,,

Universiteit Leiden
The Netherlands

Data Science & Artificial Intelligence

Enhancing Biodiversity Stability:

Adaptive Parameter Optimization in Abstract Ecosystem Simulations

Oumnia Chaara

Supervisors:
Richard M.K. van Dijk & Marcello Bonsangue & André Deutz

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl
09/07/2025

Abstract

This thesis is part of the LIACS LUDev project, which explores how simulated ecosystems can serve as ground-truth references to support biodiversity monitoring. The goal is to improve the evaluation of real-world tools such as camera traps, eDNA, and acoustic sensors. Within this context, the research investigates how ecological parameters can be adjusted to maintain biodiversity stability over time. This is done through adaptive optimization in agent-based simulations, where species-level traits are updated based on the system's current state to reduce extinction and improve survival.

The main research question is: *'How can adaptive optimization of species traits help maintain biodiversity stability in agent-based ecosystem simulations?'* To answer this, the study follows a two-phase approach. In Phase 1, an abstract ecosystem model was developed to simulate key ecological dynamics such as energy flow, trophic interactions, energy conservation, and spatial structure. The model was evaluated under different configurations to test whether expected ecological patterns emerged. In Phase 2, a feedback-based optimization module, the Balancer, was introduced to dynamically adjust species parameters based on biodiversity indices.

Results from Phase 1 showed that the model reproduced core ecological behavior. Energy transfer and trophic stability were observed, with population dynamics responding realistically to changes in agent density, grid size, and energy constraints. Agents clustered in high-resource areas, and lower-level species showed greater stability, while top-level predators were more prone to extinction. In Phase 2, the Balancer improved biodiversity stability in both balanced and stressed ecosystems. In stable scenarios, it maintained higher diversity levels compared to runs without optimization. In unbalanced setups, it helped delay collapse and supported partial recovery through adaptive trait updates. Over multiple trials, adaptive updates led to improved diversity indices and more consistent population dynamics.

This work contributes to the field of agent-based modeling by integrating ecological simulation with adaptive parameter tuning. Organisms are modeled as agents grouped into species levels, interacting through feeding, movement, reproduction, and energy exchange. The model evolves through feedback-based adjustments of level traits, enabling exploration of biodiversity resilience under varied ecological pressures without relying on real-world data.

Acknowledgments

I would like to thank my supervisors, Richard M.K. van Dijk, Marcello Bonsangue, and André Deutz, for their valuable guidance, feedback, and support throughout this project.

Contents

1 Introduction

1.1 Background

Biodiversity is declining globally, raising concerns about long-term ecosystem stability. Understanding how species interact and persist under changing conditions is a central problem in ecology.

Studying biodiversity directly in the field is often difficult. Ecosystems are complex, and observations are limited by time, environmental variability, and incomplete data. In most cases, there is no ground truth available to validate or compare outcomes. This limits our ability to understand how ecosystems change over time.

Simulation modeling provides a controlled environment to study these dynamics. It allows researchers to define specific conditions and observe how ecological patterns emerge under those conditions [Murphy et al., 2020].

Agent-Based Modeling (ABM) simulates individuals (agents) interacting with each other and their environment through simple rules. ABMs help explore population-level outcomes based on local interactions and behaviors.

However, ABMs depend heavily on model parameters, which influence outcomes such as biodiversity and ecosystem stability. Choosing and tuning these parameters manually can be difficult.

To address this, recent work has combined machine learning (ML) and optimization techniques to automatically adjust simulation parameters. ML models can learn the relationship between parameters and ecological outcomes, while optimization algorithms search for parameter sets that improve biodiversity or system stability.

This thesis builds on this by designing and implementing an agent-based simulation model to study biodiversity and ecosystem stability. It then applies machine learning combined with optimization techniques to adaptively adjust species-level parameters, aiming to improve biodiversity outcomes through continuous tuning during the simulation.

1.2 LUDev Project

This thesis is part of the LUDev project[1], a collaborative project between researchers and students from LIACS, coordinated by Richard van Dijk. The project aims to study biodiversity monitoring using sensor-based observations in the field, such as human sightings, camera traps, environmental DNA (eDNA), and acoustic recorders. These observations are location-based and help estimate biodiversity in real ecosystems.

A core part of the project is building a simulated ecosystem that serves as a ground-truth model to evaluate the accuracy of these monitoring methods. This thesis supports LUDev by working

[1] https://ludevprojectbidding.com/project-details.php?category=other&id=biodiversity

on the optimization of parameters in an agent-based simulation. The aim is to design an abstract ecosystem model that shows realistic biodiversity behavior over time and assess how it evolves through parameter optimization. This simulation can then be used as a reference to evaluate how well different monitoring methods perform, both in the simulation and in real environments.

1.3 Objectives

Existing research has focused mainly on simulating simplified ecosystem relationships involving specific predator-prey dynamics, such as the well-known wolf-sheep-grass [Wilensky, 1997] and rabbit-fox [Hussein, 2013] models. These studies often define a set of common parameters and interaction rules to study the behavior of the system. To analyze biodiversity and stability over time, researchers typically adjust initial parameters, compare simulation outputs to real-world ecological data, or use mathematical models like the Lotka–Volterra equations [Korobeinikov, 2009].

However, these models typically include only one predator and one prey species, which limits the representation of the complexity of the ecosystem. In contrast, real ecosystems involve a wider variety of species and interactions that affect ecological balance and stability.

Building on this previous work, this thesis focuses on adaptive parameter optimization as a way to support biodiversity stability in a simulated ecosystem. The objectives of this thesis are the following:

- To design an abstract ecosystem [Austin and Cook, 1974] through a network of interacting classes, that share common parameters and actions;
- To simulate the evolution of this ecosystem over time using Agent-Based modeling;
- To implement an adaptive optimization process using a defined objective function based on biodiversity stability, integrated within the simulation loop to adjust selected parameters during runtime;
- And to examine how these adaptations affect biodiversity and stability throughout the simulation by running different scenarios and comparing them.

1.4 Research Questions

The main research question is: ‘How can adaptive optimization of species traits help maintain biodiversity stability in agent-based ecosystem simulations?’ Sub-questions include:

SQ1 How can an abstract agent-based ecosystem be designed to simulate biological dynamics such as energy flow and species behavior?

SQ2 How does adaptive optimization of species parameters affect biodiversity stability compared to no optimization?

This supports LUDev’s goal of using simulations as ground-truth to evaluate real-world biodiversity monitoring tools.

1.5 Thesis Overview

This thesis is structured as follows. Section 2 reviews related research in agent-based modeling, biodiversity indices, and adaptive optimization. Section 3 presents the two-phase methodology used to answer the main research question, starting with the design of an abstract ecosystem simulation, followed by the implementation of an adaptive parameter optimization module. Section 4 evaluates the results of both phases through a series of simulation experiments. Section 5 discusses the findings, while Section 6 outlines potential directions for future research. Finally, Section 7 summarizes the overall contributions of the thesis.

2 Related Work

2.1 Foundations of Ecological Simulation Modeling

2.1.1 Model Development Process

Simulation modeling is a method used to represent a real-world system in a simplified and controlled way. It helps researchers study how the system works and test different scenarios without needing to experiment in the real world [Borshchev, 2013].

Building a simulation model involves several key steps. These steps help ensure that the model is useful, understandable, and represents the system accurately. Based on [Centeno and Carrillo, 2001] and [Montevechi et al., 2015], the main steps include:

- **Problem formulation**: Define the goal of the simulation and identify the key parts of the system that need to be included.
- **Choosing the level of abstraction**: Decide how detailed the model should be. This depends on the system and the research question.
- **Selecting the simulation approach**: Choose a modeling method, such as System Dynamics, Discrete Event Simulation, or Agent-Based Modeling [Ngo-Hoang, 2020]. Each approach fits different types of problems.
- **Implementation**: Convert the design into code using a simulation tool or programming language.
- **Verification and validation**: Check that the model is correctly built (verification) and that it matches the real-world behavior it is supposed to represent (validation).

These steps are important to consider in ecological modeling, where systems are complex and difficult to measure directly. In this thesis, they guide the development of an agent-based model to explore species behavior and ecosystem dynamics.

2.1.2 Applications and Model Selection Criteria

Simulation modeling is used in many fields like military planning, business forecasting, public policy, social sciences, and artificial intelligence [Luan, 2024]. Different fields need different levels of detail in their models depending on how complex the system is and what the model is intended for. For example, some models are very detailed and focused on specific tasks, while others are more general and cover bigger, more complex systems.

