Agent-Based Modelling of Microorganism Dynamics in Regenerative Environments using Boids Algorithm and Cellular Automata

Shubhamita Banerjee **Sayan Saha**

Department of Computer Science, Scottish Church College

Contents

1 Abstract

This project presents a hybrid agent-based modelling framework that combines the emergent flocking behaviour of Boids with the cellular automata dynamics of Conway's Game of Life (GoL). By integrating these two systems, we aim to explore the dual nature of emergent patterns in decentralized environments. The simulation introduces various food distribution types—including uniform, clustered, and Gaussian—on a 2D grid to observe how Boid agents interact with evolving environments governed by GoL rules. A user interface has been developed to visualize the system's real-time behaviour and to manipulate interaction rules. This hybrid model provides a platform to study multi-agent coordination, resource-driven movement, and environmental feedback, with potential applications in biology, swarm robotics, and artificial life. The study aims to contribute valuable insights toward future research in Boids, Game of Life, and agent-based system design.

2 Introduction

Multi-agent systems and cellular automata represent two fundamental paradigms in computational modelling that have independently contributed to our understanding of complex emergent behaviours.
The Boids algorithm, developed by Craig Reynolds in 1986, simulates the collective flocking behaviour of birds through simple local rules, while Conway's Game of Life, created by John Conway in 1970, demonstrates how complex patterns can emerge from basic cellular automaton rules. This project explores the synergistic potential of combining these two influential algorithms to create a hybrid modelling framework for studying population dynamics in regenerative environments.

The Boids algorithm models autonomous agents that exhibit flocking behaviour through three fundamental rules: separation (avoiding crowding), alignment (steering towards the average heading of neighbours), and cohesion (steering towards the average position of neighbours). These simple rules give rise to complex collective behaviours observed in natural systems, from bird flocks to fish schools and even human crowd dynamics. The algorithm has found extensive applications in computer graphics, robotics, and biological modelling due to its ability to generate realistic emergent behaviours from decentralized local interactions.

Conway's Game of Life operates on a grid of cells that can be either alive or dead, with each cell's state determined by simple rules based on its neighbouring cells. Despite its simplicity, the Game of Life can produce remarkably complex patterns, including stable structures, oscillators, and even self-replicating patterns. As a cellular automaton, it serves as a powerful tool for studying emergence, self-organization, and the evolution of complex systems from simple initial conditions.

Flocking and foraging behaviours are fundamental aspects of biological systems that have evolved to optimize survival and resource acquisition. Flocking provides advantages such as protection from predators, improved navigation, and efficient information sharing among group members. Foraging behaviour involves the search for and acquisition of resources, often requiring organisms to balance exploration of new areas with exploitation of known resource locations. These behaviours are particularly relevant when studying how organisms adapt to dynamic environments where resources regenerate over time.

Population dynamics examines how populations change over time due to births, deaths, immigration, and emigration. In biological systems, population dynamics are influenced by factors such as resource availability, environmental conditions, and inter-species interactions. Understanding these dynamics is crucial for predicting population stability, growth patterns, and the long-term survival of species in changing environments. Computational models provide valuable tools for exploring these complex interactions under controlled conditions.

Multi-agent systems offer a computational approach to modelling complex systems composed of multiple interacting autonomous agents. Each agent operates according to local rules and has limited knowledge of the global system state, yet their collective behaviour can exhibit sophisticated emergent properties. This paradigm is particularly well-suited for studying biological systems where individual organisms make decisions based on local information, leading to complex group behaviours and system-level phenomena.

The motivation for combining the Boids algorithm with Conway's Game of Life stems from the opportunity to study how collective agent behaviour adapts to dynamic environmental conditions. Traditional boids implementations typically operate in static environments, while Conway's Game of Life provides a continuously evolving substrate that can represent regenerating resources or changing environmental conditions. This hybrid approach allows us to investigate how flocking behaviour influences resource distribution patterns and how environmental dynamics affect population stability and survival strategies.

The significance of this project lies in its potential to advance our understanding of multi-agent systems operating in dynamic environments. By systematically exploring different initial distribution patterns of Game of Life cells—including uniform random, clustered and Gaussian —we can investigate how environmental structure influences collective behaviour and population dynamics. The project contributes to the broader field of agent-based modelling by demonstrating novel interactions between autonomous agents and cellular automata, providing insights that could inform applications in swarm robotics, biological modelling, and complex systems research.

In the following sections, we will detail the theoretical foundations of both algorithms, describe our hybrid modelling approach, present the system architecture and implementation details, and analyse the emergent behaviours observed under different experimental conditions.

3 Background / Review of related work

3.1 Boids Algorithm

3.1.1 History and Development

The Boids algorithm represents a seminal contribution to the field of artificial life and computer graphics, developed by Craig Reynolds in 1986 at Symbolics Inc. Reynolds was inspired by the mesmerizing patterns of bird flocks, fish schools, and other collective animal behaviours that seemed to emerge from simple local interactions without any central coordination. His groundbreaking work, first presented in the paper "Flocks, Herds, and Schools: A Distributed Behavioural Model," introduced the concept that complex group behaviours could arise from individual agents following simple local rules.

The term "boid" itself is a contraction of "bird-oid object," reflecting Reynolds' original inspiration from avian flocking behaviour. However, the algorithm's implications quickly extended far beyond its ornithological origins, finding applications in computer animation, robotics, crowd simulation, and biological modelling. The algorithm gained widespread recognition when it was used to create the realistic flocking behaviour of bats in Tim Burton's 1992 film "Batman Returns," demonstrating its practical utility in computer graphics and animation.

3.1.2 Algorithm Principles

The Boids algorithm operates on the principle of distributed behavioural modelling, where each individual agent (boid) makes decisions based solely on its local environment and the positions and velocities of nearby neighbours. Unlike centralized control systems, no single agent has knowledge of the global state or issues commands to other agents. This decentralized approach mirrors natural flocking behaviour, where individual birds or fish respond to their immediate neighbours without awareness of the entire flock's structure or destination.

Each boid in the system possesses several key attributes: position, velocity, and a limited perception radius that defines its "neighbourhood." The algorithm updates these attributes at each time step based on the application of three fundamental steering behaviours. The beauty of the system lies in its simplicity – complex, lifelike group behaviours emerge from the interaction of just these three basic rules applied consistently across all agents in the population.

3.1.3 Fundamental Rules

3.1.3.1 Separation

The separation rule ensures that boids avoid crowding their local flock mates by steering away from nearby neighbours when they come too close. The separation force is inversely proportional to distance—closer boids generate stronger repulsive forces. This creates a "personal space" around each agent, preventing collisions and maintaining comfortable spacing similar to natural bird flocks.

3.1.3.2 Alignment

The alignment rule guides boids to steer towards the average heading of their local neighbours, creating coordinated movement throughout the flock. By averaging velocity vectors of nearby neighbours, this behaviour ensures that agents move in similar directions, preventing random dispersal and creating the smooth, coordinated movements observed in natural flocks like starling murmurations.

3.1.3.3 Cohesion

The cohesion rule attracts boids toward the average position of their local neighbours, preventing the flock from spreading indefinitely and maintaining group integrity. This gentle attractive force draws agents toward the centre of their local neighbourhood, ensuring the group remains together for protection and information sharing benefits observed in biological systems.

