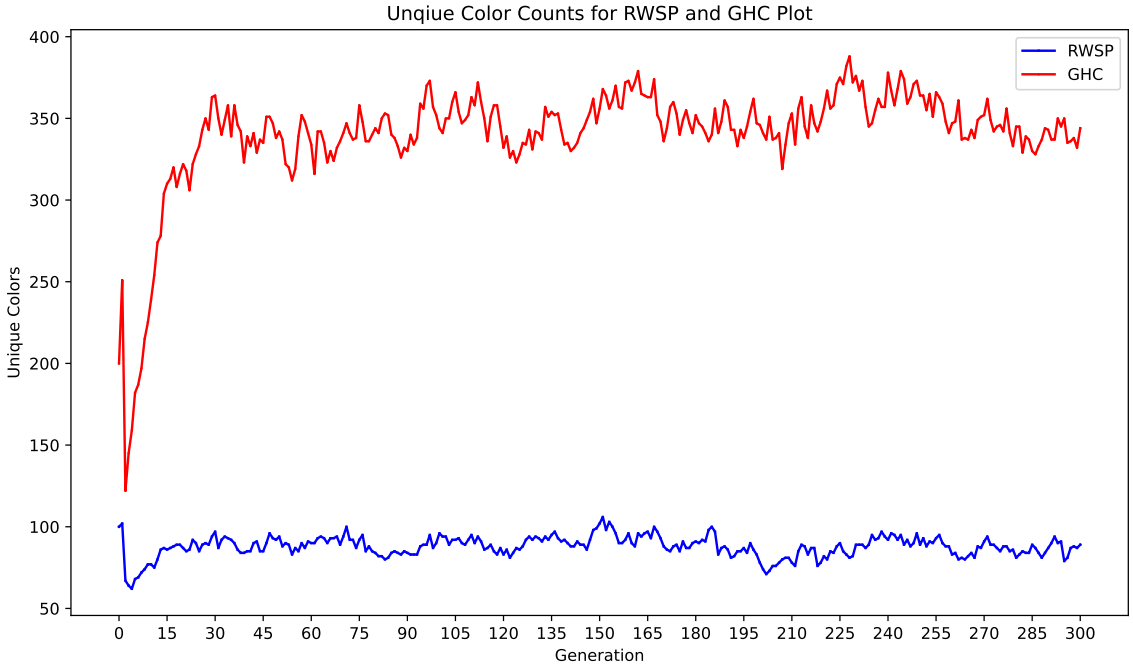


Figure 6.14: As observed, combined plot rightly captures the problem in RWSP plot vs GHC plot. Even in high PPP, the RWSP tool is not capturing the diversity in genetic material. In other words, RWSP if had captured the diversity it should have shown similar or little poor performance than GHC tool, instead it just performs worst with high PPP.

only have so many resources to work with, so they have to decide which genes to use. Sometimes they pick genes that help them grow quickly, like ones for making ribosomes, but that means they might miss out on other cool traits. This balancing act helps them survive in different environments, but it can also slow down their growth. Researchers are exploring this to discover methods for more effectively managing bacteria, which could be useful for fields such as medicine or food production.

6.2 Alternative Tools, Usability and Metaphors

The Proposed Neural CA serves as an intricate ecosystem, analog to a biological community, where agents engage in dynamic interactions with their neighbors to foster for novelty. In a parallel metaphor drawn from molecular biology, the continuous arrangement of millions of self-replicating processes replicates the overall behavior of a tissue or organ. The agents, corresponds to evolving organisms, undergo the continuous cycle of self-replication with inheritance and mutation as their sole survival strategy, corresponding to the intricate processes governing the growth and adaptation of living tissues. This generational evolution emerge complex behaviors within the cellular biome, resembling the specific dynamics shaping the developmental growth of living organisms. To quantify the complexity within this cellular ecosystem, Phenotypic Diversity (PD) and Genetic Diversity (GD) tools are used. PD tools act as a lens on the overall growth of cellular behavior, revealing the occurrence and distribution of different species or cell types within the biome. The proposed plots not only quantify the prevalence of species but also delve into homogeneity,

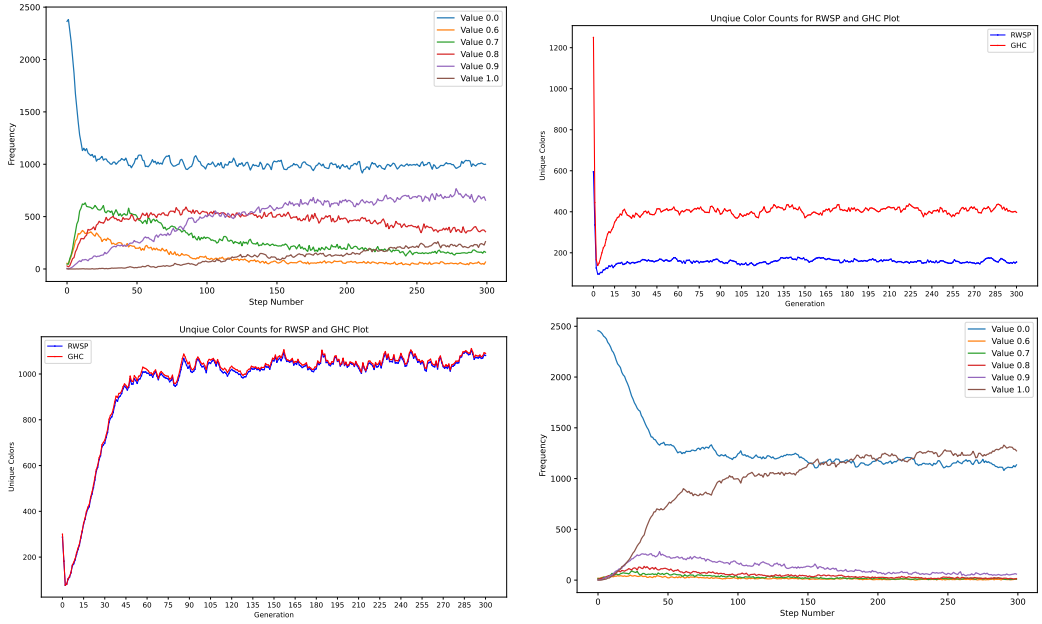


Figure 6.15: Tradeoff Visualised. First row shows CTFP plot preserving the PD while RWSP-GHC count plot shows decreased GD. Second row shows RWSP-GHC count plot preserving GD while CTFP plot shows decreased diversity as one specie overrides other species.

heterogeneity (entropy), local behavioral changes influencing global biome behavior, and local-global variances (overall diversity). This can be extended and allow us to evaluate patterns in molecular biology and genetics, providing insights into the cellular growth of specific genotypes and offering an understanding of the complex workings of cells within a living organism. On the other hand, GD tools function as genetic excavators, showing the propagation of specific traits across the genealogies within cell groups. Similar to the study of genetic evolution, these tools infer which traits prove unsuitable for the agents, leading to their extinction within the biome. This works as tool to dissect the intricate relationships of genetic information and trace the fate of traits as they influence the survival or demise of cellular lineages (genetic agencies) within the simulated ecosystem.