Ecosystems are very complex because they involve many species interacting in unpredictable ways. [Ngo-Hoang, 2020] show that ecosystems usually need models that can handle this complexity by using high-level abstractions. To address this complexity, there are several modeling approaches commonly used, including System Dynamics, Discrete Event Modeling, and Agent-Based Modeling (ABM) [Ngo-Hoang, 2020]. Choosing the right model depends on the system's complexity and the study's goals. To help with this choice, researchers use decision frameworks that explain when each modeling approach works best. For example, [(Letcher et al., 2013] offer a framework for environmental studies that highlights ABM as a good fit when modeling individual agents and studying their interactions is important. This fits well with ecological research on biodiversity and species relationships (see Figure 1)

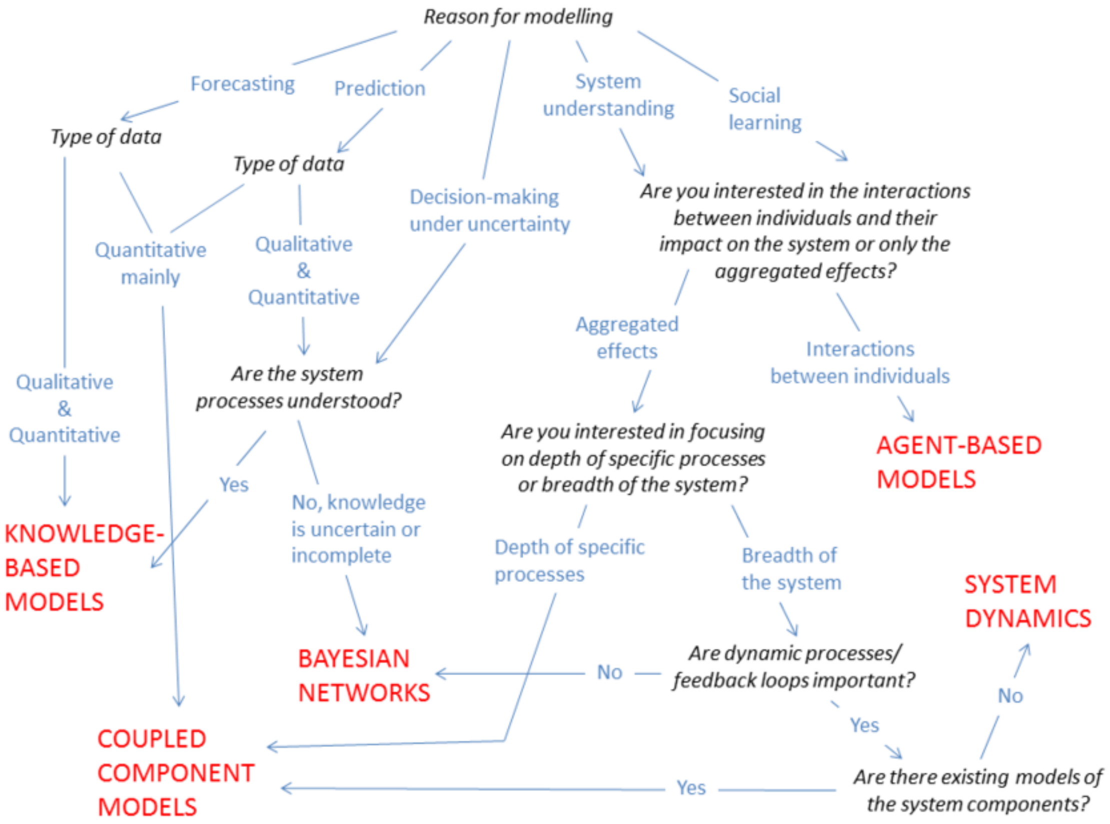

Reason for modelling
Forecasting
Prediction
System understanding
Social learning
Type of data
Type of data
Decision-making under uncertainty
Are you interested in the interactions between individuals and their impact on the system or only the aggregated effects?
Quantitative mainly
Qualitative & Quantitative
Aggregated effects
Interactions between individuals
Qualitative & Quantitative
Are the system processes understood?
AGENT-BASED MODELS
Are you interested in focusing on depth of specific processes or breadth of the system?
Yes
No, knowledge is uncertain or incomplete
KNOWLEDGE-BASED MODELS
Depth of specific processes
Breadth of the system
SYSTEM DYNAMICS
BAYESIAN NETWORKS
No
Are dynamic processes/ feedback loops important?
No
Yes
COUPLED COMPONENT MODELS
Yes
Are there existing models of the system components?

Figure 1: Decision framework for model selection (adapted from [(Letcher et al., 2013]), highlighting that Agent-Based Modeling (ABM) is most suitable for modeling systems with individual agents and understanding their interactions.

2.1.3 Challenges in Ecological Simulation Modeling

Simulation modeling faces several key challenges. The complexity increases exponentially as the number of parameters and possible actions grows, known as the curse of dimensionality. Many ecological processes are stochastic, which adds uncertainty to simulation results [Luan, 2024]. Validating models against real-world systems can be difficult, especially when data is limited. Building accurate models requires both technical skills and detailed ecological knowledge. Additionally, simulations are scenario-based tools—they do not directly optimize system performance or solve problems automatically. Instead, their results depend heavily on the quality of input data and the assumptions built into the model. Simulations also cannot represent system properties that are not explicitly included in the model, which limits their scope and usefulness [Centeno and Carrillo, 2001].

2.2 Agent-Based Modeling (ABM) in Ecological Simulations

Agent-Based Modeling (ABM) has become more widely used in ecological studies in recent years. This is reflected in the growing number of applications. For example, [Nugroho and Uehara, 2023] show that ABM is increasingly used to explore the interactions between humans and nature. Similarly, [Ozkal et al., 2025] emphasizes its rising importance in environmental research.

ABMs offer valuable insights into how individual behaviors influence ecological systems. Classic models include the Wolf-Sheep-Grass and Rabbit-Fox predator-prey model, and were starting points for more complex ecological models [Wilensky, 1997, Hussein, 2013].

Recent studies show the value of ABM in real-world scenarios. For example, [Neil et al., 2025] applied ABM to rewilding at the Knepp Estate in the UK. It simulates the behaviors and interactions of herbivores and their effects on vegetation. The study shows that ABMs can support decision-making in biodiversity restoration and habitat management. Similarly, [Purathekandy et al., 2024] modeled human-elephant conflict in India using movement data from radio-tagged elephants. The simulation showed how factors like rainfall, body temperature, and starvation influenced elephant behavior and crop raiding, providing insights for wildlife management. [McLane et al., 2011] also reviewed multiple ABM studies to understand how animals like Eurasian nuthatches and woodland caribou choose habitats and move through landscapes.

An ABM typically includes three core components: agents, interactions, and the environment. Agents are individual entities with specific attributes and behaviors. Interactions define how agents affect each other, and the environment provides the space where this process occurs. This allows researchers to explore how simple agent-level rules lead to complex system-level patterns [Macal and North, 2011]

ABMs offer several advantages. According to [Bazghandi, 2012], they can capture emergent behaviors that aren't directly programmed, describe real-world systems in a natural way, and offer flexibility to scale complexity as needed. They are also relatively cost-effective since they reduce the need for extensive field studies.

However, ABMs come with challenges. As discussed earlier in Section 2.1.3, simulating real-world complexity, especially when involving human or social behaviors, can be difficult. Models need a clear purpose to avoid being overly generic. In addition, performance issues like long runtimes or software limitations may arise, especially in large-scale simulations [Bazghandi, 2012, Murphy et al., 2020].

2.3 Parameter Optimization Methods in Ecological Simulation Models

Ecological simulations, especially Agent-Based Models (ABMs), involve complex dynamics controlled by many parameters. Optimizing these parameters is essential to improve outcomes such as biodiversity. This section reviews existing work on: (i) machine learning approaches, (ii) simulation-based optimization methods, and (iii) their combination to enhance simulation results.

2.3.1 Machine Learning Approaches

Machine learning has been increasingly integrated into ecological and agent-based models to improve prediction and realism. For example, [Majic et al., 2025] integrated XGBoost into a tourism-focused ABM to simulate tourist movement patterns more accurately. Traditional ABMs often rely on hardcoded rules or simple heuristics to guide agent behavior, which may not reflect real-world dynamics. Instead, they used a supervised ML model trained on real tourist check-in data to predict the next point of interest a tourist might visit based on their visit history. These predictions were then used to update the agent's decisions at each timestep, effectively replacing fixed decision rules with learned behavior patterns. This allowed the simulation to better capture how agents (tourists) move through the environment.

In this thesis, a similar approach is used by training ML models to capture the relationship between input parameters from the simulation and the resulting biodiversity indices. The goal is to predict biodiversity outcomes based on simulation data, allowing the model to learn patterns that would be difficult to define manually.

However, a key limitation of ML approaches is that they require large, high-quality datasets for training and cannot actively explore which parameter configurations yield the best outcomes. This limitation motivates the integration of optimization techniques.

2.3.2 Simulation-Based Optimization Methods

Simulation-based optimization[2] offers a way to actively explore the parameter space to identify optimal configurations. This is particularly useful in ecological modeling, where systems are often nonlinear, high-dimensional, and computationally expensive to evaluate. Additionally, simulation models often contain stochastic elements, meaning repeated runs with identical inputs can yield different results.

Rather than relying on a known mathematical form, simulation-based optimization methods use feedback from the simulation itself to iteratively adjust parameters and guide the search process. [Carson and Maria, 2000] categorize these methods into seven types, such as gradient-based techniques, response surface methods, and metaheuristics. In ecological contexts, metaheuristic approaches , especially evolutionary algorithms, have been the most widely adopted due to their robustness in handling complex, nonlinear systems.

Genetic algorithms (GAs) are the most commonly used evolutionary algorithm in ecological modeling. For example, [Calvez and Hutzler, 2005] applied GAs to efficiently search large parameter spaces in an ecological agent-based model, and [Broniec et al., 2021] enhanced this by integrating knowledge-based approximations to reduce simulation runs. [Keller, 2012] applied multi-objective optimization using Pareto front analysis to balance competing goals such as biodiversity and resource consumption.

[2] https://en.wikipedia.org/wiki/Simulation-based_optimization

In addition, Differential Evolution (DE) has gained popularity as an alternative evolutionary algorithm. [Karterakis et al., 2007] describe DE as “one of the most promising novel Evolutionary Algorithms in terms of efficiency, effectiveness, and robustness,” positioning it alongside GAs and particle swarm optimization as a strong candidate for simulation-based optimization.