3.1.4 Population Dynamics and Collective Behaviour in Biological Systems

Population dynamics encompasses the study of how populations of organisms change over time and space, influenced by factors such as birth rates, death rates, migration patterns, and environmental constraints. In the context of microorganisms and social species, collective behaviours like flocking and foraging emerge from simple individual interactions that scale up to complex group phenomena.

Flocking behaviour, observed in birds, fish, and even bacterial colonies, demonstrates how local interactions between individuals following basic rules can produce sophisticated emergent patterns at the population level. Flocks can split around obstacles and reform on the other side, create swirling vortex patterns, exhibit leader-follower dynamics, and demonstrate collective decision-making when faced with environmental challenges. These emergent behaviours arise naturally from the local interactions between agents, without any explicit programming of group-level behaviours and serves multiple evolutionary purposes, including predator avoidance, improved foraging efficiency, and enhanced navigation capabilities through collective sensing.

Foraging behaviour in populations involves the strategic search for and acquisition of resources, often exhibiting patterns that optimize energy expenditure while maximizing resource acquisition. Microorganisms exhibit particularly interesting foraging strategies, such as chemotaxis in bacteria where individuals move toward favourable chemical gradients, creating collective migration patterns.

3.1.5 Applications

The boids algorithm has found extensive applications across multiple domains such as computer graphics, robotics and biological research. In computer graphics and animation, boids are widely used to create convincing flocks of birds, schools of fish, and crowd simulations in movies and video games, providing natural-looking movement patterns that would be computationally expensive to animate manually. The algorithm's simplicity and effectiveness have made it a standard tool in the entertainment industry for generating organic-looking group behaviours.

In robotics and autonomous systems, boids principles are applied to coordinate swarms of robots, drones, and autonomous vehicles, enabling them to work together without centralized control. This distributed approach is particularly valuable for applications such as search and rescue operations, environmental monitoring, and military reconnaissance where robust, scalable coordination is essential. The algorithm's fault tolerance and adaptability make it suitable for dynamic environments where individual agents may fail or new agents may join the system.

Biological research has embraced boids modelling to understand and predict animal behaviour patterns, from bacterial colony growth to bird migration routes. Researchers use modified boids algorithms to study how environmental factors influence collective behaviour, test hypotheses about evolutionary advantages of flocking, and predict population responses to habitat changes. The algorithm's mathematical foundation allows for quantitative analysis of biological systems while maintaining computational efficiency for large-scale simulations.

3.2 Conway's Game of Life

3.2.1 History and Development

The Game of Life, also known as Conway's Game of Life or simply Life, was conceived by British mathematician John Conway in 1970, emerging from his interest in creating a cellular automaton that could simulate simple forms of artificial life. Conway designed the Game of Life as a zero-player game, meaning its evolution is determined entirely by its initial state without requiring further input from players. It is Turing complete and can simulate a universal constructor or any other Turing machine.

The game gained widespread attention after being featured in Martin Gardner's "Mathematical Games" column in Scientific American in October 1970, sparking global interest in cellular automata and computational biology. Conway's motivation was to create the simplest possible rules that could produce complex, unpredictable behaviour, challenging the notion that sophisticated patterns require complicated underlying mechanisms.

The development of Game of Life represented a significant milestone in the study of emergent systems and computational complexity. Conway spent considerable time experimenting with different rule sets before settling on the final version, testing various configurations to ensure the system would neither die out too quickly nor grow indefinitely. His work built upon earlier cellular automata research by mathematicians like John von Neumann and Stanisław Ulam.

3.2.2 Cellular Automata

A cellular automaton is a discrete model of computation studied in automata theory. It consists of a regular grid of cells, each in one of a finite number of states, such as on and off. The grid can be in any finite number of dimensions. For each cell, a set of cells called its neighbourhood is defined relative to the specified cell. An initial state (time t = 0) is selected by assigning a state for each cell. A new generation is created (advancing t by 1), according to some fixed rule that determines the new state of each cell in terms of the current state of the cell and the states of the cells in its neighbourhood.

This mathematical framework was first introduced by John von Neumann and Stanisław Ulam in the 1940s as a theoretical foundation for studying self-replication and complex systems, particularly von Neumann's interest in understanding how machines could reproduce themselves.

The theoretical foundation of cellular automata rests on the principle that complex global behaviours can emerge from simple local interactions, making them powerful tools for studying emergent phenomena in natural and artificial systems. Each cell in the automaton acts as a simple computational unit that processes information from its immediate neighbours according to predetermined rules, yet the collective behaviour of all cells can produce sophisticated patterns and dynamics. This bottom-up approach to modelling complex systems has proven invaluable for understanding phenomena where local interactions give rise to global properties, from biological processes like tissue growth and immune responses to physical phenomena such as crystal formation and fluid dynamics. The mathematical elegance of cellular automata lies in their ability to capture the essence of distributed computation and self-organization through remarkably simple rule sets.

3.2.3 Game of Life

The Game of Life represents a binary cellular automaton that operates on an infinite two-dimensional grid of square cells, where each cell exists in one of two states: alive (1) or dead (0). The system employs a Moore neighbourhood structure, meaning each cell's evolution depends on its current state and the states of its eight adjacent neighbours (horizontally, vertically, and diagonally adjacent). What distinguishes Game of Life as a zero-player game is that once the initial configuration is set, the system evolves deterministically without any further external input, with each generation computed simultaneously across all cells based on the previous generation's state.

The system's mathematical elegance lies in its ability to exhibit all four classes of cellular automaton behaviour: patterns that die out (Class I), patterns that stabilize into static or oscillating structures (Class II), chaotic patterns with no apparent regularity (Class III), and complex patterns that exhibit both regularity and randomness (Class IV). The Game of Life has been proven to be Turing-complete, meaning it can simulate any computation that can be performed by a universal Turing machine, despite its deceptively simple rules and binary state space.

3.2.4 Fundamental Rules

Conway's Game of Life operates according to four simple rules that govern cell state transitions from one generation to the next. These rules are applied simultaneously to every cell in the grid based on the current state of the cell and the count of its eight living neighbours.

3.2.4.1 Underpopulation

Any live cell with fewer than two live neighbours dies, as if by underpopulation.

3.2.4.2 Survival

Any live cell with two or three live neighbours lives on to the next generation.

3.2.4.3 Overpopulation

Any live cell with more than three live neighbours dies, as if by overpopulation.

3.2.4.4 Reproduction

Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

The specific values in these rules were carefully chosen by Conway through extensive experimentation to create a system that balances growth and decay, preventing the automaton from either dying out completely or expanding indefinitely while producing the most interesting emergent behaviours.

3.2.5 Emergent Patterns and Structures

Conway's Game of Life produces three primary categories of emergent patterns that arise spontaneously from the fundamental rules.

3.2.5.1 Still lifes

Still lifes are static patterns that remain unchanged across generations, such as blocks (2×2 squares) and beehives, representing stable equilibrium states.

3.2.5.2 Oscillators

Oscillators are patterns that cycle through a sequence of configurations before returning to their original state, with the blinker (alternating between horizontal and vertical lines) being the simplest example with a period of 2.