Continuing from the *Motivation from the Origins*, Evolution of the agents (or cells) take place in a favourable environment, that is the Neural CA. This acts as a biome for agents for evolution to takes place. Analogously, our biome starts with very few active cells (or agents), which further transforms itself to a very complex system with generations (a tissue or an organ). Similar to self-replication in cells, the Neural CA implements the same with inheritance and mutation for the overall growth. As of DNA, our agents carry neural weights which has different genes. As genes decides the traits mostly, our agents implement this by yielding out a cell value (after processing the neighbors) to represent the phenotype. We keep the inheritance (self-replication) rate very small to idealise the biological process and hence the realistic convergence of the cell-growth. Along with α, this process not only ensures the balance between order and chaos, but also produces a perfect balance of new offsprings which do not let the biome to collapse or burst. Self-replication with mutation and inheritance also ensures only right genetic traits pass to the progeny which are beneficial for the overall survival of the species. Narrowing our study towards

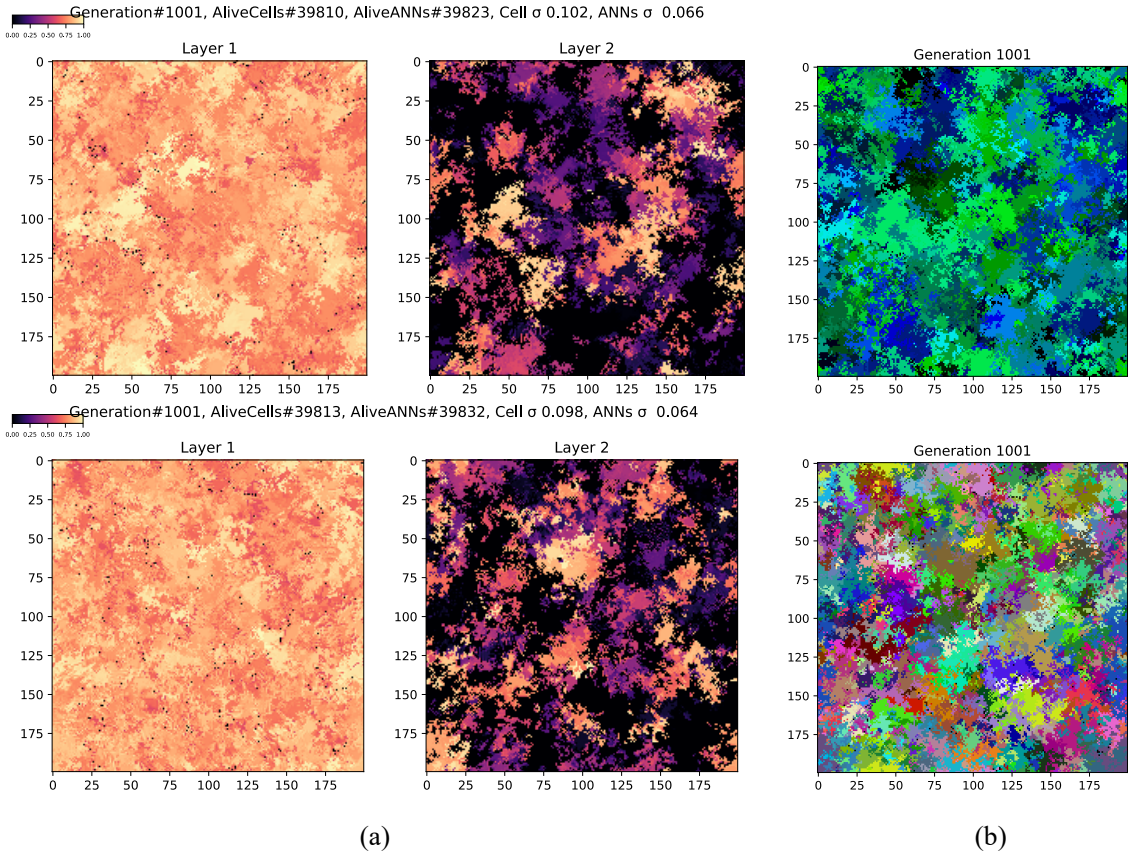


Figure 6.16: Phenotype-Genotype Relationships and Differences for two different NCA evolutionary runs (Experiment 19 from 4.5) (a) Evolved NCA biome (b) Corresponding RWSP plots.

only cellular forms of life, it becomes essential for a small cell (and hence our agents) to perform critical actions for its survival. For now, we do not introduce the ATP-like (Energy) based protocols in our Neural CA framework, however, we assume agents to have abundant energy with them however they can get replaced with other agencies (genetic information via inheritance) over generations. Moreover we carry a life-budget to make cells die forcefully. Further, for this life-like system, interesting dynamics emerge over generations, for example agents with some level of intelligence to keep their species longer to survive. This intelligence can be tracked where similar species has captured the biome, some species collaborate to live in diversity, some forms narrow clusters to make them distinct from other species. For example, Figure 6.16 shows two different runs for few generations with same configurations. While it shows same phenotypic traits for both the NCAs in (a) but has completely different corresponding genealogies in (b). For phenotype, we are not only considering the limited type of cellular species in NCA, but also the types of semi-local colonies that have emerged in α channel of the NCA. In addition, the effect of α is not present in the other layer (for chemical, energy or matter), however it has shown very interesting dynamics over generations to show similar phenotypic traits (for example, agents consuming similar energies, or have similar chemistry, or may have similar composition of matter), but genetic differences are clearly visible in the Figure 6.16 (b).