Other metaheuristics such as particle swarm optimization (PSO), Markov Chain Monte Carlo (MCMC), and chaos game optimization (CGO) have also been applied. [Vlad et al., 2024] compared several of these techniques and evaluated their performance across models with varying complexity and levels of stochasticity.

A persistent challenge for all these methods is the high computational cost, as many simulation runs are typically needed to evaluate and compare candidate solutions.

2.3.3 Hybrid Machine Learning and Optimization Approaches

While both ML and simulation-based optimization have their strengths, each has limitations when used in isolation. ML can capture complex input-output relationships and make rapid predictions, but it cannot search for new input configurations. In contrast, optimization algorithms can explore the parameter space but are computationally demanding when applied directly to full simulations.

To address these limitations, hybrid approaches combine ML with optimization. For example, [Testolina et al., 2019] proposed a framework that integrates a machine learning surrogate model with a parameter optimizer. Instead of applying optimization directly to the full simulation, they first trained an ML model on simulation data to approximate the objective function. This was then used to rapidly evaluate candidate solutions during optimization, significantly reducing computational cost.

Their study tested various regression models as surrogates, including linear regression, Gaussian process regression, support vector regression (SVR), and random forests. For optimization, they used gradient-free methods such as genetic algorithms, which are effective for noisy or complex landscapes.

Inspired by this, the approach used in this thesis adopts a similar strategy. Simulation runs from the ABM are used to train an ML model, such as XGBoost, to learn the relationship between input parameters and biodiversity outcomes. The trained model then serves as a fast evaluator in the simulation-based optimization process, enabling efficient exploration of the parameter space. In this way, ML provides rapid approximations, while the optimizer generates new candidate solutions, combining speed with effective search.

2.4 Biodiversity Indices in Ecosystem Modeling

Biodiversity is an important concept in ecology and can be defined in different ways. It refers to the variety of life in a given area, including the number of species (richness) and how individuals are spread across those species [Begon et al., 2006] [3]. These basic measures help describe biodiversity,

[3] https://www.cbd.int/convention/articles?a=cbd-02

but they do not capture the full picture of how balanced or uneven an ecosystem is.

For this reason, more informative indices are often used. The Shannon index [Shannon, 1948] was originally developed in information theory to measure entropy, which quantifies uncertainty in communication systems. Ramon Margalef was one of the first ecologists to suggest applying this concept to ecology [Margalef, 1957], showing that entropy can describe species diversity by including both richness and evenness. As discussed by [Sherwin and Prat i Fornells, 2019], Margalef's work helped introduce entropy-based approaches into ecology, making the Shannon index a common measure of biodiversity.

The Simpson index [Simpson, 1949] was developed mainly for ecological studies. It measures the probability that two individuals picked at random belong to the same species. This index gives more weight to the most common species in a community (dominance).

Both indices describe biodiversity, but they emphasize different aspects. The Shannon index reflects how evenly individuals are spread across species and is more sensitive to rare species. The Simpson index, on the other hand, gives more weight to common species and reflects dominance. Using both indices provides a more complete picture of species diversity in ecological studies.

For example, [Su et al., 2024] used Shannon and Simpson indices to study biodiversity across plants, rodents, bacteria, and fungi. They combined these into a multitrophic biodiversity index (MBI) to explore how diversity at different trophic levels affects ecosystem functioning and resilience. A similar approach is also taken in Section 3.3.3, where multiple biodiversity indices are weighted and combined into a single objective score for prediction and optimization.

However, not all biodiversity indices perform equally well. [Kitikidou et al., 2024] compared seventeen different indices and found that only a few consistently reflected changes in biodiversity. This shows the importance of choosing metrics based on the specific aim of the study. [Jones and Laughlin, 2009] also demonstrated that biodiversity outcomes depend not only on which index is used, but also on how data are sampled. This is especially relevant when biodiversity measures are used as part of a model or decision-making system.

In line with this, [Carnus et al., 2015] emphasize that stability and diversity should be assessed in ways that match the ecological question being addressed. This reinforces the idea that biodiversity metrics are not universal and should be chosen based on the specific goals of the model.

3 Methods

The methodology is structured around two main phases, each addressing a key component of the abstract ecosystem model and its optimization. These phases directly correspond to the sub-questions outlined in this study:

- **Phase 1: Simulation Execution**
  Addressing the first sub-question: SQ1, this phase involves the design and implementation of an agent-based simulation that models ecological interactions. It begins with defining an abstract model structure and continues with the initialization of agents, classes, and environment parameters. Agents, representing individual organisms, are assigned behaviors such as movement, feeding, reproduction, and death. These behaviors drive the simulation dynamics over time, resulting in energy flow and population changes.

- **Phase 2: Balancer Module**
  This phase addresses sub-question SQ2 by explaining how the Balancer module adapts the ecosystem during the simulation. After running for several timesteps, the module uses system feedback to update agent parameters. It employs an ML regression model to predict those parameters, such as energy costs for movement and reproduction chances, that could improve biodiversity. These predictions are optimized using differential evolution and applied as mutations to a subset of agents. This allows the ecosystem to gradually adapt while the simulation evolves.

Together, these phases provide a structured framework for analyzing how abstract ecosystems behave and how adaptive strategies can influence their long-term stability.
The full code used to implement this methodology is available in the GitHub repository[4].

3.1 Tools, Software, and Frameworks for Ecological Simulations

3.1.1 ABM Framework: Agents.jl

There are several tools and frameworks available for building and simulating agent-based models (ABM), each offering distinct features for ecological simulations. These frameworks help researchers design, run, and analyze complex models where agents interact within a defined environment.

NetLogo [Wilensky, 1997] is a widely used platform with a user-friendly interface, written in Java and Scala. It allows for quick model creation and is accessible to users of all levels, featuring a rich library of pre-built models and visualization tools.

MASON [Luke et al., 2005], a Java-based framework, is popular for its scalability and flexibility, handling large and complex models efficiently. It's especially useful for high-performance simulations, accommodating large-scale research.
Mesa [Masad and Kazil, 2015] is a Python-based framework offering an easy-to-use environment for creating ABMs. It comes with core components for model management, data collection, and

[4] https://github.com/LiacsProjects/BioDiversityMonitor

analysis, as well as browser-based visualization through Solara.

Agents.jl [Vahdati, 2019], is based on Julia, providing the simplicity of Python with the performance of C/C++. Its speed and scalability make it ideal for large, fast simulations, supporting scientific research that requires efficient execution without sacrificing ease of use.

In a recent comparison of ABM frameworks [Datseris et al., 2021], Agents.jl was highlighted for its minimal code complexity and excellent performance, making it ideal for computationally demanding simulations. For this thesis, Agents.jl (Julia) was chosen due to its combination of high performance and ease of use, making it well-suited for the complex and dynamic simulations required for optimizing ecosystem models.

3.1.2 Balancer Implementation

The Balancer module combines machine learning and simulation optimization to guide parameter updates during ecosystem dynamics. The following tools support this integration:

- **XGBoost (Python):** A machine learning library for gradient-boosted decision trees. It builds an ensemble of trees to model complex relationships in data. In this work, it predicts beneficial parameter adjustments based on simulation features and outcomes. [Chen and Guestrin, 2016]
- **Optuna (Python):** An automatic hyperparameter optimization framework. It efficiently searches the parameter space to improve model performance by minimizing validation error. Here, it tunes key XGBoost parameters. [Akiba et al., 2019]
- **Differential Evolution (Python):** A population-based evolutionary algorithm used for optimization. It explores diverse parameter settings within the forking strategy, evaluating candidates on biodiversity indices in simulated scenarios. [SciPy Developers, 2025]
- **PyCall (Julia-Python Bridge):** A Julia package that allows calling Python functions from Julia code. This enables seamless integration of the Python-based ML components within the Julia simulation environment. [Contributors, 2025]

3.2 Phase 1: Simulation Execution

This phase outlines how the simulation model was constructed and operated. The model represents an abstract ecosystem composed of three trophic levels. Each level is defined by specific classes, agent behaviours, and parameter configurations. These components form the structural basis for simulating ecosystem dynamics.

The model was implemented using the Julia programming language and mainly the **Agents.jl** framework (see Section 3.1.1). This framework was chosen because it enables scalable and efficient agent-based modelling, well-suited to the structure of ecological systems.

It begins by initializing the environment and assigning parameters to each agent. After that, the main simulation loop begins. During every time step t, every agent performs a fixed sequence of actions: moving, feeding, reproducing, and dying. These actions are based on predefined rules which depend on the agent's internal state and its surrounding conditions.

After each time step, the simulation updates the ecosystem. Energy is redistributed, agent statuses are revised (e.g., movement or reproduction), and the overall system responds to evolving dynamics. Data collected at each step is later used in Phase 2, where parameters are optimized through a balancer module to reach ecological stability.

3.2.1 Abstract Ecosystem Design

In ecology, an ecosystem consists of different species, each with its own characteristics and role within the system. Based on the framework proposed by Austin [Austin and Cook, 1974], this model constructs a simplified, abstract version of an ecosystem to enable flexible and controlled simulation of ecological dynamics.

We define the abstract ecosystem as a directed-graph-based model. Agents are grouped into species classes, and each class is assigned to a trophic level. Edges between classes define one-way feeding relationships, which determine how energy flows and predation occurs.