3.2.5.3 Spaceships

Spaceships are translating patterns that move across the grid while maintaining their shape, with the glider being the most famous example—a 5-cell pattern that travels diagonally one cell every four generations.
The most significant discovery was the glider, which demonstrated that static rules could produce self-propelling structures, fundamentally changing understanding of movement in discrete systems. More complex spaceships include lightweight, middleweight, and heavyweight spaceships that travel horizontally or vertically at different speeds. These patterns exhibit remarkable properties like they can interact with each other to create more complex behaviours, collide to produce new patterns, and even be engineered to perform computational operations. The existence of these emergent structures, particularly their ability to carry and transmit information across the grid, contributes to the Game of Life's Turing completeness and its capacity for universal computation.

3.2.6 Applications

In mathematics and theoretical computer science, Game of Life serves as a fundamental tool for studying computational complexity, decidability problems, and the relationship between local

rules and global behaviour. Mathematicians also use Game of Life to explore questions in dynamical systems theory, particularly how simple deterministic rules can produce unpredictable long-term behaviour, contributing to chaos theory and complexity science.

Computer science applications span algorithm design, parallel computing, and artificial intelligence research. Game of Life implementations serve as benchmarks for testing parallel processing architectures and GPU computing performance due to their inherently parallel nature and computational intensity. In software engineering education, the game teaches fundamental programming concepts including data structures, optimization techniques, and algorithm efficiency. Machine learning researchers use cellular automata variants to study emergent behaviour in neural networks and explore self-organizing systems.

Beyond computational applications, Game of Life has influenced diverse fields including biology, physics, and philosophy. Biologists employ cellular automata models to simulate population dynamics, tumour growth, and ecosystem evolution, while physicists use similar principles to model crystal growth, fluid dynamics, and phase transitions in materials science. The game's philosophical implications extend to discussions about emergence, reductionism, and artificial life, serving as a concrete example of how simple rules can generate complex, apparently purposeful behaviours. These interdisciplinary applications demonstrate the universal relevance of cellular automata principles in understanding complex systems across multiple domains.

3.3 Multi-Agent Systems and Swarm Intelligence

Multi-agent systems (MAS) represent computational frameworks composed of multiple interacting autonomous agents that collectively solve problems through decentralized coordination. These systems are characterized by distributed control, where agents must coordinate through local interactions and communication protocols without centralized oversight. The computational power of MAS lies in their ability to tackle complex problems by decomposing them into smaller, manageable tasks that can be handled by individual agents working in parallel.

Swarm intelligence algorithms represent a specific class of multi-agent systems inspired by the collective behaviour of social insects. Ant Colony Optimization (ACO) mimics the pheromone-trail-following behaviour of real ants, where artificial agents deposit virtual pheromones to mark successful paths, enabling the swarm to converge on optimal solutions for routing and scheduling problems. Particle Swarm Optimization (PSO) emulates the flocking behaviour of birds, where agents adjust their positions based on their own best-known position and the global best position discovered by the swarm, making it effective for continuous optimization problems. These algorithms demonstrate superior performance in complex, dynamic environments where traditional centralized approaches struggle due to incomplete information.

The robustness of swarm intelligence systems stems from their inherent redundancy, fault tolerance, and adaptability. Individual agent failures rarely compromise overall system performance, while the distributed nature of computation allows for scalable solutions that can handle large problem spaces. These characteristics make swarm intelligence particularly valuable for applications in robotics, network optimization, and distributed artificial intelligence, where systems must operate reliably in unpredictable environments while maintaining efficient collective problem-solving capabilities.

3.4 Agent-Based Modelling

Agent-based modelling (ABM) is a computational methodology that simulates complex systems by modelling individual autonomous agents and their interactions within a defined environment. Each agent in an ABM represents a discrete entity with its own set of characteristics, behaviours, and decision-making rules, operating independently while influencing and being influenced by other agents and the environment. This bottom-up approach allows researchers to study how macroscopic patterns and behaviours emerge from microscopic interactions, making it particularly valuable for understanding systems where individual heterogeneity and local interactions drive global dynamics. ABM differs from traditional top-down modelling approaches by focusing on the individual components rather than attempting to directly model system-level behaviour through aggregate equations.

These models can incorporate spatial relationships, temporal dynamics, and stochastic elements that reflect real-world uncertainty and variability. Agents can adapt their behaviour based on past experiences, environmental conditions, or interactions with other agents, allowing for the study of learning, evolution, and adaptation in complex systems. This flexibility makes ABM particularly suitable for modelling biological systems where individual organisms exhibit diverse behaviours and strategies that collectively produce population-level patterns.

In biological research, ABM allows researchers to test hypotheses about individual-level mechanisms and observe their consequences at the population or ecosystem level, providing insights into phenomena such as species coexistence, evolutionary dynamics, and ecosystem stability. These models can incorporate realistic biological constraints such as energy limitations, reproductive cycles, and mortality rates, while allowing for controlled experimentation that would be impossible or unethical in natural systems. The ability to manipulate individual agent properties and environmental parameters makes ABM a powerful tool for exploring "what-if" scenarios and understanding the robustness of biological systems to perturbations.

3.5 Biomedical Applications of Agent-Based Modelling

3.5.1 Tumour Cell Response to Medicines

This application utilizes multi-agent systems to model cancer cells as heterogeneous agents with varying genetic profiles, drug sensitivities, and adaptive capabilities. These models simulate how cancer cells interact with therapeutic agents, healthy tissue, and the immune system within a dynamic tumour microenvironment. The hybrid approach allows researchers to model both the cellular agents and the evolving chemical gradients of drugs and nutrients, providing insights into drug resistance mechanisms, optimal dosing strategies, and combination therapy effectiveness. These simulations have accelerated cancer research by enabling virtual clinical trials, reducing the need for extensive animal testing, and helping identify promising treatment strategies before expensive human trials.

3.5.2 Microorganism Behaviour on Food Substrates

This model represents a fundamental application where agent-based models simulate bacterial colonies, fungi, or other microorganisms as they interact with nutrient-rich environments. These simulations model individual microorganisms as agents with behaviours such as chemotaxis, reproduction, and resource consumption, while the food substrate acts as a dynamic

environment that depletes over time. The hybrid nature of these models allows researchers to study how local interactions between microorganisms and their environment lead to complex spatial patterns, biofilm formation, and competitive dynamics. Such models have been instrumental in understanding antibiotic resistance development, optimizing fermentation processes, and predicting microbial contamination patterns in food safety applications.

3.5.3 Protein Folding Simulations

These employ agent-based approaches where amino acid residues act as individual agents with specific interaction rules based on their chemical properties. These hybrid models incorporate both the local interactions between adjacent residues and the global conformational constraints of the protein structure. The dynamic environment includes solvent molecules, temperature effects, and molecular crowding conditions that influence folding pathways. Such simulations have been crucial for understanding protein misfolding diseases, designing new proteins with specific functions, and developing therapeutic strategies for conditions like Alzheimer's and Parkinson's disease.

3.5.4 mRNA Vaccine Development

This application leveraged agent-based modelling to simulate immune system responses to novel vaccine formulations, dramatically accelerating the COVID-19 vaccine development process. These models represented immune cells, viral particles, and vaccine components as interacting agents within a dynamic physiological environment. The simulations enabled researchers to predict immune responses, optimize vaccine formulations, and identify potential side effects before clinical trials. The hybrid nature of these models, incorporating both cellular agents and molecular interactions, allowed for rapid iteration and testing of multiple vaccine candidates simultaneously, contributing to the unprecedented speed of vaccine development while maintaining safety and efficacy standards.