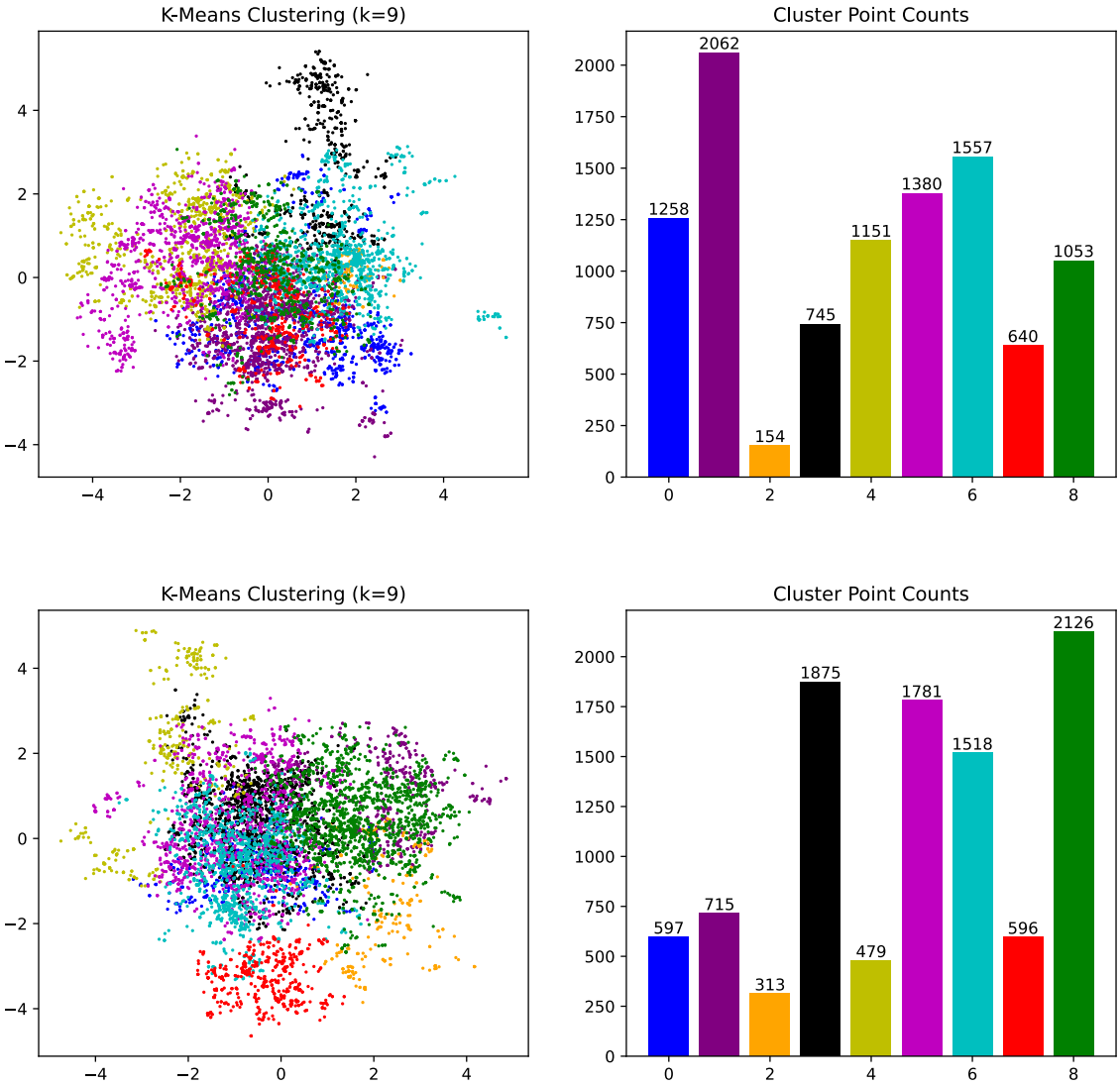


Figure 6.17: CNWA results for two different runs at a specific generation.

As illustrated in Figure 6.17, the CNWA results for both evolved NCAs at some later generation provide another interesting example. In the upper NCA depiction, the black cluster distinctively shifts away from overlapping clusters, showcasing a notable adaptation over evolutionary steps (including generations, inheritance, and mutations). The agents within this NCA appear to strategically modify the biome to enhance their survival, resulting in significant variations within certain species, as exemplified by the distinct genetic traits of the black cluster in the top CNWA result. Similarly, in the lower NCA, the red cluster, representing a distinct species, delineates itself from others. Instances where species overlap, yet exhibit differences in clusters, can be interpreted as indicative of shared genes with minor variations, implying robust ancestor-descendant relationships among them. This dynamic representation captures the intricate genetic transformations occurring within the evolving CA, providing valuable insights into the adaptability and divergence of species over successive generations. Please note that for CNWA, the results are observed one at a time for corresponding generation as preserving centroid changes significantly

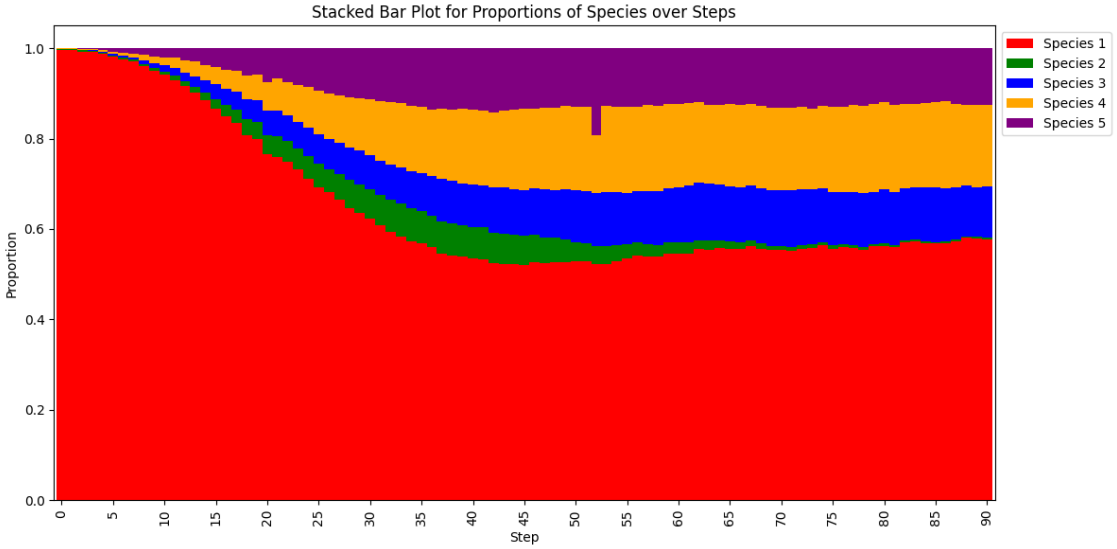


Figure 6.18: Speciation plots example, derived from CNWA tool.

over generations for data points and hence the colors are changed after intervals. This is a known problem with this tool which can be an interesting task to work upon. One way to resolve this could be to preserve centroids for all generations and then assigning colors iteratively. However this can be an expensive computation.

Figure 6.18 shows a small example of speciation derived from CNWA tool. The specific color spread in an area of the plot resembles the proportion dominance of the specie. To derive this from CNWA, we keep the counts of the clusters as we saw in Figure 6.17 for each generation corresponding to a specie and then put all of those counts in this plot. In summary, this plot shows the count of each cluster (specie) for generations. This can also be assumed as a generation wise bar plot but with bars covering complete plot resembling proportions of different specie for that generation. Since this tool is derived from the CNWA analysis, this plot also requires the same upgrade as we discussed for CNWA. While we are keeping CNWA as alternative methodology rather than main proposed methodology, the plot should only be considered as prototype and for reference. Notably, the visual analysis portrays the interesting life cycle of species, such as specie *2* depicted in green is only staying for few generations and then becomes extinct for later. The observed dynamics in each plot strongly suggest that, with the progression of evolutionary steps, agents within the NCA have undergone mutations, leading to the genetic diversification and emergence of new species.

Chapter 7

Conclusion

We started with a goal of development of an artificial substrate that can help thrive cellular agents grow. We used Cellular Automata (CA) as the substrate with neural operations and hence we called it neural CA (NCA) where the agents can multiple by self-replication and inheritance with mutation. Further, in order to quantify their growth, we devised several tools that could measure and track NCA growth by both the means (cell-wise and agency-wise). Overall, we started with motives of building an artificial life framework, let it grow by specific rules and then measure its complexity. In order to achieve this, (1) we developed NCA framework on top of classical CA with a neural operator layer, (2) infused it with life-like rules where the only way to survive is to self-replicate, inherit with mutation and (3) then quantify its growth using mathematical tools that includes entropy, mean and variance measures. In this chapter we will (1) answer to the research questions, (2) provide an overall picture of the thesis results, and (3) future work and scope of improvement.

7.1 Addressing Research Questions

The research questions on which this thesis was started, asks about (but not limited to) speciation of the evolved NCA biome, genotypic diversity, phenotypic diversity, self-replication, morphogenesis (autopoiesis), what enables diversity in such stochastic systems, and genotype-phenotype relations. Let us address these research questions one by one in following subsections.