The purpose of this abstraction is not to recreate a specific real-world ecosystem, but rather to represent key ecological processes in a generalized form. This approach allows us to explore how population dynamics, adaptation, and energy flow emerge from basic rules, without being tied to the constraints of a specific species or habitat.

The model is built around three core ecological concepts: trophic structure, feeding interactions, and energy flow. These are simplified but essential for modeling how agents interact, consume resources, and maintain themselves in a structured environment.

Each node in the graph represents a species class. Each class contains multiple agents, which represent individual organisms. Directed edges between nodes define which classes can feed on which, meaning predation occurs only in one direction. This is illustrated in Figure 2, which shows a sample network with five classes arranged across three trophic levels. In the model:

- Nodes correspond to classes, which represent general types of species.
- Each class contains multiple agents, which act as individual organisms.
- Edges represent predation links, where one class consumes agents from another, but not vice versa.

The ecosystem is divided into three trophic levels:

- Level 1 (base): classes that generate energy, such as producers.
- Level 2 (middle): classes that feed on Level 1 and are consumed by higher-level classes.

- Level 3 (top): classes that feed on the lower levels and are not consumed by others.

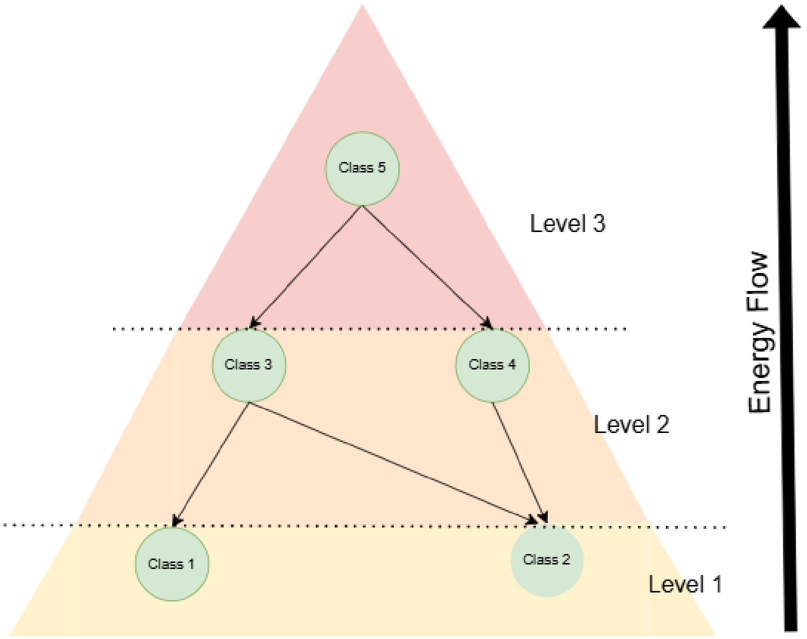


Figure 2: Abstract ecosystem shown as a directed interaction network. The five nodes represent species classes, arranged across three trophic levels. For example, nodes 1 and 2 belong to Level 1. Arrows between nodes indicate unidirectional feeding links, where one class consumes agents from another. The side arrow indicates the overall energy flow, moving from producers at Level 1 to top consumers at Level 3.

Three levels are used because most food chains in nature are short. As energy is lost at each step, ecosystems rarely support more than three or four levels [Borrelli and Ginzburg, 2014]. Three levels also allow both bottom-up and top-down effects to be observed, without introducing unnecessary complexity.

Multiple species classes can exist within each trophic level. All agents share the same set of behavioural parameters, such as energy cost for movement, reproduction cost, and death probability. However, the actual values of these parameters depend on the trophic level the agent belongs to.

This setup is based on how organisms typically behave in real ecosystems. Agents in lower levels are given lower energy needs and higher reproduction rates, since producers and herbivores tend to grow quickly and reproduce often. Agents in higher levels are given higher energy needs and lower reproduction rates, because top consumers usually grow slowly and need more resources to survive.

As a result, trophic levels behave differently in the simulation. Lower levels maintain larger populations, while higher levels have fewer agents due to their slower growth and higher energy cost.

During simulation, energy flows from Level 1 to Level 3, following the feeding links. Agents at higher levels depend on the availability of agents below them to survive.

The parameter values assigned to each trophic level are not fixed.In Phase 2, a parameter-balancing algorithm will be used to adjust these values. The goal is to allow the ecosystem to reach a stable state, where all species can coexist over time.

3.2.2 Class and Agent Definitions

To make this abstract structure more functional , we define classes and agents in more detail.

Class Parameters (Species) Each class belongs to one of three trophic levels, as shown in Figure 2. Multiple classes can be assigned to the same trophic level, meaning they share a common set of initial parameters. Table 1 shows these shared parameters for each level. However, classes behave differently due to their positions in the interaction network, random initial placement, and the probabilistic actions of their agents. This introduces variety for classes belonging to the same level as well.

Table 1: Initial parameter values assigned to agents by trophic level. Each level is associated with specific energy costs and behavior probabilities: movement energy cost (E_{move}), reproduction energy cost (E_{rep}), movement probability (m_id), birth probability (r_id), and death probability (d_id). These values influence agent dynamics within the simulation.

Level	**Move Energy Cost** (E_{move})	**Reproduction Energy Cost** (E_{rep})	**Movement Probability** (m_id)	**Birth Probability** (r_id)	**Death Probability** (d_id)
1	0.02	0.1	0.4	0.55	0.08
2	0.1	0.25	0.5	0.35	0.15
3	0.25	0.4	0.7	0.25	0.25

All parameter values lie within the normalized range [0,1] to standardize interpretation. Energetic parameters (e.g., E_move = 0.1) represent the proportion of energy lost per action. Probability parameters (e.g., reproduction r_id, movement m_id, death d_id) define the chance of those events at each time step, controlling the agent behavior and ecosystem dynamics. These values are set by trophic level and will be fine-tuned automatically in Phase 2 using a parameter-balancing algorithm.

The class parameters in this model are based on the actions each agent can take: moving, feeding, reproducing, or dying. Each of these actions requires one or more parameters to define when and how it occurs. For example, movement requires an energy cost and a probability of occurring, while reproduction is defined by its own energy cost and reproduction probability. Death is handled as a natural probability, unless an agent runs out of energy—in which case it dies immediately.

This structure is inspired by previous modeling work. For instance, studies such as [Hussein, 2013], [Colon et al., 2015], and [Niemann et al., 2021] use rates or probabilities to define birth, death, and reproduction events in predator–prey models. In other studies, especially those that simulate real-world ecosystems, parameters are often drawn from empirical observations or sensor data. For example, [Murphy et al., 2020] used values like predator speed, prey density, detection distance, weight, and age based on real-world measurements.

Energy costs for movement and reproduction were included to reflect the biological reality that animals must spend energy to perform these actions. As shown by [Malishev and Kramer-Schadt, 2021], integrating energy budgets captures real-world trade-offs: if an agent moves or reproduces too often, it risks running out of energy. This prevents agents from acting without limits and leads to more realistic population dynamics, where only individuals with sufficient energy can afford costly behaviors.

In contrast to models that simulate specific ecosystems, this model describes an abstract environment that generalizes ecological systems. It is not designed to replicate a particular species or habitat, but it reflects ecological principles such as energy trade-offs, population dynamics, and trophic interactions. The goal is not to recreate a real-world scenario in detail, but to explore how changes in class parameters affect species adaptation, mutation, and the stability of biodiversity over time.

Agent Structure Each agent represents an individual organism and maintains its own internal state and behavior. It inherits the parameters from its class but also tracks individual variables to represent its current state within the simulation. The individual variables include the current energy level E_i_id, E_gain_id energy gained so far, E_loss_id, energy lost so far, E_loss_env energy lost to the environment so far and the behavioral flags: moved, eaten , reproduced, and has_died. The structure of an individual agent is shown in Listing 1.

Listing 1 Agent structure in Agents.jl model, including ID, position, class ID, energy states, and behavioral flags for each simulation step.

```
@agent struct Individual(GridAgent{2})
  class_id::Int              # links to AgentClass.id
  E_i_id::Float64            # current energy level
  E_gain_id::Float64         # energy gained so far
  E_loss_id::Float64         # energy lost so far
  E_loss_env::Float64        # energy lost to environment so far
  moved::Bool                # has the agent moved at this step?
  eaten::Bool                # has the agent eaten at this step?
  reproduced::Bool           # has the agent reproduced at this step?
  has_died::Bool             # is the agent dead at this step?
end
```

This structure allows energy balance to be tracked per agent. The variables E_gain_id, E_loss_id, and E_loss_env together provide a breakdown of how much energy an agent gains and loses over time. This was done to monitor whether energy intake and energy loss are consistent with basic energy conservation principles. A similar approach is used by [Semeniuk et al., 2012], where energy gain and loss were tracked to check if simulated behaviors led to realistic survival and reproduction outcomes.

Agents receive a unique ID and (x,y) positions automatically via *Agents.jl* framework. The class_id links each agent to its class, which maps to a trophic level as defined in Listing 2.

Listing 2 Definition of the `ClassLevel` struct

```julia
struct ClassLevel
  level::Dict{Int,Int}  # Maps class ID --> trophic level (1=base, 2=middle, 3=top)
end
```

Boolean flags like *moved*, *eaten*, and *reproduced* are reset at every time step to monitor the agent's activity. In contrast, the *has_died* flag remains true once an agent dies. These flags are used for validation and allow full data retention throughout the simulation. Without this approach, using Agents.jl's built-in dead agent function would remove dead agents from the model entirely and lose their data. This hinders full traceability and analysis. With agents and classes defined, the model requires a spatial setup to simulate interactions.