4 System Design and Architecture

4.1 Game of Life Implementation

Food in the simulation evolves dynamically using Conway's Game of Life, a cellular automaton in which grid cells change state (alive/dead) based on the number of live neighbours.

4.1.1 Mathematical Basis

Each cell $C_{x,y}$ follows these rules:

- **Survival**: If alive and has 2 or 3 neighbours, then stays alive.
- **Birth**: If dead and has exactly 3 neighbours, then becomes alive.
- **Death**: Otherwise, becomes or remains dead.

These rules are applied to a 2D matrix representing the environment every N simulation steps.

4.1.2 Metaphor

This adds an emergent, pseudo-biological food ecosystem — food grows, clusters, and dies off in patterns resembling algae blooms or microbial colonies.

4.1.3 Algorithm Steps

1. Traverse each grid cell.
2. Count its live neighbors.
3. Apply rules to determine next state.
4. Update the grid in-place or via a copy for stability.

4.2 Boids Implementation

The movement of bacteria in this simulation is governed by the Boids algorithm, originally formulated by Craig Reynolds to simulate flocking behaviour in birds. This decentralized model ensures that each agent (bacterium) follows simple rules based on its local environment, which collectively give rise to emergent swarm-like behavior.

4.2.1 Mathematical Basis

Each bacterium's new velocity $\vec{v}_{\text{new}}$ is deermined as a weighted combination of three vetor rules:

$$\vec{v}_{\text{new}} = w_a \cdot \vec{v}_{\text{alignment}} + w_c \cdot \vec{v}_{\text{cohesion}} + w_s \cdot \vec{v}_{\text{separation}} + w_f \cdot \vec{v}_{\text{food\_attraction}}$$

where:

- $\vec{v}_{\text{alignment}}$ - Match heading with neighbours.
- $\vec{v}_{\text{cohesion}}$ - Steer toward average position of neighbours.
- $\vec{v}_{\text{separation}}$ - Steer away from close neighbors.

- $\vec{v}_{\text{food\_attraction}}$ - Vector toward nearby food.
- w_a, w_c, w_s, w_f - Corresponding slider-controlled weights.

4.2.2 Metaphor

This simulates natural flocking, where bacteria instinctively follow group behavior without centralized control — reflecting real-world microbial motility patterns.

4.2.3 Algorithm Steps

1. For each bacterium, find neighbors within a perception radius.
2. Calculate alignment, cohesion, and separation vectors.
3. Add weighted food attraction vector if food is in range.
4. Normalize and cap the resultant velocity.
5. Update position using velocity.

4.3 Interaction Mechanisms

Interaction between bacteria and food, and among bacteria themselves, is modelled through biologically inspired rules like hunger, aging, reproduction, and death.

- Hunger - Bacteria lose energy every frame. If they reach a hunger threshold without consuming food, they die.
- Aging - Each bacterium has a life counter that increments every step. If it exceeds a max age, the bacterium dies naturally.
- Reproduction - Bacteria reproduce asexually when certain conditions (e.g., energy level, food nearby, low crowding) are met, cloning themselves with a fresh timer.

4.3.1 Metaphor

These mimic real-life microbial lifecycle dynamics — competition, survival, and self-replication in a resource-limited ecosystem.

4.3.2 Algorithm

1. At each frame, update hunger and age values.
2. If energy < threshold then mark for death.
3. If age > lifespan then mark for death.
4. If conditions met then spawn offspring.

4.4 Distribution Types

Initial placement of bacteria and food can follow different spatial distribution strategies:

- **Uniform Random** - Uniformly placed over the grid.
- **Clustered** - Gaussian blobs in specified zones.
- **Gaussian** - Placed towards the middle.
- **Linear** – Diagonal Gradient where intensity decreases diagonally.

4.4.1 Mathematical Basis

- Uniform random uses uniform(a, b)
- Clustered distribution uses normal(μ, σ)
- Others are based on weighted probabilities over coordinate zones.

These influence early-stage survival, competition, and the rate of ecosystem stabilization.

4.5 Parameter Controls

The simulation is highly tuneable via real-time sliders, allowing users to experiment with:

- Alignment Weight
- Separate Weight
- Cohesion Weight
- Food Attraction Weight
- Movement Speed

These values directly affect bacterial behaviour and can result in drastically different emergent dynamics (e.g., tight swarming vs. scattered foraging).

4.5.1 Metaphor

Parameter tuning is equivalent to genetic or environmental manipulation in biological systems — researchers can simulate mutation effects, environmental pressure, or ecological shifts without touching code.

4.5.2 Technical Details

Each slider maps a GUI element to an internal constant in constants.py, affecting real-time calculations during movement and interaction updates.

5 Methodology

5.1 Overview

This project adopts a modular, simulation-based methodology using Python and Pygame. The aim is to model complex bacterial behavior within a 2D environment that features essential biological dynamics such as hunger, aging, movement, and reproduction. Bacteria movement is governed by the Boids algorithm to emulate natural flocking behavior, while the food substrate evolves according to the rules of Conway's Game of Life, introducing emergent complexity into the ecosystem. This section details the system architecture, UI design, data visualization strategies, and the technical stack used.

5.2 Tech Stack

- Language – Python
- Graphics and Simulation – Pygame
- Data Handling – CSV, Matplotlib
- Visualization – Matplotlib for offline analysis and a real-time graphing system using Pygame

5.3 System Design

5.3.1 Repository Structure

```
/theaurafarmer/
├── core/                  # Contains all the core components
│   ├── bacterium.py       # Individual bacteria behavior (movement, hunger, aging)
│   ├── food.py            # Representation of food units using Conway's Game of Life
│   └── simulation.py      # Simulation manager, orchestration of steps and logic
├── data/                  # Data outputs from simulation
│   ├── bacteria_analysis.png # Offline graph visualizations
│   └── bacteria_stats.csv # Logged simulation data (population, deaths, etc.)
├── engine.py              # Entry point and main simulation loop
├── plot_stats.py          # Script to generate statistical plots from saved data
├── ui.py                  # Graph rendering and real-time visualization using Pygame
└── utils/
    ├── constants.py       # Configuration and tuning parameters
```

```
    ├── slider.py          # Interactive sliders for parameter control
└── vector.py              # Custom 2D vector math operations
```

5.3.2 Modular Design

- Core Module - Includes logic for bacteria (bacterium.py), food (food.py), and simulation control (simulation.py).
- UI & Control – the ui.py and slider.py files manage user interactions and display of real-time simulation data and graphs.
- Data Management – plot_stats.py processes CSV outputs for deeper statistical analysis.
- Utilities – constants.py and vector.py provide reusable values and vector math support.

5.4 UI Design

The user interface (UI) is designed to provide an interactive and intuitive experience for real-time observation and control of the simulation environment. Built entirely using Pygame, the UI serves multiple functions: parameter tuning, visual monitoring of the simulation state, and graphical representation of population trends.

5.4.1 Slider Controls

The simulation includes a dedicated slider-based interface that enables the user to dynamically modify the behavioural and physical properties of the bacteria in real time. This allows for hands-on experimentation with emergent behaviors arising from parameter changes.