7.1.1 RQ1 What are the key species that emerge, survive, reproduce and becomes extinct (dead) in the evolved substrate?

In the evolved NCA as it continue to grow with inheritance and mutation, more species continue to emerge. Speciation is the technique to classify the NCA growth for each generation in terms of available cell types, that means quantifying the counts of each of the cell types. To achieve this, Cellular Type Frequency Plot (CTFP) tracks growth of each cell type separately by showing count of each cell type for generations. As we have already seen in results and discussion chapters, CTFP plots show different dynamics corresponding to respective configurations, upon which we derived categorisation of experiments as ”one specie dominate”, ”coexistence of species” and ”species consumes all space to coexist” based on their phenotypical changes in CTFP plot.

7.1.2 RQ2 How does the genotypic diversity evolve over successive generations of the substrate?

Tracking genealogies of agents helps answering questions like how long a gene could dominate NCA, which further determines the overall growth of the species. As we showed in results and animated videos corresponding to the experiments, we visually track the lineage of agents by two methods namely Random Weight Selection Plot (RWSP) and Genotypic Hash Coloring (GHC). Both the tools provide a representation of agent genes in a RGB color value. While RWSP is sub-optimal since it ignores large genetic markup of an agent where as GHC uses complete genetic material to represent the corresponding RGB color. Simply, as experiments showed, GHC performed better than RWSP in many cases. Fallowing statements answer this research question:

1. With growing NCA, agents undergo self-replication, inheritance and mutation due to which their genetic markup changes very frequently on the basis of parameter perturbation and inheritance. As the agent changes their genes, corresponding representation also changes of the agents in both the tools. This help to track genealogy of a cell. A small change will not lead to very diverse color, however as the gene pool undergo multiple inheritance and mutations, the color changes drastically after few generations.

2. We also use unique color count plot that tells number of unique gene-pools existing at a specific generation. In other words, number of unique color at a generation provides number of unique agents (and hence different gene pool) of population. As we saw in results and discussions, we categorised as ”loss of phenotypic diversity when high mutation is enforced”, ”loss of genotypic diversity in low mutations”, ”effect of inheritance”, and ”effect of budget” in genotypic diversity.

3. Genotypic Diversity evolve on the basis of initial configurations used over successive generations. It continues to grow till it reaches an equilibrium state. RWSP and GHC tools provide visualisations which helps to track the genealogies of the agents over generations. An evaluation can easily be made by just perceiving the visuals how agents inherit (stays same color), mutates (changes color a little) and becomes a completely new specie (when all of its genes are mutated) over generations.

7.1.3 RQ3 What is the impact of the self-replication and self-maintenance processes, characteristic of autopoiesis, on the genetic makeup of the evolving substrate?

Most of NCA configurations show a self-maintenance behaviour where they learn to maintain the diversity over generations. We implemented a budget counter system (cells dying after fixed generations as they exhaust budget counter) which helped us to ensure the robustness of the system. Rather than resulting dead NCA simulations, we observe NCA was able to maintain a specific count of cells and agents. Though the agents were never able to capture complete space of the NCA but they self-replicate, produce new genes and becomes adaptable to evolve and reproduce to grow. We also see similar growth dynamics for higher budget systems. One supporting observation from the experiments that genomes are propagated in further generations which make them continue to produce diversity over time even if they are dying, its not like all genomes have died, even if there

are some chances that all genomes are lost somehow, it can still be preserved in higher budget as higher chance of inheritance taking place. Thus system behaves as if it has learned to preserve diversity by creating self. Therefor, systems (whether they are cells or populations) have built mechanisms that ensure their ongoing functionality and capacity to respond to environmental changes.

7.1.4 RQ4 How do phenotypic variations manifest within the substrate, and what are the underlying factors contributing to this diversity?

We strongly answer that phenotypic variations are driven by the underlying genetic markup that each cell or agent posses. As we have discussed the genotypic diversity and phenotypic diversity tradeoff, it can clearly be seen that genetic information largely affects phenotypic variations. As we observed, maintaining a specific level of genotypic diversity also maintains phenotypic diversity where as when we tune the NCA system for very large GD, we lose PD. Therefor, we see diverse population in substrate for critical cases which are result of genotypic variations. It is also wise to recall that in our system first agents become alive (by inheritance) and then they result a cell value in NCA for phenotypical variations.

7.1.5 RQ5To what extent do the cellular activities within the substrate exhibit sensitivity to initial conditions (for example number of time steps) and how does this influence the genetic and phenotypic outcomes?

We answer this research question by extensively performing experiments with different initial configurations of small runs and thus performing large experiments for hand-picked small experiments. As we discussed multiple cases of small runs in chapter 6, we categorised various behaviour of such NCA systems. We release large set of experimental setup for this research questions in Appendix C and all supporting results in chapter 6. We were able to observe following cases:

1. Coexistence of Species;
2. PD-GD tradeoff;
3. Species Consumes All Space of NCA to Coexist;
4. One Specie Dominate (Loss of Diversity case);
5. High mutations leading some species to co-dominate;
6. Low mutations helping different species to coexist with balanced GD;
7. Mediocre mutations show glider like behavior (recall GoL) with convergence to one specie;
8. High inheritance leads to high true diversity (GHC);
9. Very high mutations lead RWSP and GHC perform same which means agents are forced to adapt the environment at cost of losing genetic information very fast.

7.2 Summary

To summarise, we envisioned for a biome where cellular agents, when allowed to perform local interactions by self-replication and inheritance, continue to grow and show interesting long term dynamics. As we continue to build NCA framework as biome, we were able to perform large number of experiments on the basis of different initial conditions. This includes using CTFP, GEP, GCVP, and CLOGV for PD and RWSP and GHC for GD, enabling analyse various NCA biomes quantitatively and qualitatively respectively. We performed experiments on the basis of asking questions like what happens when we tune high-low mutations, inheritance, NCA size, life budget and death threshold. We found, NCA is not only robust to these conditions but also able to interesting dynamics and self-discover new ways to self-maintain. These conditions, though enforced *in-silico* for NCA, could also be suitable for *in-vitro* (petridish) experiments where NCA can be assumed as isolated closed-system experiment like test-tube experiments (wetware). In short, NCA can be seen as a large and complex multi-specie biome that evolves over generations with specific initial conditions and PD-GD tools work as coarse grainining lens to analyse and observe the behavioural changes of the NCA on the basis of entropy and variance.

7.3 Future Scope of Improvement

With the closing note, we release few points that are worth exploring and improving:

1. There are few alternative approaches we have proposed in methodology and discussed in later chapters, includes Clustering approach (CNWA) can be further improved by adjusting the centroid and preserving colors corresponding to each generation. This increases computational complexity but opens three other tools that can be derived from CNWA, namely, specie area plot, frequency plot for cluster counts and kernel density estimation. We provide prototypes of these plots in chapter 6.

2. The proposed NCA framework can also be tested on robustness by adding external adversaries (for example some patches that evade all agents). One can build kernel structure of 0s and 1s that can be multiplied with alpha channel to completely remove agents from a specific region. It would be interesting to see how NCA is able to show growth at the dead spot as well. In this scenario, the kernel functions as a hostile adversary, causing a grid annihilation event.

3. Faster computations can be achieved using Google JAX library rather than torch. JAX utilises parallel computing for CA like simulations.