3.2.3 Environment Setup and Initialization

Agents are randomly placed on a two-dimensional Euclidean grid with a periodic topology. This means the grid wraps around edges. Agents moving off one side reappear on the opposite side. This design prevents agents from being restricted by edges, allowing uninterrupted movement throughout the simulation area. Multiple agents can occupy the same grid cell, facilitating direct interactions such as predation and competition.

Each agent is assigned a random location and initial parameter values based on its class. Internal states, including energy levels and behavioral flags, are set according to the agent structure. This ensures agents are ready for simulation from the first timestep.

3.2.4 Simulation Step

After defining the species classes and individual agents, the next important component of the agent-based model is the simulation step, where agents interact and update their states over discrete time steps.
At each time step, every agent performs one or more of the following actions:

- **Move:** An agent moves with probability m_{id}. Each move costs energy equal to E_{move} multiplied by the agent's current energy. This energy is deducted from the agent's reserve. The probability adds variation in agent behavior. The energy cost prevents agents with low energy from moving too much.

- **Feed:** When a predator and prey share the same grid cell, the predator feeds on the prey (see Listing 3). The predator gains all the prey's energy. The prey's energy is set to zero, it is marked dead, and removed from the simulation. This rule requires agents to be in the same location and keeps energy transfer straightforward.

- **Reproduce:** Agents reproduce with probability r_{id}. Reproduction costs the parent agent an amount of energy equal to E_{rep} times its current energy. This energy is transferred to the offspring, which inherits the parent's parameters and behaviors. The reproduction probability limits excessive reproduction, while the energy cost ensures only agents with enough energy reproduce.

- **Die:** Agents die if their energy drops below zero or due to a natural death probability d_{id}. Upon death, an agent's energy is set to zero and it is permanently marked as dead. This combines energy-based death with random natural mortality.

Listing 3 Definition of an interaction network representing feeding relationships between agent classes. In this example, Class 3 feeds on Classes 1 and 2, Class 4 feeds on Class 2, and Class 5 feeds on Classes 3 and 4. These relationships are represented using a dictionary structure.

```
interaction_network = InteractionNetwork(Dict(
    3 => [1, 2],  # Class 3 interacts with Classes 1 and 2
    4 => [2],     # Class 4 interacts with Class 2
    5 => [3, 4]   # Class 5 interacts with Classes 3 and 4
))
```

These actions together describe how energy moves through the ecosystem. Movement, feeding, reproduction, and death cause energy to be shared or lost, which affects how populations change over time. The following section explains how energy flows and updates are handled.

3.2.5 Updates in the Simulation

At each timestep, energy states are updated based on agent actions. The system tracks:

- $E_{\text{i\_id}}$: current energy of the agent

- $E_{\text{gain\_id}}$: energy gained by the agent

- $E_{\text{loss\_id}}$: energy lost and transferred to other agents

- $E_{\text{loss\_env}}$: energy lost to the environment

This tracking mechanism ensures that energy is conserved across the system, except for controlled losses to the environment. It maintains ecological balance and prevents unrealistic energy accumulation.

In real ecosystems, energy lost by one organism is often reused by others, such as microbes or plants recycling waste. In our setup, for simplicity, lost energy is only redistributed to neighbors during the same timestep. If no neighbors are present, the energy is lost to the environment and not stored for later use.

Energy updates per action

- Movement: Agents move with probability m_{id}. When moving, the agent loses energy proportional to its current energy:

$$E_{\text{loss\_env}} \mathrel{+}= E_{\text{move}} \times E_{\text{i\_id}}$$

This energy is lost to the environment because it represents the cost of movement.

- Feeding: When predator i and prey j share a grid cell, the predator can consume the prey:(see Listing 3)

$$E_{\text{gain\_id}} \mathrel{+}= E_{\text{j\_id}}$$
$$E_{\text{j\_id}} = 0, \quad \text{has\_died}(j) = \text{true}$$

The predator gains all the prey's energy. The prey dies and no energy is lost to the environment during feeding.

- Reproduction: At each timestep, agents reproduce with probability r_{id}. When reproduction happens:

$$E_{\text{loss\_id}} \mathrel{+}= E_{\text{rep}} \times E_{\text{i\_id}}$$

(parent)

$$E_{\text{gain\_id}} = E_{\text{rep}} \times E_{\text{i\_id}}$$

(offspring)
The parent loses a portion of its energy, which becomes the offspring's starting energy. The new agent inherits class parameters and is placed randomly on the grid. Changing r_{id} and E_{rep} controls reproduction frequency and energy passed to offspring.

Death: Agents die if:

- Their energy falls below zero, or
- A natural death event occurs with probability d_{id}.

Upon death:

$$E_{\text{loss\_id}} \quad \text{(energy redistributed)} \quad \text{or} \quad E_{\text{loss\_env}} \quad \text{(if redistribution fails)}$$

The `has_died` flag is set permanently.

Energy transferred from dead agents or reproduction is redistributed to agents in the same or adjacent grid cells. The distribution is weighted by trophic level, with higher-level agents receiving a larger share, reflecting their greater energy needs as in real ecosystems. If redistribution is not possible due to lack of neighbors or rounding limits, the residual energy is added to E_loss_env. The full implementation of the update and redistribution logic is provided in Appendix A

3.2.6 Data Collection

After each simulation step, we collect data in preparation for Phase 2, during which the balancer module will be trained. The dataset contains, for each timestep, trophic level, and agent class, a set of parameters: `E_move_id`, `E_rep_id`, `m_id`, `r_id`, and `d_id`, along with the biodiversity index. Details on the biodiversity indices are provided in Section 3.3.3.

Data is collected at fixed intervals (e.g., every 3 steps) to continuously monitor the ecosystem's state over time. Although data is gathered regularly, parameter updates based on this data are applied only after a cooldown period. This cooldown is a predefined number of simulation steps (e.g., 3 steps) during which no updates occur. It ensures the system has enough time to evolve between changes and prevents updates that are too frequent or noisy, which could destabilize the simulation. Moreover, this threshold filters out minor fluctuations so that adjustments are made only in response to meaningful shifts, improving the effectiveness of training the balancer module.

This process uses a *forking* approach, illustrated in Figure 3. The current simulation state is cloned into multiple independent forks. Each fork explores a distinct set of parameter adjustments guided by two key measures:

- **Deviation severity:** a normalized measure of how far the biodiversity index lies outside or near the boundaries [0.55, 0.75].
- **Local survival penalty:** the deviation of each agent class's survival rate from the optimal target of 0.6.

For each fork, parameters are adjusted proportionally to these measures and then clamped within the valid range [0, 1]. The forks are simulated independently to completion, after which the biodiversity index and relevant metrics are recorded.

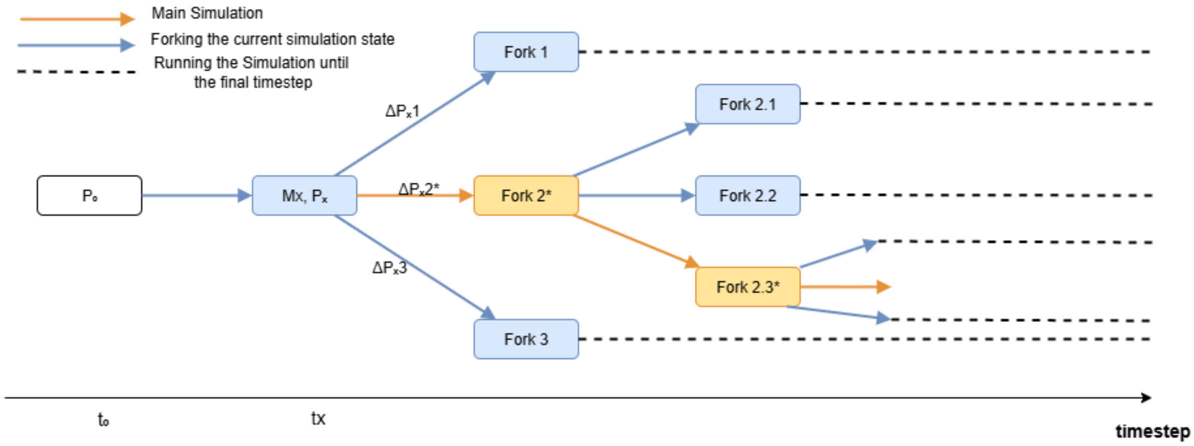


Figure 3: Diagram of the forking-based data collection process. When biodiversity falls outside the target range, the simulation forks into multiple variants with adjusted parameters that run until the final timestep. The fork with the best outcome (orange) updates the main simulation and continues forward.

After all forks complete, the fork with the highest final biodiversity index is selected. The main simulation updates its parameters to match this best-performing fork and continues running.