The following sliders are available:

- **Alignment Weight** – Adjusts how strongly a bacterium attempts to align its direction with nearby bacteria. A higher value leads to tighter, more synchronized swarming.
- **Separation Weight** – Controls how forcefully a bacterium avoids crowding. Increasing this value helps reduce overlap or clustering, especially in dense areas.
- **Cohesion Weight** – Governs how much a bacterium is attracted to the center of nearby agents. This helps maintain the overall flock or colony structure.
- **Food Attraction Weight** – Influences how strongly bacteria are pulled toward nearby food units. Higher values result in more aggressive food-seeking behavior.
- **Movement Speed** – Sets the maximum velocity of the bacteria. A higher speed results in faster movement across the environment, which can affect food consumption rate, collision avoidance, and overall system dynamics.

Each slider is visually represented as a labeled horizontal bar with numerical indicators, allowing for fine-grained control. Real-time updates ensure that changes take immediate effect in the simulation, making the UI ideal for live testing and behavioral tuning.

5.4.2 Visual Feedback and Status Indicators

To provide the user with meaningful and immediate insight into the simulation's current state, the UI includes clearly labelled real-time visual indicators. These indicators are displayed

persistently on the simulation screen, allowing continuous monitoring without interrupting the experiment flow.

The following statistics are shown:

- **Steps** – displays the current frame or simulation step, which represents how long the system has been running.
- **Population** – shows the current number of living bacteria in the environment. This helps track survival, reproduction, and decline patterns in real time.
- **Food Cells** – indicates the number of food units currently available on the grid. Since food evolves dynamically, this count reflects the real-time state of the resource landscape.
- **Total Births** – tracks the cumulative number of bacteria that have been born since the start of the simulation. This helps quantify reproduction and population growth over time.
- **Total Deaths** – displays the total number of bacteria that have died, regardless of cause (e.g., starvation, aging). Useful for measuring mortality rate and stress in the system.

  These indicators are designed to update smoothly with every simulation step and are color-coded or grouped logically for readability. Their placement avoids cluttering the main visual field, ensuring that users can simultaneously observe both microscopic interactions and macroscopic trends.

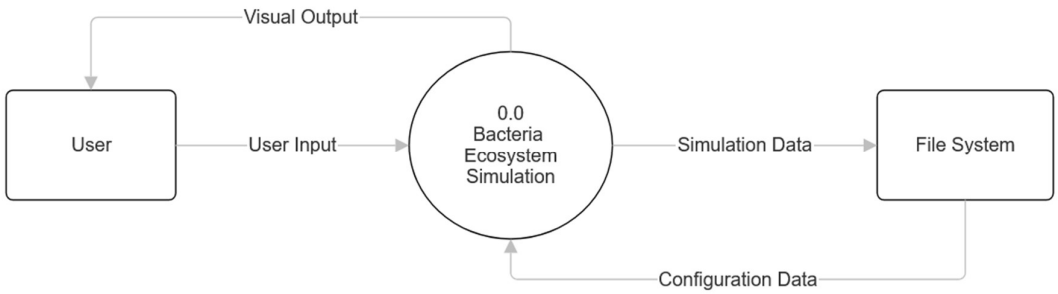
5.4.3 Graph Display Panel

The right or lower section of the screen hosts a real-time graph, plotting population changes over time. This graph updates dynamically with every simulation step and visually represents:

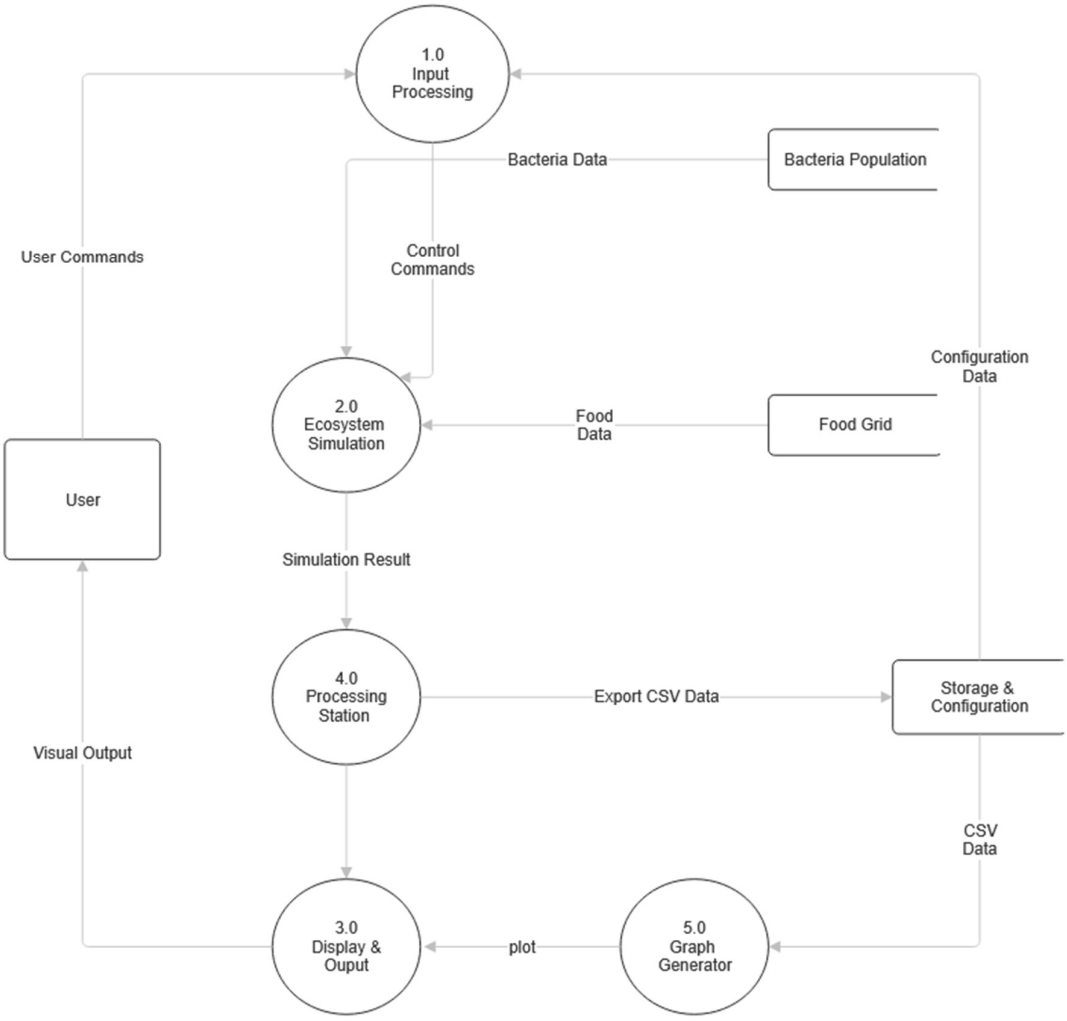
- Total population
- Available food units

The graph is built using a custom drawing system in Pygame for full integration and responsiveness.