4. One can use *in-vitro* dataset and analyse it using the proposed tools for bio-informatics and verify the relationships between *in-silico* and *in-vitro* agent behaviours.

Bibliography

[1] J. v. Neumann, "Theory of self-reproducing automata," *Edited by Arthur W. Burks*, 1966.

[2] M. Gardner, "Mathematical games," *Scientific american*, vol. 222, no. 6, pp. 132–140, 1970.

[3] S. Wolfram, "Statistical mechanics of cellular automata," *Reviews of modern physics*, vol. 55, no. 3, p. 601, 1983.

[4] S. Wolfram, "Cellular automata as models of complexity," *Nature*, vol. 311, no. 5985, pp. 419–424, 1984.

[5] H. R. Maturana and F. J. Varela, *Autopoiesis and cognition: The realization of the living*. Springer Science & Business Media, 1991, vol. 42.

[6] M. Mitchell, J. P. Crutchfield, R. 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)*, Moscow, vol. 8, 1996.

[7] V. Benci, C. Bonanno, S. Galatolo, G. Menconi, and F. Ponchio, "Information, complexity and entropy: A new approach to theory and measurement methods," *arXiv preprint math/0107067*, 2001.

[8] D. C. Crawford and D. A. Nickerson, "Definition and clinical importance of haplotypes," *Annu. Rev. Med.*, vol. 56, pp. 303–320, 2005.

[9] M. Mitchell *et al.*, "Computation in cellular automata: A selected review.," *Non-standard computation*, pp. 95–140, 2005.

[10] J. Secretan, N. Beato, D. B. D Ambrosio, A. Rodriguez, A. Campbell, and K. O. Stanley, "Picbreeder: Evolving pictures collaboratively online," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2008, pp. 1759–1768.

[11] M. Mitchell, *Complexity: A guided tour*. Oxford university press, 2009.

[12] L. H. Hartwell, L. Hood, M. L. Goldberg, A. E. Reynolds, and L. M. Silver, *Genetics: from genes to genomes*. McGraw-Hill, 2011.

[13] J. Lehman and K. O. Stanley, "Novelty search and the problem with objectives," *Genetic programming theory and practice IX*, pp. 37–56, 2011.

[14] G. Tufte and S. Nichele, "On the correlations between developmental diversity and genomic composition," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011, pp. 1507–1514.

[15] D. Medernach, T. Kowaliw, C. Ryan, and R. Doursat, "Long-term evolutionary dynamics in heterogeneous cellular automata," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, 2013, pp. 231–238.

[16] K. O. Stanley and J. Lehman, *Why greatness cannot be planned: The myth of the objective*. Springer, 2015.

[17] F. Baluška and M. Levin, "On having no head: Cognition throughout biological systems," *Frontiers in psychology*, vol. 7, p. 902, 2016.

[18] S. Nichele, T. E. Glover, and G. Tufte, "Genotype regulation by self-modifying instruction-based development on cellular automata," in *Parallel Problem Solving from Nature–PPSN XIV: 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings 14*, Springer, 2016, pp. 14–25.

[19] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Frontiers in Robotics and AI*, vol. 3, p. 202 845, 2016.

[20] A. De, V. S. Chakravarthy, and M. Levin, "A computational model of planarian regeneration," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 32, no. 4, pp. 331–347, 2017.

[21] K. O. Stanley, J. Lehman, and L. Soros, "Open-endedness: The last grand challenge you've never heard of," 2017, While open-endedness could be a force for discovering intelligence, it could also be a component of AI itself.

[22] B. W.-C. Chan, "Lenia-biology of artificial life," *arXiv preprint arXiv:1812.05433*, 2018.

[23] P. Stano, Y. Kuruma, and L. Damiano, "Synthetic biology and (embodied) artificial intelligence: Opportunities and challenges," *Adaptive Behavior*, vol. 26, no. 1, pp. 41–44, 2018.

[24] J. Clune, "Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence," *arXiv preprint arXiv:1905.10985*, 2019.

[25] J. S. McCaskill and N. H. Packard, "Analysing emergent dynamics of evolving computation in 2d cellular automata," in *Theory and Practice of Natural Computing: 8th International Conference, TPNC 2019, Kingston, ON, Canada, December 9–11, 2019, Proceedings 8*, Springer, 2019, pp. 3–40.

[26] K. O. Stanley, "Why open-endedness matters," *Artificial life*, vol. 25, no. 3, pp. 232–235, 2019.

[27] M. Braccini, A. Roli, and S. A. Kauffman, *Online adaptation in robots as biological development provides phenotypic plasticity*, 2020. arXiv: `2006.02367 [cs.RO]`.

[28] B. W.-C. Chan, "Lenia and expanded universe," *arXiv preprint arXiv:2005.03742*, 2020.

[29] J. Kim, A. Darlington, M. Salvador, J. Utrilla, and J. I. Jiménez, "Trade-offs between gene expression, growth and phenotypic diversity in microbial populations," *Current opinion in biotechnology*, vol. 62, pp. 29–37, 2020.

[30] A. Mordvintsev, E. Randazzo, E. Niklasson, and M. Levin, "Growing neural cellular automata," *Distill*, vol. 5, no. 2, 2020. eprint: `e23`.

[31] E. Randazzo, A. Mordvintsev, E. Niklasson, M. Levin, and S. Greydanus, "Self-classifying mnist digits," *Distill*, vol. 5, no. 8, e00027–002, 2020.

[32] L. Damiano and P. Stano, "A wetware embodied ai? towards an autopoietic organizational approach grounded in synthetic biology," *Frontiers in bioengineering and biotechnology*, vol. 9, p. 724 023, 2021.

[33] K. Gregor and F. Besse, "Self-organizing intelligent matter: A blueprint for an ai generating algorithm," *arXiv preprint arXiv:2101.07627*, 2021.

[34] W. Slackermanz, "Understanding multiple neighborhood cellular automata," *Slackermanz*, Jun. 2021. [Online]. Available: `https://slackermanz.com/understanding-multiple-neighborhood-cellular-automata/`.

[35] G. Hamon, M. Etcheverry, B. W.-C. Chan, C. Moulin-Frier, and P.-Y. Oudeyer, "Learning sensorimotor agency in cellular automata," 2022.

[36] A. Mordvintsev, E. Niklasson, and E. Randazzo, *Particle lenia and the energy-based formulation*, 2022.

[37] E. Plantec, G. Hamon, M. Etcheverry, P.-Y. Oudeyer, C. Moulin-Frier, and B. W.-C. Chan, "Flow lenia: Mass conservation for the study of virtual creatures in continuous cellular automata," *arXiv preprint arXiv:2212.07906*, 2022.

[38] P. Ball, *Synthetic morphology lets scientists create new life-forms*, May 2023. [Online]. Available: `https://www.scientificamerican.com/article/synthetic-morphology-lets-scientists-create-new-life-forms/`.

[39] B. W.-C. Chan, "Towards large-scale simulations of open-ended evolution in continuous cellular automata," *arXiv preprint arXiv:2304.05639*, 2023.

[40] O. M. Cliff, A. G. Bryant, J. T. Lizier, N. Tsuchiya, and B. D. Fulcher, "Unifying pairwise interactions in complex dynamics," *Nature Computational Science*, pp. 1–11, 2023.