This enriches the dataset used for machine learning in Phase 2, improving the model's ability to learn how parameter changes affect biodiversity. As a result, the Balancer can make more effective optimization decisions.
The corresponding logic is outlined in the pseudocode below:

Algorithm 1 Data collection using forking strategy

```
if step % Data_Collection_Interval = 0 then
    collect_data(update = (cooldown = 0))
    if cooldown = 0 then
        cooldown ← Cooldown_Period
    end if
end if
if cooldown = 0 then
    Calculate severity based on biodiversity index deviation from [0.55, 0.75]:
        $$\text{severity} = \begin{cases} \frac{0.55-\text{bio\_index}}{0.2} & \text{if bio\_index} < 0.55 \\ \frac{\text{bio\_index}-0.75}{0.2} & \text{if bio\_index} > 0.75 \\ \frac{|\text{bio\_index}-0.65|}{0.2} & \text{otherwise} \end{cases}$$
    if severity > 0 then
        For each impact factor in $[0.3, 0.55, 0.8]$:
            Fork simulation, adjust parameters by combined severity and survival penalty, simulate
            and record results
        Select best fork and update main simulation
        cooldown ← Cooldown_Period
    else
        Continue simulation normally
    end if
else
    Continue simulation normally
    cooldown ← max(cooldown - 1, 0)
end if
```

Importantly, this parameter adjustment step uses simple rule-based methods solely to introduce variation in input parameters and generate diverse training data. It does not perform parameter optimization. The actual optimization is performed later by the Phase 2 balancer module, which uses machine learning on this varied training data input.

3.3 Phase 2: Balancer Module

This phase focuses on implementing the Balancer Module, which uses an adaptive optimization process to improve biodiversity during the simulation. The module adjusts agent-level parameters based on feedback from previous timesteps. It includes data preparation, a sliding window function for train/test input, and an objective function based on biodiversity indices. An XGBoost model is trained with hyperparameter tuning using Optuna to learn the relationship between input parameters and biodiversity outcomes. Once trained, the model is used as a surrogate evaluator within a differential evolution process that searches for parameter sets predicted to yield high biodiversity. These optimized parameters are then applied in the next simulation run. This creates a feedback loop between simulation, prediction, and parameter adjustment to improve outcomes over time.

3.3.1 Triggering Mechanism During Simulation

Data collection, as described in Section 3.2.6, continuously occurs at fixed intervals. Parameter updates, however, are applied only when the cooldown period has ended (cooldown_period = 0), ensuring the system has time to adjust before further changes.

The balancer module runs every **ml-interval** step (e.g., every 4 steps) but activates only when the machine learning cooldown timer has expired (ml_cooldown == 0). Once the balancer module identifies improved parameters, it applies these updates and resets the cooldown timer (ml_cooldown = 3), preventing immediate reactivation and allowing the ecosystem time to stabilize before the next optimization cycle.

The balancer module and data collection operate independently, so both can occur within the same simulation step if their conditions are met. This ensures continuous data gathering while regulating parameter updates through separate timing and cooldown mechanisms, maintaining system flexibility without blocking either process.

By controlling the timing of updates in this way, the system avoids premature or overlapping changes, ensuring each adjustment is based on stable, representative data that reflects the ecosystem's true state.

3.3.2 Data Preparation

1 - Reshaping Data for Machine Learning: Data is collected from multiple simulation forks, each representing a sequence of discrete timesteps. At every timestep, 5 input features are recorded per class across 3 levels, along with a biodiversity index that serves as the target variable to predict and maximize.

Since the raw data is organized by fork, timestep, level, and class, it is inherently multi-dimensional. However, the goal is to optimize parameters per level (not per class), so the data is simplified by randomly selecting one class per level at each timestep. This reduces complexity while retaining representative features, balancing information richness with dimensionality.

Because the biodiversity index is a global measure recorded once per timestep (not per level), the reshaped data includes this single target value alongside the selected input features from all levels. This results in a combined feature-target vector of length:

$$L \times F + 1 = 3 \times 5 + 3 = 16$$

where L is the number of levels, F is the number of input features per level, and the $+1$ accounts for the single biodiversity index. A sliding window is then applied across the data matrix to maintain alignment between features and targets.

Since XGBoost requires a 2D input matrix with each row as a sample and columns as features, the time dimension is flattened. For each fork (sample), all timesteps are concatenated into a single flat vector of length $T \times 16$, where T is the number of timesteps.
Final input shape:

$$\text{XGBoost input} = (X, T \times 16)$$

Example:
For a fork with 3 timesteps, the input vector is:

$$[\text{param1}_{l1,t}, \ldots, \text{param5}_{l1,t}, \ldots, \text{param5}_{l3,t}, \text{bio}_t, \text{param1}_{l1,t+1}, \ldots, \text{bio}_{t+1}, \ldots]$$

This vector captures all features and the biodiversity index over time as a single sample for training.

2 - Sliding Window: After reshaping the data, during the machine learning process, the collected data is split into training and testing sets using an 80-20% ratio. Instead of using data from all timesteps, a sliding window approach focuses on relevant time segments.

The window captures w_{train} steps before an update point for training, and w_{test} steps after for testing. Here, both are set to 4. Since data is collected every three timesteps, this ensures the training and testing sets include the immediate surroundings of each update, in terms of parameter values and biodiversity outcomes. This helps the model track evolving behavior without being biased by older data.

As the simulation runs, the window moves forward step-by-step, updating the training and testing segments dynamically. This keeps the model focused on recent, local patterns rather than older data that may no longer represent the current system state. Since targets are stored with features, the sliding window extracts both together while keeping alignment.

To prevent data leakage and overfitting [Majic et al., 2025], splits are made at the level of collected forks (i.e., simulation scenarios). When a fork is in the training set, it is excluded from testing, ensuring the model is evaluated on completely unseen data.

Finally, within each split, the data is separated into feature matrices (X_{train}, etc.) and target vectors (y_{train}, etc.) by selecting the feature columns only for inputs and the biodiversity index columns for targets.

3.3.3 Objective Function

1 - Biodiversity Indices: Biodiversity indices are used to monitor ecosystem state and guide the balancer module during simulation. At each timestep, three normalized indices are computed to capture distinct aspects of diversity:

- **Richness** — number of unique species (classes)
- **Dominance** — concentration of individuals across trophic levels
- **Evenness** — distributional balance across levels

The simulation includes three fixed trophic levels, while the number of species (classes) can increase over time through mutation. This distinction influences how metrics are computed:

- **Richness** is calculated at the class level to reflect species diversity.
- **Simpson's and Shannon's indices** are computed across trophic levels to capture ecological function rather than raw species count.

The three main normalized indices used[5] are:

- Normalized Richness (R')
- Shannon's Evenness (E_h) [Pielou, 1966]
- Normalized Gini-Simpson Index ($D' = 1 - D$)

Normalized Richness (R') Counts the number of living species S, normalized by the maximum possible number $S_{\max}$:

$$R' = \frac{S}{S_{\max}} \tag{1}$$

Normalized richness R' ranges from 0 (no species) to 1 (all possible species present).

Shannon's Evenness (E_h) First compute Shannon entropy:

$$H' = -\sum_{i=1}^{L} p_i \ln(p_i), \quad \text{where } p_i = \frac{n_i}{N}$$

This captures how evenly individuals are distributed across trophic levels. Then normalize:

$$E_h = \frac{H'}{\ln(S)} \tag{2}$$

where S is the number of species (classes). E_h ranges from 0 (uneven) to 1 (perfectly even distribution).

[5] https://en.wikipedia.org/wiki/Diversity_index

Gini-Simpson Index (D') Measures diversity across trophic levels, defined as the complement of Simpson's Index D:

$$D' = 1 - D \tag{3}$$

$$D = \sum_{i=1}^{L} \frac{n_i(n_i - 1)}{N(N - 1)}$$

where n_i is the number of individuals in level i, N is the total population, and L is the number of trophic levels. Lower D values indicate higher diversity; D' increases with diversity.

Survival Rate per Class In addition to the core indices, we track class-specific survival over time. For each class j, the survival rate is:

$$\text{Survival}_j = \frac{\text{AliveCount}_j}{\text{AliveCount}_j + \text{DeadCount}_j}$$

This helps monitor how well each class persists during the simulation.

2 - Objective Function: The three normalized indices defined in Equations 1, 2, and 3 are combined into a single weighted objective used for prediction and optimization:

$$\text{Objective: Combined Diversity Score} = w_1 R' + w_2 E_h + w_3 D' \tag{4}$$

where w_1, w_2, w_3 are weights that define the relative contribution of each index. We set $w_1 = 0.2$, $w_2 = 0.35$, and $w_3 = 0.45$, with the weights summing to 1.

These values were selected based on repeated testing. Richness (R') received the lowest weight because it only changes when a new class appears or an existing one goes extinct. While it reflects species count, it does not capture how individuals are distributed or how dominant certain levels are.

Shannon's Evenness (E_h) was given moderate weight. It increases when populations become more evenly spread across levels, which often happens near the end of the simulation due to high mortality. In such cases, evenness can increase even if diversity decreases.

The Gini-Simpson index (D') was given the highest weight. It captures how individuals are concentrated across trophic levels and responds to shifts in dominance more effectively than the other two indices.

This combination was used to form a single diversity score that integrates species presence, distribution balance, and dominance. This objective function will be used for the machine learning model.

3.3.4 Machine Learning Model

1 - Train-Test Split and Cross-Validation: The forked data was split into 80% training and 20% testing sets. This keeps some data separate for final evaluation while most is used for training the model.

Within the training set, 3-fold time series cross-validation was applied. The data was divided into three consecutive folds, training on earlier folds and validating on the next to respect time order. This method helps assess the model's ability to generalize to unseen data by providing multiple validation results while avoiding leakage from future data.