5.5 Data Flow Diagrams



Level 0 Data Flow Diagram of the project

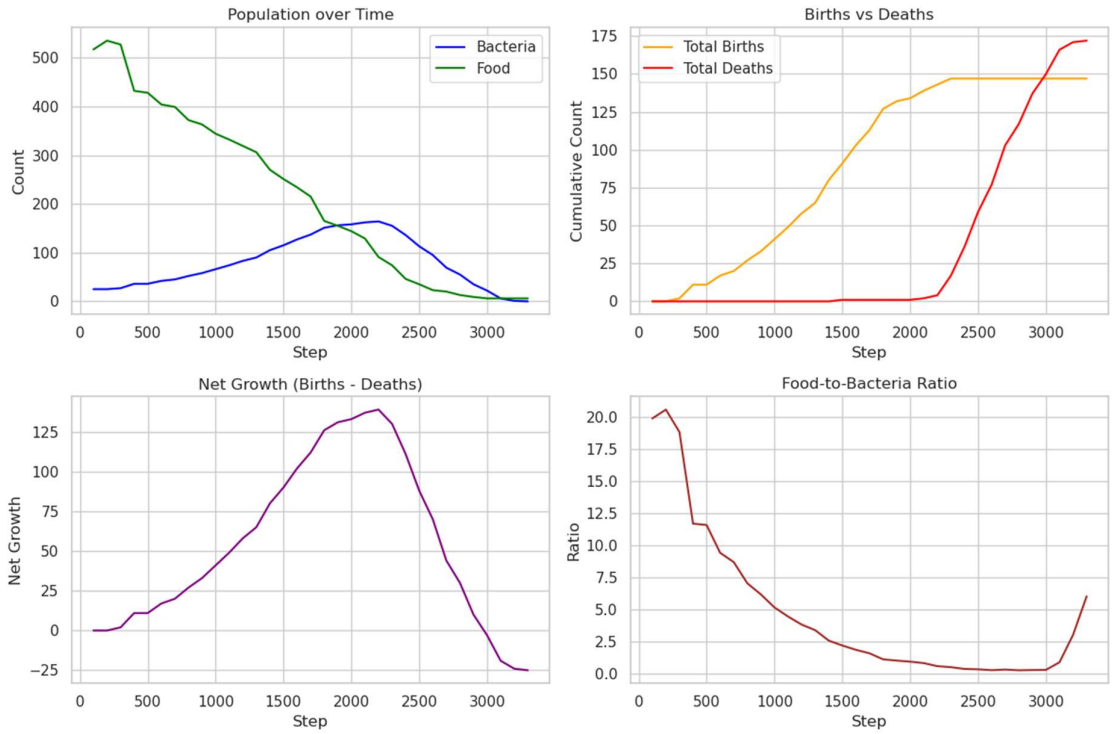


Level 1 Data Flow Diagram of the project

6 Comparative Study

6.1 Simulation Results & Observations

This section analyzes the performance of 25 bacteria agents (Boids) across different food distribution strategies — **Random**, **Clustered**, **Linear**, and **Gaussian**. The evaluation is based on population dynamics, food consumption behavior, and overall system efficiency.

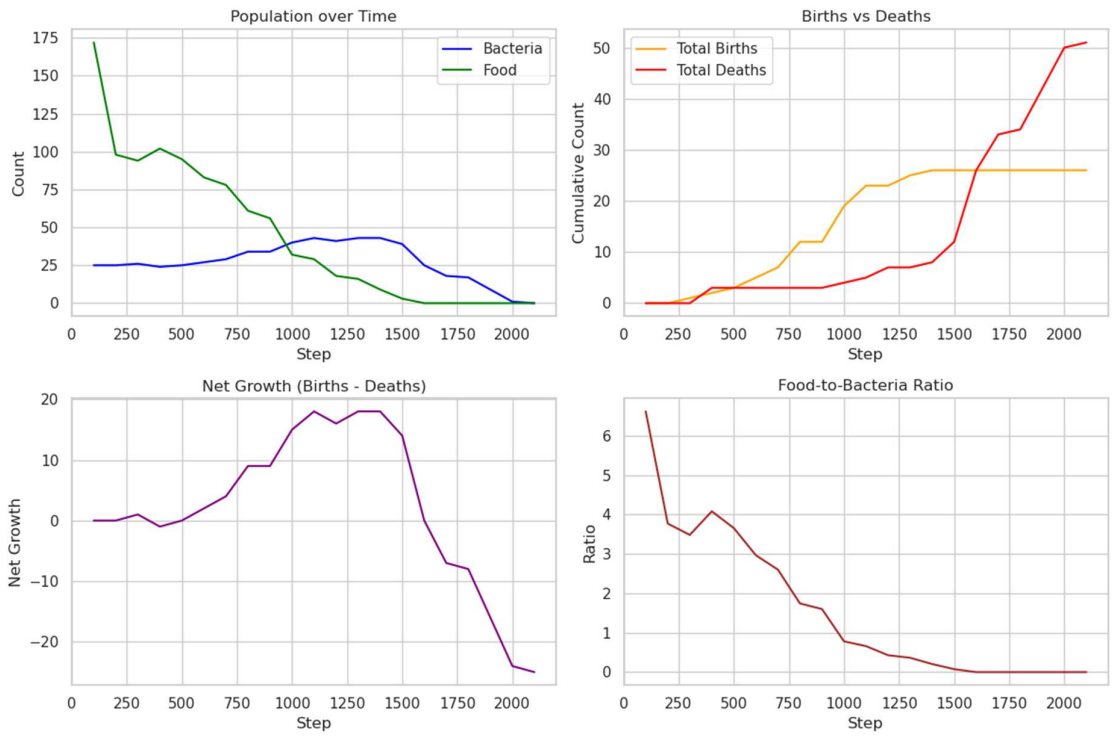


6.1.1. Random Distribution

- **Population Over Time**:
  A clear **inverse correlation** is observed between food availability and population growth. Once the number of bacteria surpasses the remaining food (evident from the intersection point on the graph), a **sharp population decline** begins due to starvation.

- **Food-to-Bacteria Ratio**:
  Starts around **20:1** and steadily drops to **zero**, indicating high initial food availability that gradually depletes. The spread-out nature of the food enables bacteria to explore and consume effectively for a longer period.

- **Takeaway**:
  Supports the **longest survival** with the **smoothest growth-decay curve**, providing an ideal balance for exploration and resource access.