[41] S. Jain and S. Nichele, "Frequency-histogram coarse graining in elementary and 2-dimensional cellular automata: Coarse graining in ca," *Nordic Machine Intelligence*, vol. 3, no. 2, pp. 15–110, 2023.

[42] S. Jain and S. Nichele, *MNCA portal — s4nyam.github.io*, `https://s4nyam.github.io/mncaportal/`, [Accessed 12-11-2023], 2023.

[43] S. Jain, A. Shrestha, and S. Nichele, "Capturing emerging complexity in lenia," *arXiv preprint arXiv:2305.09378*, 2023.

[44] E. Randazzo and A. Mordvintsev, "Biomaker ca: A biome maker project using cellular automata," *arXiv preprint arXiv:2307.09320*, 2023.

[45] L. Sinapayen, "Self-replication, spontaneous mutations, and exponential genetic drift in neural cellular automata," *arXiv preprint arXiv:2305.13043*, 2023.

[46] N. H. Packard and J. S. McCaskill, "Open-endedness in genelife," *Artificial Life*, pp. 1–34, 2024.

Appendix A

Mathematical Proofs and Examples

A.1 Phenotypic Diversity Tools

In this section we will see some examples for the Phenotypic Diversity (PD) tools. Please refer 3 for more conceptual details for corresponding tools.

A.1.1 Cellular Type Frequency Plot (CTFP)

Consider a 3×3 CA grid represented by matrix G, where W is the width and H is the height.

$$G^{(1)} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$G^{(2)} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$G^{(3)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Now, let's denote the counts of each state in the dictionaries as $Count^{(1)}$, $Count^{(2)}$, and $Count^{(3)}$:

$$\text{Step 1: } Count^{(1)}(0) = 4, Count^{(1)}(1) = 5$$

$$\text{Step 2: } Count^{(2)}(0) = 6, Count^{(2)}(1) = 3$$

$$\text{Step 3: } Count^{(3)}(0) = 7, Count^{(3)}(1) = 2$$

These counts represent the frequency of each cellular state in the corresponding dictionaries for the Grid (G) at time step (t). This is calculated for each time step. And hence a list containing dictionaries of the state and their corresponding count for each step. For example in Listing A.1,

```
state_count_list = [
    {0: 4, 1: 5},  # Step 1
    {0: 6, 1: 3},  # Step 2
    {0: 7, 1: 2}   # Step 3
]
```

Listing A.1: Example list of dictionaries for frequency-count

A.1.2 Global Entropy Plot (GEP)

For example, if we have a 3×3 grid with entities having possible states $S_i = \{0, 1, 2\}$, and let us assume:

$$\begin{aligned} &\text{Entities in state 0:} \quad 4 \\ &\text{Entities in state 1:} \quad 3 \\ &\text{Entities in state 2:} \quad 2 \\ &\text{Total entities in the model } (T_c: \quad 4+3+2=9 \end{aligned}$$

Let's calculate the spatial frequencies ($\rho(si)$) for each state:

$$\begin{aligned} \rho(0) &= \frac{4}{9} \\ \rho(1) &= \frac{3}{9} \\ \rho(2) &= \frac{2}{9} \end{aligned}$$

Now, calculate the contribution for each state using $Contribution(si) = -\rho(si) \cdot \log(\rho(si))$:

$$\begin{aligned} \text{Contribution}(0) &= -\left(\frac{4}{9}\right) \cdot \log\left(\frac{4}{9}\right) \\ \text{Contribution}(1) &= -\left(\frac{3}{9}\right) \cdot \log\left(\frac{3}{9}\right) \\ \text{Contribution}(2) &= -\left(\frac{2}{9}\right) \cdot \log\left(\frac{2}{9}\right) \end{aligned}$$

Calculate the total sum of contributions:

$$\Sigma\text{Contribution}(si) = \text{Contribution}(0) + \text{Contribution}(1) + \text{Contribution}(2)$$

Finally, calculate the global entropy ($H(t)$):

$$H(t) = \frac{1}{\log(3)} \cdot \Sigma\text{Contribution}(si)$$

In this example, $H(t)$ is calculated further ($\approx$0.63). This value provides a measure of the diversity of states within the model, where a lower value suggests less diversity or more uniformity among the states of the automaton. This $H(t)$ is calculated for each time step and then plotted.

A.1.3 Gross Cell Variance Plot (GCVP)

Consider a 3×3 grid with the following cell states:

$$\begin{matrix} 1 & 0 & 1 \\ 2 & 0 & 0 \\ 1 & 2 & 2 \end{matrix}$$

The ascending order of states is $0, 0, 0, 1, 1, 1, 2, 2, 2$. The median state ($S_{\text{med}}$) is *1*. Total entities in the model (T_c) is 9. Also $|S_0| = 3, |S_1| = 3, |S_2| = 3$.

Perform the following calculations for each cell ($C_{i,j}$) in the grid:

$$\begin{aligned} \sigma_{\text{gross}}(C_{1,1}) &= \frac{1}{\sqrt{T_c}} \sqrt{1 - \delta(S(1,1), S_{\text{med}})} \\ &= \frac{1}{3} \sqrt{1 - \delta(1,1)} \\ &= \frac{1}{3} \sqrt{1 - 1} \\ &= 0 \end{aligned}$$

$$\begin{aligned} \sigma_{\text{gross}}(C_{1,2}) &= \frac{1}{\sqrt{T_c}} \sqrt{1 - \delta(S(1,2), S_{\text{med}})} \\ &= \frac{1}{3} \sqrt{1 - \delta(0,1)} \\ &= \frac{1}{3} \sqrt{1 - 0} \\ &= \frac{1}{3} \end{aligned}$$

$$\begin{aligned} \sigma_{\text{gross}}(C_{1,3}) &= \frac{1}{\sqrt{T_c}} \sqrt{1 - \delta(S(1,3), S_{\text{med}})} \\ &= \frac{1}{3} \sqrt{1 - \delta(1,1)} \\ &= \frac{1}{3} \sqrt{1 - 1} \\ &= 0 \end{aligned}$$

$$\begin{aligned} \sigma_{\text{gross}}(C_{2,1}) &= \frac{1}{\sqrt{T_c}} \sqrt{1 - \delta(S(2,1), S_{\text{med}})} \\ &= \frac{1}{3} \sqrt{1 - \delta(2,1)} \\ &= \frac{1}{3} \sqrt{1 - 0} \\ &= \frac{1}{3} \end{aligned}$$

$$\begin{aligned}\sigma_{\text{gross}}(C_{2,2}) &= \frac{1}{\sqrt{T_c}}\sqrt{1-\delta(S(2,2),S_{\text{med}})}\\ &= \frac{1}{3}\sqrt{1-\delta(0,1)}\\ &= \frac{1}{3}\sqrt{1-0}\\ &= \frac{1}{3}\end{aligned}$$

$$\begin{aligned}\sigma_{\text{gross}}(C_{2,3}) &= \frac{1}{\sqrt{T_c}}\sqrt{1-\delta(S(2,3),S_{\text{med}})}\\ &= \frac{1}{3}\sqrt{1-\delta(0,1)}\\ &= \frac{1}{3}\sqrt{1-0}\\ &= \frac{1}{3}\end{aligned}$$

$$\begin{aligned}\sigma_{\text{gross}}(C_{3,1}) &= \frac{1}{\sqrt{T_c}}\sqrt{1-\delta(S(3,1),S_{\text{med}})}\\ &= \frac{1}{3}\sqrt{1-\delta(1,1)}\\ &= 0\end{aligned}$$

$$\begin{aligned}\sigma_{\text{gross}}(C_{3,2}) &= \frac{1}{\sqrt{T_c}}\sqrt{1-\delta(S(3,2),S_{\text{med}})}\\ &= \frac{1}{3}\sqrt{1-\delta(2,1)}\\ &= \frac{1}{3}\sqrt{1-0}\\ &= \frac{1}{3}\end{aligned}$$

$$\begin{aligned}\sigma_{\text{gross}}(C_{3,3}) &= \frac{1}{\sqrt{T_c}}\sqrt{1-\delta(S(3,3),S_{\text{med}})}\\ &= \frac{1}{3}\sqrt{1-\delta(2,1)}\\ &= \frac{1}{3}\sqrt{1-0}\\ &= \frac{1}{3}\end{aligned}$$