2 - Model Hyperparameters: The following parameters were used for the XGBoost model[6]. They were selected to balance complexity, generalization, and training efficiency given the structure of the simulation data.

- **Booster**: Specifies the type of model used. Since the data is a multivariate time series involving multiple variables across three trophic levels, the `'dart'` booster was chosen. It builds decision trees and applies dropout during training, which helps reduce overfitting and improve performance on noisy, non-linear data.
- **Learning rate**: Controls the step size during optimization. Lower values slow down learning, making it more stable and reducing the risk of overfitting.
- **Max depth**: Sets the maximum depth of the individual trees. Greater depth increases model capacity but may lead to overfitting.
- **Number of estimators**: Determines how many trees are built. More trees generally improve performance up to a point but increase training cost.
- **Tree method**: Defines the algorithm used to construct trees. The `'hist'` method was used for faster training, especially with larger datasets.
- **N_thread**: Sets the number of CPU threads used for parallel training. This was set to 4 to improve training speed.

3 - Hyperparameter Optimization (HPO): To improve predictive accuracy, a subset of model parameters was optimized using Optuna[7]. The objective was to minimize the root mean squared error (RMSE) on the training set.

Three key parameters were tuned: the learning rate (range: 0.05 to 0.2), maximum tree depth (4 to 8), and number of estimators (50 to 150). These were selected based on their strong influence on model behavior and complexity. Other parameters, including the booster type (`dart`) and tree construction method (`hist`), were fixed based on earlier experiments.

[6] https://xgboost.readthedocs.io/en/stable/python/

[7] https://optuna.readthedocs.io/en/stable/

Each trial generated a new parameter combination sampled from the defined ranges. XGBoost was then trained with that configuration, and the RMSE was recorded. After a predefined number of trials (e.g., 30), Optuna returned the best-performing parameter set, which was used in the final model deployment.

3.3.5 Optimization Using Differential Evolution (DE)

To select the best parameter updates, we used Differential Evolution (DE)[8], a population-based optimization algorithm. The goal is to find the input parameter set that maximizes the predicted biodiversity score.

Before running DE, a regression model is trained to approximate the relationship between input parameters and biodiversity outcomes. This model serves as a fast, surrogate evaluator that replaces direct simulation during optimization. DE proposes candidate parameter sets within fixed bounds $(0.0, 1.0)$ for all features and uses the trained model to predict the biodiversity outcome for each one.

The algorithm begins with a population of randomly sampled parameter sets. In each iteration, new candidates are generated by mutating and recombining existing ones. If a candidate yields a higher predicted biodiversity score, it replaces the previous solution. The process continues for up to 100 iterations with a population of 15 candidates, balancing exploration and exploitation throughout.

At the end, the parameter set with the highest predicted score is selected and applied in the simulation as the next intervention point. This creates a feedback loop where parameter updates are guided by the trained model rather than random selection.

Output: The DE process returns the best-performing parameter set found during optimization, which is then used to inform the next simulation step.

3.3.6 Mutation and Parameter Updates

After the Balancer Module is triggered and run, it outputs optimized parameters for all levels. These parameters are then used to update a selected part of the agent population for the next simulation run. Instead of replacing all existing parameters directly, we apply mutation. In biology, mutation refers to a random change in genetic material that introduces variation[Cleveland Clinic, 2025].

In our case, a selected proportion of agents per level will mutate by receiving the new parameters. This is similar to a level that originally consists only of wolves slowly evolving into a mix of wolves and foxes, rather than replacing all wolves with foxes at once.

Changing these values affects how agents behave in the simulation. Applying mutation gradually allows variation to spread without overwriting the population. This helps maintain diversity in each level and keeps interactions more dynamic and balanced. It also prevents early convergence to

[8] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html

a single behavior.

Mutation is done per level. First, we check how many agents are alive in that level. If there are fewer than 10, we select all of them for mutation. Otherwise, we randomly select 30% of the agents. These agents will receive new parameter values. After selecting the agents, we apply one of two mutation cases depending on how many classes already exist in the level:

- **If the number of classes is less than three:**
  We create a new class with the new parameters. The selected agents are moved into this class. Since it is a new class, we also update the interaction network. At least one existing class is assigned as a predator of the new class. At least one prey is also assigned to it if possible. The rest of the predator–prey connections are chosen randomly within allowed limits. This ensures the new class is properly connected in the food chain.

- **If there are already three classes**:
  We find the weakest class, meaning the one with the fewest agents. Its agents are reassigned to the strongest class, which has the most agents. This leaves the weakest class empty. We then overwrite its parameters with the new ones and move the selected agents into it. This allows us to introduce the new behavior without adding a new class.

This design helps control the number of classes in the simulation. Without limits, a new class would be added every time mutation happens. Over time, this would lead to many small and unstable classes that can go extinct quickly. Limiting the number of classes to three per level avoids this issue. It also reduces unnecessary complexity and improves performance. Keeping a small number of well-populated classes makes the system more stable and easier to manage.

The choice to mutate 30% of the agents has no strict rule behind it. It was chosen as a balanced value, not too low to have little effect, and not too high to replace most of the population at once. It provides steady variation without making the system unstable in one update.

4 Evaluation

4.1 Phase 1 Evaluation :

This section addresses SQ1 by examining whether the model can produce ecologically realistic dynamics under controlled conditions. The evaluation focuses on two main aspects:

(1) how energy moves through the system, and
(2) how agents behave, specifically in terms of movement, feeding, reproduction, and mortality.

We start with a baseline simulation that uses fixed parameters. This serves as a foundation to check whether the ecosystem maintains an energy balance, supports expected trophic interactions, and allows species to remain active over time.

To test the model's robustness, we then run three additional scenarios. Each one changes a single variable while keeping the rest of the parameters the same as the baseline. This helps isolate the impact of each change:

- **Grid size** is increased to see how a larger space affects the likelihood of agents encountering one another. A sparser environment could make it harder for agents to feed or reproduce.
- **Top-level agent count** is increased to simulate stronger predation pressure. This checks whether populations at lower trophic levels can still survive under more intense top-down control.
- **Movement cost** is decreased to explore whether weaker energy constraints lead to excessive movement or disrupt population balance.

After describing the results of each scenario, we compare them based on overall energy loss, patterns in agent behavior, and how agents are distributed in space. This helps us understand how different conditions affect the system's dynamics.

Finally, we explore the importance of energy conservation. In this specific diagnostic run, agents no longer gain energy through feeding, they only lose it to the environment. This lets us test whether internal energy flow is critical for maintaining realistic and stable population dynamics.

4.1.1 Baseline Setup

Before diving into the results, table 2 below shows the initial configuration of the main parameters used in the simulation. The setup defines the number of agents per trophic level, their initial energy ranges, energy costs for movement and reproduction, and the probabilities that guide movement, birth, and death.

Table 2: Initial parameters settings per trophic level

Level	Number of Agents	Initial Energy Range ($E_{i_{id}}$)	Move (E_{move})	Reproduction (E_{rep})	Movement Probability (m_{id})	Birth Probability (r_{id})	Death Probability (d_{id})
1	12	(0.2, 0.8)	0.05	0.15	0.4	0.5	0.07
2	6	(2, 3.5)	0.1	0.25	0.5	0.35	0.15
3	4	(4.5, 6)	0.25	0.4	0.6	0.25	0.25

The initial energy for each agent is randomly sampled from the given range per level. The grid size was set to 10×10, and the simulation was run for 30 timesteps. The parameter values were selected to fit a simple trophic structure: lower levels have more agents with lower energy, while higher levels have fewer agents with higher energy. The probabilities for movement, birth, and death were adjusted per level to create differences in behavior and lifespan across the trophic hierarchy.

4.1.2 Baseline Results

(a) Energy dynamics This part evaluates how energy flows through the system during the baseline simulation. The first bar plot (Figure 4a) compares the cumulative total energy gained with the total energy lost through agent interactions. These two values are exactly equal. This confirms that energy is consistently transferred between agents without loss at the interaction level. It shows that the simulation captures a basic form of energy conservation, similar to how energy circulates in real ecosystems between producers, consumers, and decomposers.

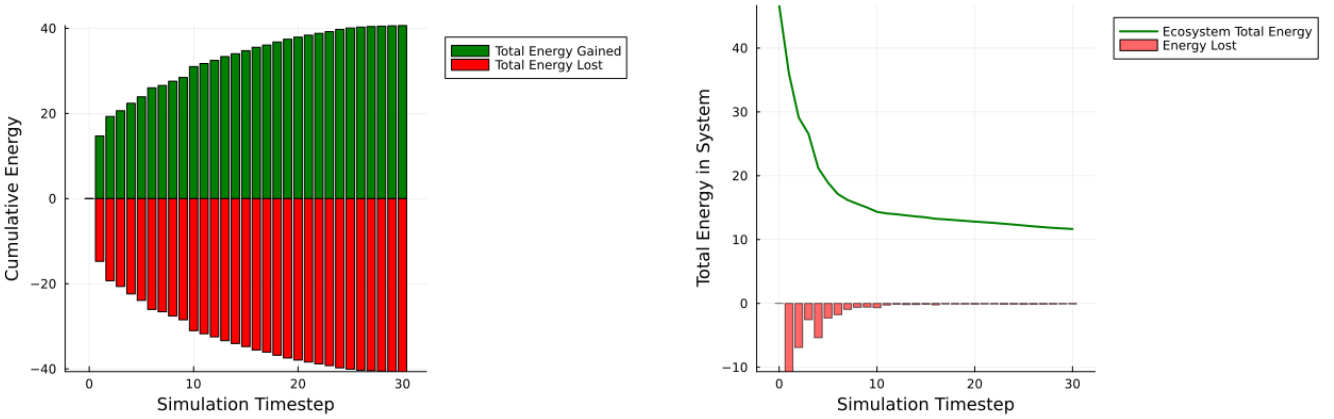
While the system balances energy during interactions, it does not fully recycle all energy. Some energy is lost to the environment when redistribution is not possible. For example, when agents move or die without neighbors to receive their remaining energy, that energy exits the system. The specific redistribution logic and rules are explained in Sections 3.2.2 and 3.2.5.