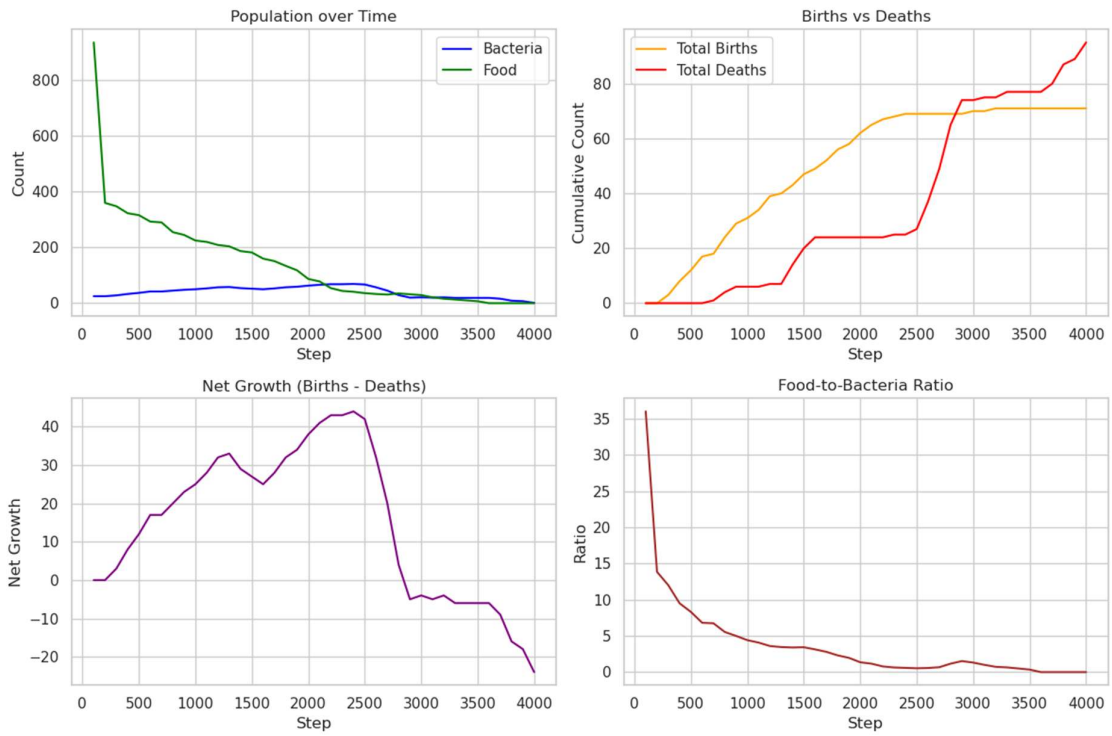


6.1.2. Clustered Distribution

- **Population Over Time**:
  Initial food availability is **lower than in Random**, resulting in a **slower population increase**. Bacteria benefit from the clustered layout early on but quickly face scarcity.

- **Efficiency**:
  The **tumor-like food blobs** are consumed rapidly and efficiently, leading to **faster early population growth**, but the ecosystem collapses when these clusters are depleted.

- **Takeaway**:
  Short-term survival is efficient, but long-term sustainability is limited. Bacteria struggle once clusters vanish, as remaining food becomes sparse.

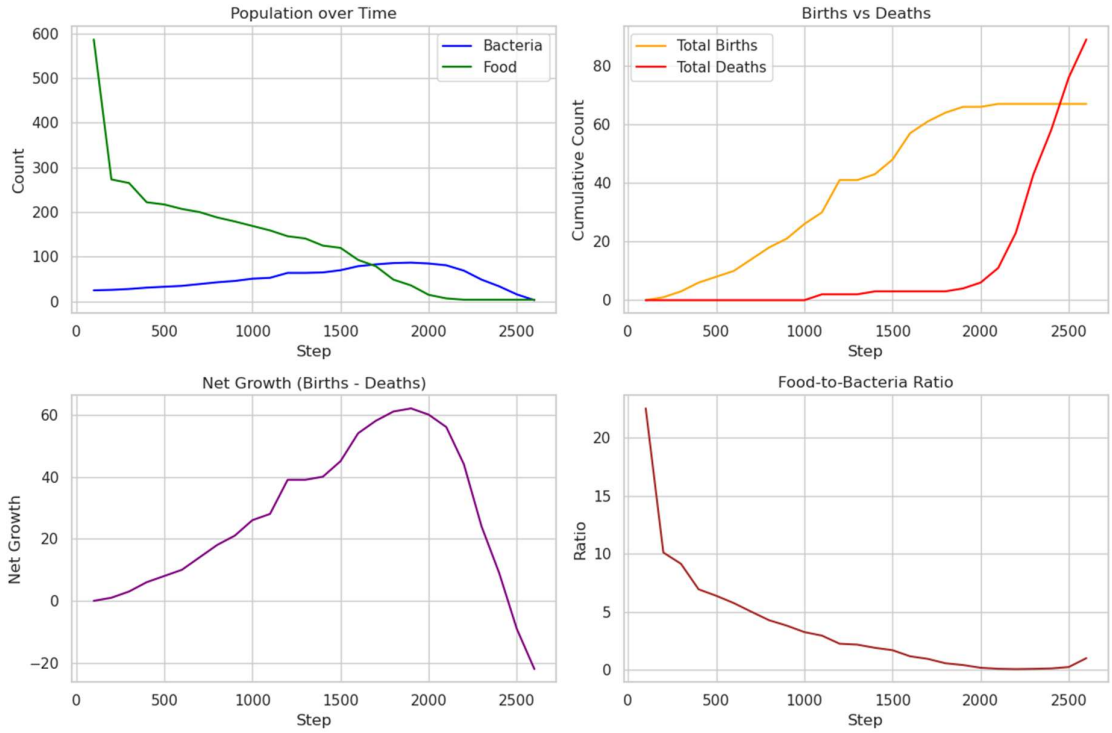


6.1.3. Linear Distribution

- **Population Over Time**:
  Exhibits a **controlled and gradual population growth**, with bacteria adapting to food arranged along a linear or edge-biased path.

- **Food-to-Bacteria Ratio**:
  Starts around **35:1**, but food is consumed in a relatively uniform rate. The pattern leads to more **predictable but limited** foraging behavior.

- **Takeaway**:
  Least favorable for long-term sustainability. The linearly distributed food doesn't encourage enough spatial variation for swarm dynamics, leading to **early depletion** and **gradual decline**.



6.1.4. Gaussian Distribution

- **Population Over Time**:
  A **rapid decline in food** is noted at the beginning, as the central dense area is quickly consumed. Bacteria thrive initially but struggle when food shifts to peripheral zones.

- **Accessibility**:
  Central concentration of food allows for **early growth**, but **corner-located food becomes hard to reach**, especially with decaying energy or older agents.

- **Takeaway**:
  Peaked early performance followed by a harsh decline. Accessibility is great at the start but drops off significantly over time.

6.2 Insights & Biological Metaphor

- **Random** represents a balanced ecosystem, favoring **exploration** and **sustainable growth**.

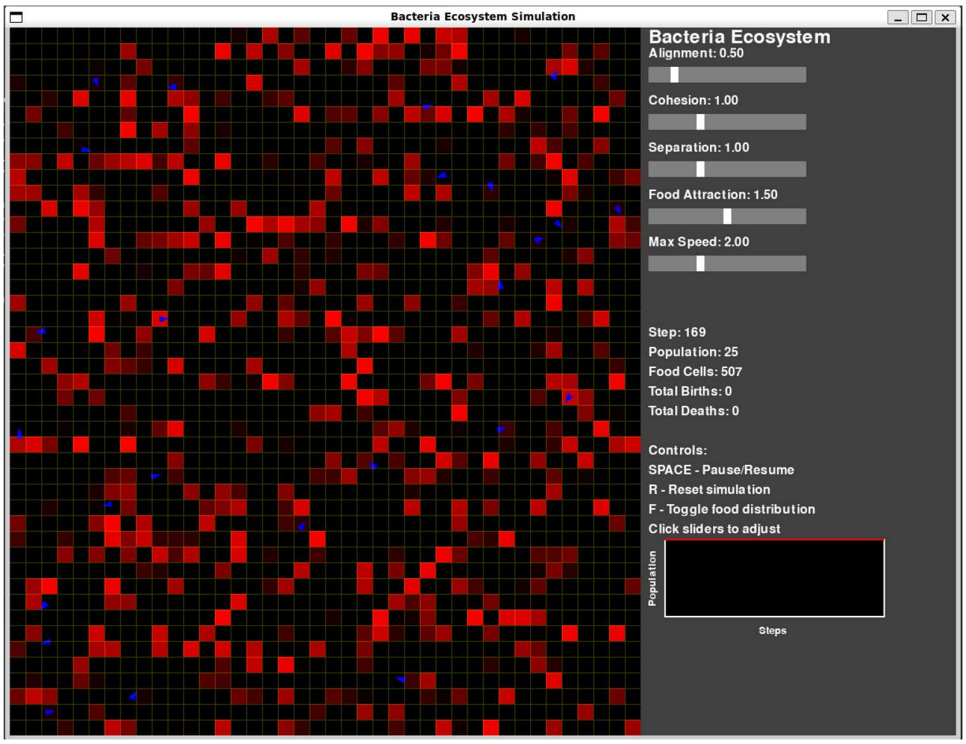
- **Clustered** mimics **tumor-like nutrient pockets**, useful for studying burst-phase behavior and starvation.