Finally, $\sigma_{gross} = \frac{1}{3} \times \sqrt{6}$ which is ≈ 0.81. It needs to be experimentally decided whether "0.81" is significantly higher to claim enough diversity. this value is calculated for the grid G at each time step t and plotted with x-axis as time steps.

A.1.4 Cell Local Organisation Global Variance (CLOGV)

Consider a 3×3 grid with the following cell states:

$$\begin{matrix} 0 & 1 & 2 \\ 0 & 1 & 1 \\ 2 & 1 & 0 \end{matrix}$$

First, lets calculate Normalized Frequency of Cell States. However before that, lets note down frequencies for each state. It can be seen from CA as, $\rho(S_0) = 3, \rho(S_1) = 4, \rho(S_2) = 2$,. Also, $\rho_{min} = 2$ and $\rho_{max} = 4$ are minimum and maximum frequencies respectively. Now we calculate ρ_{norm} as,

$$\begin{aligned} \rho_{\text{norm}}(S_0) &= \frac{\rho(S_0) - \rho_{\text{min}}}{\rho_{\text{max}} - \rho_{\text{min}}} \\ &= \frac{3-2}{4-2} \\ &= \frac{1}{2} \\ &= 0.5 \end{aligned}$$

$$\begin{aligned} \rho_{\text{norm}}(S_1) &= \frac{\rho(S_1) - \rho_{\text{min}}}{\rho_{\text{max}} - \rho_{\text{min}}} \\ &= \frac{4-2}{4-2} \\ &= \frac{1}{1} \\ &= 1 \end{aligned}$$

$$\begin{aligned} \rho_{\text{norm}}(S_2) &= \frac{\rho(S_2) - \rho_{\text{min}}}{\rho_{\text{max}} - \rho_{\text{min}}} \\ &= \frac{2-2}{4-2} \\ &= \frac{0}{2} \\ &= 0 \end{aligned}$$

Secondly, calculate Local State Frequency Mean and Variance. Before we jump to calculation, it is wise to understand and interpret as, $\mu_{loc}(state)$ and $\sigma_{loc}(state)$ is calculated for each cell in the grid. For cells we want to calculate μ_{loc} which is a mean of the local state frequency in a particular neighborhood of the cell in wrap-around condition. For each cell $C_{i,j}$, consider a 3×3 neighborhood ν_M:

$$\begin{aligned} \mu_{\text{loc}}(C_{1,1}) &= \frac{1}{9} \sum_{c \in \nu_M(C_{1,1})} \rho_{\text{norm}}(S_c) \\ &= \frac{1}{9} \left(3 \times \rho_{\text{norm}}(S_0) + 4 \times \rho_{\text{norm}}(S_1) + 2 \times \rho_{\text{norm}}(S_2)\right) \end{aligned}$$

Similarly, calculate μ_{loc} for other cells. In this particular case, since the grid size and the neighborhood sizes are same (3×3) and hence, we can simply say that $\mu_{\text{loc}}(C_{i*,j*})$ and $\sigma_{\text{loc}}(C_{i*,j*})$ will be same.

To provide an example, calculate local variance, $\sigma_{\text{loc}}(C_{1,1})$:

$$\sigma_{\text{loc}}(C_{1,1}) = \left(\frac{1}{9} \sum_{c \in \nu_M(C_{1,1})} (\rho_{\text{norm}}(S_c) - \mu_{\text{loc}}(C_{1,1}))^2 \right)^{1/2}$$

Similarly, calculate σ_{loc} for other cells.
Finally, we calculate Global Mean and Variance

$$\begin{aligned}
\mu_{\text{glob}} &= \frac{1}{T_c} \sum_{C_{i,j} \in \text{G}} \sigma_{\text{loc}}(C_{i,j}) \\
&= \frac{1}{9}(9 \times \mu_{loc} \text{ for any cell}) \\
&= \mu_{loc} \text{ for any cell} \\
\sigma_{\text{glob}} &= \left(\frac{1}{T_c} \sum_{C_{i,j} \in \text{G}} \left(\sigma_{\text{loc}}(C_{i,j}) - \mu_{\text{glob}}\right)^2 \right)^{1/2}
\end{aligned}$$

This process provides a measure of the variance in local organizations of cell states within the entire CA.

Appendix B

Additional Notes

In this Chapter we will define some important terminologies that are related Molecular Biology. These definitions are used throughout the thesis, and hence it is wise to know them. A surface level information is briefly provided in this Chapter.

B.1 A Biologist View - Triangulation Study

In this section, we enlist the specific ideas and concepts taken from genetics which we use in different sections of this research work. We list them as follows:

1. **Allele:** A specific form or variant of a gene, typically occurring at a specific location on a chromosome.
2. **Genotype:** The genetic makeup of an individual, referring to the combination of alleles present for a particular set of genes.
3. **Phenotype:** The observable traits or characteristics of an organism, resulting from the interaction between its genotype and the environment.
4. **Genetic Variation:** Differences in the genetic material (DNA) among individuals of a population or species.
5. **Speciation:** The process by which new and distinct species evolve from existing species.
6. **Genetic Drift:** Changes in the frequency of alleles within a population over generations due to random events.
7. **Homologous Structures:** Similar structures in different species that share a common evolutionary origin.
8. **Homology:** Similarity between structures or genes due to shared ancestry.
9. **Analogous Structures:** Structures in different species that have similar functions but different evolutionary origins.
10. **Pleiotropy:** A single gene influencing multiple, seemingly unrelated traits.
11. **Epistasis:** The interaction of multiple genes to determine a single phenotype.

12. **Codon:** A three-nucleotide sequence in DNA or RNA that codes for a specific amino acid.
13. **Mendelian Inheritance:** The principles of inheritance proposed by Gregor Mendel, involving the transmission of discrete traits.
14. **Heterozygous:** Having two different alleles for a particular gene.
15. **Homozygous:** Having two identical alleles for a particular gene.
16. **Polygenic Inheritance:** Inheritance influenced by multiple genes.
17. **Phylogeny:** The evolutionary history and relationships among species.
18. **Convergent Evolution:** The independent evolution of similar traits in unrelated species.
19. **Divergent Evolution:** The evolution of different traits in closely related species.
20. **Founder Effect:** Genetic drift occurring when a small group establishes a new population.
21. **Codominance:** A situation where both alleles of a gene are fully expressed in a heterozygote.
22. **Epistatic Gene:** A gene that masks the phenotypic effects of another gene.
23. **Phenotypic Plasticity:** The ability of a single genotype to produce different phenotypes in response to environmental changes.
24. **Hardy-Weinberg Equilibrium:** A model describing the stable frequency of alleles in a population that is not evolving.
25. **Reproductive Isolation:** Barriers that prevent members of different species from interbreeding.
26. **Phenotypic Assortative Mating:** Individuals preferentially mating with others that have similar phenotypes.
27. **Sympatric Speciation:** The formation of new species within the same geographic area.
28. **Allopatric Speciation:** The formation of new species due to geographic isolation.
29. **Haplotype:** A haplotype is a combination of alleles at multiple loci on a single chromosome that are inherited together. It provides information about the genetic variation within a region and is often used in the study of genetic linkage and association.

Appendix C

Experimental Setup for Small Runs

This chapter contains the configuration names enlisting each of the corresponding experimental configurations for the results listed in chapter 6, section 6.1. Each configuration is coded in a particular format. For example a configuration ppp0.02,init0.02,l3,sig,b16,i0.05 should be read as parameter perturbation probability of 0.02, initial seeds are 2%, channels or (layers) as 3, sigmoid activation on board, budget of 16 and inheritance probability of 0.05. Moreover all configurations have same NCA width and height of 50 and ran for 300 generations. Following subsections labels the category and enlists the example configurations.

1. Coexistence of Species
   (a) ppp0.02,init0.02,l3,sig,b16,i0.05
   (b) ppp0.02,init0.20,l2,sig,b8,i0.05
   (c) ppp0.02,init0.08,l2,tanh,b8,i0.4
   (d) ppp0.02,init0.12,l3,sig,b8,i0.1
   (e) ppp0.02,init0.5,l3,sig,b8,i0.300
   (f) ppp0.02,init0.5,l3,tanh,b4,i0.300
2. Species consumes all space to coexist (Low PPP)
   (a) ppp0.02,init0.12,l3,sig,b16,i0.300
   (b) ppp0.02,init0.50,l3,tanh,b16,i0.500
   (c) ppp0.02,init0.02,l2,tanh,b16,i0.400
   (d) ppp0.02,init0.12,l3,tanh,b16,i0.500
   (e) ppp0.02,init0.08,l2,sig,b16,i0.500
3. One specie dominate (Loss of Diversity) (Medium PPP)
   (a) ppp0.20,init0.12,l3,sig,b16,i0.075
   (b) ppp0.20,init0.50,l2,sig,b4,i0.200
   (c) ppp0.20,init0.02,l2,tanh,b8,i0.200
   (d) ppp0.20,init0.20,l3,tanh,b16,i0.500
   (e) ppp0.20,init0.08,l3,sig,b8,i0.500

4. High PPP
    (a) ppp0.80,init0.12,l3,sig,b8,i0.1
    (b) ppp0.80,init0.50,l2,sig,b4,i0.200
    (c) ppp0.80,init0.02,l2,tanh,b8,i0.200
    (d) ppp0.80,init0.20,l3,tanh,b16,i0.200
    (e) ppp0.50,init0.20,l2,sig,b8,i0.500
    (f) ppp0.50,init0.08,l2,sig,b16,i0.100
5. Type 1 Plots for tools 2,3,4! (Low PPP)
    (a) ppp0.02,init0.02,l3,sig,b8,i0.1
    (b) ppp0.02,init0.02,l3,tanh,b4,i0.400
    (c) ppp0.02,init0.50,l2,sig,b8,i0.100
    (d) ppp0.02,init0.20,l3,tanh,b8,i0.1
    (e) ppp0.02,init0.12,l2,sig,b4,i0.2
    (f) ppp0.02,init0.08,l2,sig,b16,i0.05
6. Type 2 Plots for tools 2,3,4! (Med PPP)
    (a) ppp0.2,init0.12,l3,tanh,b8,i0.1
    (b) ppp0.2,init0.02,l3,tanh,b4,i0.400
    (c) ppp0.2,init0.50,l2,sig,b8,i0.100
    (d) ppp0.2,init0.12,l3,sig,b4,i0.3
    (e) ppp0.2,init0.08,l3,tanh,b4,i0.2
    (f) ppp0.02,init0.08,l2,sig,b16,i0.05
7. Type 3 Plots for tools 2,3,4
    (a) ppp0.20,init0.12,l2,sig,b4,i0.1
    (b) ppp0.2,init0.08,l3,tanh,b4,i0.100
    (c) ppp0.02,init0.12,l2,sig,b4,i0.100
    (d) ppp0.2,init0.20,l2,sig,b4,i0.1
    (e) ppp0.2,init0.50,l2,tanh,b4,i0.1
8. Type 4 Plots for tools 2,3,4 (High PPP)
    (a) ppp0.80,init0.02,l2,sig,b8,i0.1
    (b) ppp0.80,init0.02,l2,sig,b4,i0.3
    (c) ppp0.80,init0.08,l2,sig,b8,i0.075
    (d) ppp0.80,init0.12,l2,tanh,b4,i0.2
    (e) ppp0.8,init0.50,l3,sig,b16,i0.05
9. Effect by inheritance probability

(a) ppp0.02,init0.12,l2,sig,b4,i0.1
(b) ppp0.02,init0.12,l2,sig,b4,i0.2
(c) ppp0.02,init0.12,l2,sig,b4,i0.3
(d) ppp0.02,init0.12,l2,sig,b4,i0.5

10. Effect by parameter perturbation probability

(a) ppp0.02,init0.50,l2,sig,b4,i0.4
(b) ppp0.2,init0.50,l2,sig,b4,i0.4
(c) ppp0.5,init0.50,l2,sig,b4,i0.4
(d) ppp0.8,init0.50,l2,sig,b4,i0.4

11. Effect by budget and inheritance

(a) ppp0.2,init0.12,l3,tanh,b4,i0.1
(b) ppp0.2,init0.12,l3,tanh,b8,i0.1
(c) ppp0.2,init0.12,l3,tanh,b16,i0.1
(d) ppp0.2,init0.12,l3,tanh,b4,i0.5
(e) ppp0.2,init0.12,l3,tanh,b8,i0.5
(f) ppp0.2,init0.12,l3,tanh,b16,i0.5

12. Some other interesting plots GD (1)

(a) ppp0.5,init0.02,l2,sig,b4,i0.4
(b) ppp0.02,init0.02,l3,tanh,b8,i0.05
(c) ppp0.02,init0.02,l3,tanh,b4,i0.1
(d) ppp0.8,init0.50,l2,sig,b8,i0.5
(e) ppp0.8,init0.12,l2,sig,b8,i0.075
(f) ppp0.8,init0.12,l2,sig,b16,i0.1

13. Some other interesting plots GD (2)

(a) ppp0.8,init0.2,l2,tanh,b16,i0.2
(b) ppp0.50,init0.50,l2,tanh,b8,i0.5
(c) ppp0.5,init0.5,l2,sig,b16,i0.4
(d) ppp0.5,init0.50,l2,sig,b16,i0.5
(e) ppp0.5,init0.50,l3,sig,b16,i0.5
(f) ppp0.8,init0.5,l2,sig,b16,i0.5

All *tar* export files for small experiments are available at `https://archive.org/download/gnca0/all_small_runs.tar`.