- **Linear** is analogous to **canal/vein-like nutrient delivery**, efficient but limited in promoting swarm adaptation.

- **Gaussian** mirrors **centralized resource systems**, leading to early booms but poor edge exploitation.

7 Implementation

```
sayan@Y1PLUSNEO:~/work/theaurafarmer/v4$ python3 engine.py
pygame 2.6.1 (SDL 2.28.4, Python 3.12.3)
Hello from the pygame community. https://www.pygame.org/contribute.html
Enter the number of bacteria : 25
Enter the food index 0
```



Random Distribution

8 Scope of Future Research

This simulation framework provides a foundational model for microbial motion and resource interaction using simple agent-based rules. To align with cutting-edge applications in synthetic biology, chemotaxis modeling, and targeted drug delivery, several avenues of future research are proposed:

1. Chemotactic Behavior Modeling

Future iterations could incorporate gradient-based chemotaxis, where agents respond to simulated chemical gradients rather than discrete food patches. This would more accurately mimic real microbial behaviors such as E. coli swimming toward higher nutrient concentrations.

- Gradients can be dynamically updated in the environment based on agent movement and chemical diffusion.
- Mathematical modeling using partial differential equations (PDEs) for diffusion fields can be layered onto the simulation for greater realism.

2. Synthetic Biology Simulations

The framework can be adapted to model engineered bacteria used in synthetic biology, including:

- Toggle-switch behavior: Simulating gene expression states toggled by environmental triggers.
- Kill-switch mechanisms: Where agents die under certain synthetic regulatory thresholds.
- Biosensors: Bacteria that move toward and fluoresce in the presence of specific environmental signals.

This enables the exploration of programmable bacteria in silico before real-world wet lab trials.

3. Intelligent Drug Delivery Systems

In targeted drug delivery research, bacteria and nanoparticles are often modeled as micro-swimmers navigating toward tumors or infection sites. This simulation can be extended to:

- Represent drug-laden synthetic bacteria as agents.
- Introduce virtual tumor zones as food proxies.
- Simulate delivery efficiency using success metrics such as time-to-target and payload accuracy.

Such models can serve as digital testbeds for precision medicine prototypes.

4. Image-Based Initialization via Computer Vision

One of the most promising future directions involves using image processing techniques (e.g., OpenCV, scikit-image) to generate real-world-based initial food distributions.

- Microscopic images of real tissue samples (e.g., tumor structures, tissue scaffolds) can be processed to extract food placement masks.
- This would create bio-realistic environments derived from histological images, enabling more relevant synthetic biology experimentation.
- Combined with chemotaxis, bacteria could be tested for their ability to navigate complex tissue environments digitally.

5. Phototaxis and Light-Responsive Behavior

In addition to chemotaxis, future agents can be programmed to respond to light gradients (phototaxis), useful in designing light-controlled synthetic organisms. This would simulate how engineered microbes could be externally guided through optical control systems.

9 Conclusion

This project presents a modular and extensible simulation framework that models bacterial behavior using swarm intelligence, nutrient dynamics, and biologically inspired lifecycle mechanics. By integrating Boids-based movement, Conway's Game of Life for food evolution, and a rich set of interaction rules such as hunger, aging, and reproduction, the simulation effectively captures the emergent complexity of microbial ecosystems in silico.

Through carefully designed visualizations and data tracking, the simulation reveals how environmental structures — such as random, clustered, linear, and Gaussian food distributions — drastically influence population growth, survival rates, and behavioral outcomes. Comparative analyses show that boid behavior is not only reactive but also highly sensitive to spatial resource patterns, demonstrating clear parallels to real microbial chemotaxis and collective adaptation.

The project's modular architecture, built in Python with Pygame, provides a flexible base for further research, offering sliders for real-time tuning and CSV-backed analytics for empirical insights. The ability to simulate lifelike population dynamics and visualize them step-by-step creates a valuable educational and experimental tool for researchers and students alike.

Moreover, this simulation lays the groundwork for future work in synthetic biology, drug delivery modeling, and chemotactic navigation, especially when integrated with image processing for biologically accurate environments. By bridging the gap between algorithmic modeling and biological plausibility, this work offers a compelling digital sandbox to explore, test, and visualize complex behaviors in artificial life systems.

In conclusion, this project is not just a simulation of virtual bacteria — it is a platform for exploring the emergent beauty of decentralized intelligence, resource-driven survival, and the untapped potential of digital life sciences.

References

1. Reynolds, C. W. (1987). *Flocks, Herds, and Schools: A Distributed Behavioral Model*. Presented at ACM SIGGRAPH '87. This seminal paper introduced the Boids algorithm for simulating natural flocking behavior onestepguide.net+11red3d.com+11github.com+11.

2. "Boids." *Wikipedia*. Overview of Craig Reynolds' 1986 artificial life simulation, describing separation, alignment, and cohesion rules onestepguide.net+6en.wikipedia.org+6beforesandafters.com+6.

3. Conway, J. H. (1970). "The Game of Life." *Scientific American*, October 1970 issue. Popularized by Martin Gardner, this cellular automaton explores emergent complexity from simple rules opastpublishers.com+4conwaylife.com+4beltoforion.de+4.

4. Conway's Game of Life. *Wikipedia*. Defines the survival, birth, and death rules for grid-based evolution red3d.com+14en.wikipedia.org+14onestepguide.net+14.

5. Caballero, L., Hodge, B., & Hernandez, S. (2016). *Conway's "Game of Life" and the Epigenetic Principle*, Frontiers in Microbiology. Discusses its heuristic use in modeling ecological and biological systems frontiersin.org+1complexsystemstheory.net+1.

6. Brown, J., et al. (2007). *Navigational Control of Bacteria: The Design of a Synthetic Chemotactic Biological System*. BMC Systems Biology. Demonstrates synthetic biology control of E. coli chemotaxis bmcsystbiol.biomedcentral.com.

7. Advances in Bacterial Chemotaxis: Navigational Control and Drug Delivery. *Science of the Total Environment* (2024). Highlights engineered bacteria for targeted therapies journals.asm.org+8sciencedirect.com+8sciencedirect.com+8.

8. Moore, L., & Emonet, T. (2024). *Chemotaxis: From Microbial Movement to Immune and Developmental Roles.* Trends in Microbiology. Provides modern insights into gradient sensing mechanisms cell.com+1biologyinsights.com+1.

9. Micali, G., & Endres, R. G. (2015). "Bacterial Chemotaxis: Information Processing, Thermodynamics, and Behavior." *arXiv*. Reviews molecular foundations and behavioral thermodynamics arxiv.org+1sciencedirect.com+1.

10. Liebchen, B., & Löwen, H. (2018). "Synthetic Chemotaxis and Collective Behavior in Active Matter." *arXiv*. Studies chemotactic principles applied to synthetic micro-swimmers arxiv.org.