This loss is shown in the second plot (Figure 4b), which tracks how much energy is lost to the environment at each timestep. These losses accumulate gradually and reduce the total energy available in the system over time.

The energy at each timestep is updated based on this formula:

$$\text{Energy}[t] = \text{Energy}[t-1] - \text{E\_lost\_to\_env}[t]$$

This means that while energy is conserved in terms of tracked gains and losses between agents, the total system energy still declines. The sharp decline in the first 5 timesteps shows the need for early exploration. Most of the energy loss here comes from movement, which cannot be redistributed. This mirrors real ecosystems, where some energy is always lost and cannot be reused.



(a) Cumulative total energy gained and lost through agent interactions. Values are equal, confirming conservation at the interaction level.

(b) Energy lost to the environment per timestep. This shows how total system energy decreases when unused energy is not redistributed.

Figure 4: Baseline scenario showing energy flow and environmental loss.

Detailed energy tracking for all 30 timesteps, including total gain, total loss, system energy, and environmental loss, is provided in the 'Energy Balance Check Table' (Appendix B). The energy distribution method used in updates is shown in Appendix A.

(b) Agent Behavior In this part, we evaluate how agents behave over time and how their actions affect each other across different levels. We first look at overall system trends, then focus on each level individually. This helps capture both global patterns and level dependencies.

Figure 5 shows the total number of actions taken by all agents during the simulation. It includes movement, feeding, reproduction, and death events. This figure helps to understand how the overall system evolves. Figure 6 includes three subplots: one for each agent level. Each subplot displays the average number of actions per agent, including movement, feeding, reproduction, and death. These plots allow us to examine how behavior changes within each level and how the levels affect each other.

In Figure 5, movement stays low until timestep 10, then rapidly increases to over 1,000 events by timestep 20, indicating a surge in agent activity. Feeding and death events also rise during this period and follow nearly the same trend, suggesting most deaths are caused by predation. Reproduction increases slightly later, peaking just before timestep 20. After that, both feeding and reproduction decline, while deaths remain high. This implies that agents are dying from energy loss rather than being eaten. Movement continues to rise, meaning agents stay active even when they are no longer feeding or reproducing.

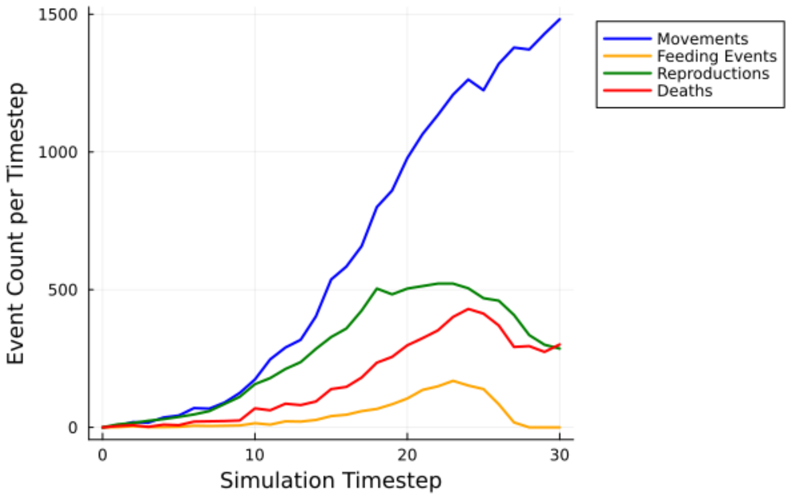
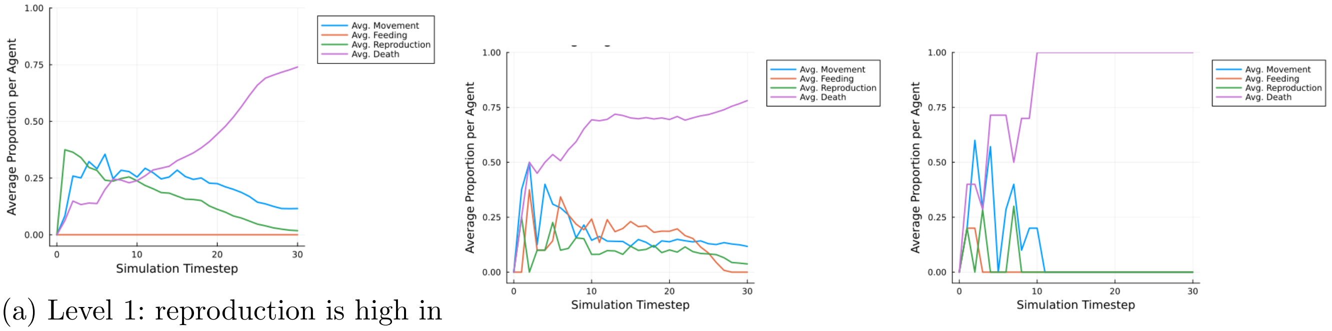


Figure 5: Baseline Case: Total number of actions taken by all agents over time, including movement, feeding, reproduction, and death. This shows how the overall system behaves throughout the simulation.

Looking more closely at individual levels in Figure 6:



(a) Level 1: reproduction is high in early steps, with no feeding. Deaths rise as energy runs out, leading to collapse.

(b) Level 2: feeding and reproduction rise while Level 1 is available, then drop as Level 1 declines.

(c) Level 3: short activity phase followed by full extinction due to lack of energy from lower levels.

Figure 6: Baseline Case: Average number of actions per agent for each level over time. These plots show how each level behaves and how they depend on one another.

- Figure 6a (Level 1): Reproduction peaks before timestep 10, then steadily declines and stops by timestep 30. Deaths rise above 0.7 per agent. Feeding stays at zero throughout, confirming that Level 1 agents don't feed but instead absorb energy from dead neighbors. Movement decreases after the initial steps, due to population and energy decline. Since Level 1 cannot feed, it depends on reproduction and absorbing energy from nearby deaths. Once reproduction drops, the level starts to collapse.
- Figure 6b (Level 2): Feeding increases in the first 10 steps, then declines after timestep 20, mirroring the decline in Level 1, its only prey. Reproduction follows the same pattern. Deaths remain high and even rise slightly toward the end, indicating energy depletion. Movement

also decreases over time. These trends show Level 2's dependence on Level 1 for both food and reproduction. As Level 1 declines, Level 2 collapses shortly after.

- Figure 6c (Level 3): Activity is very short-lived. Feeding and reproduction stop after timestep 10. The death rate then reaches 1.0, marking full extinction. This is due to the Level 2 population being too small, unstable, or distant to support Level 3. As a result, Level 3 doesn't survive long enough to grow throughout the simulation.

Together, these figures show how the survival of each level depends on the one below it. Level 1 reproduction supports the whole system. When it drops, feeding and reproduction at higher levels fall too. This leads to a chain reaction where all levels collapse. While movement supports interactions early on, it cannot prevent the system from failing once energy resources are depleted.

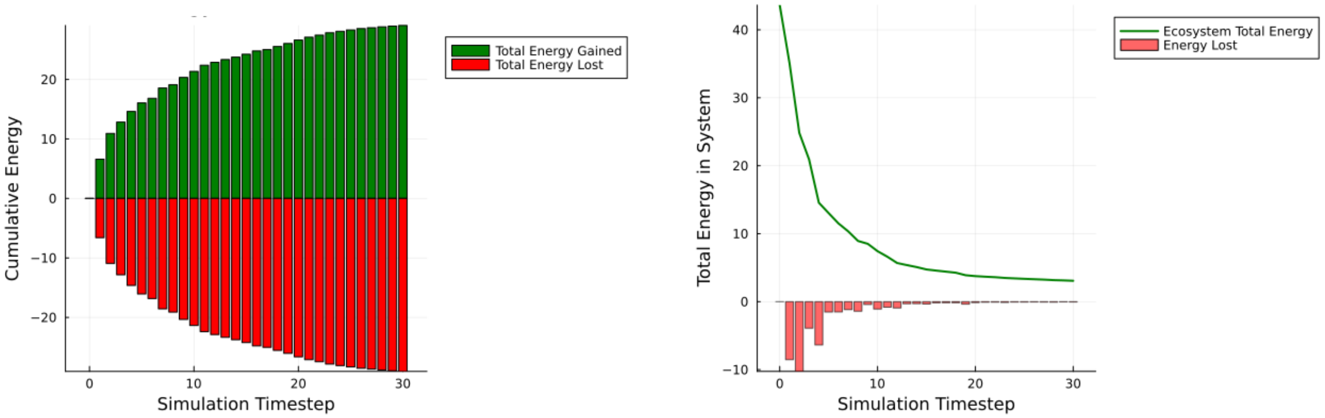
4.1.3 Robustness Scenarios

Scenario 1 — increased grid size (30×30)

In this scenario, we test how a larger grid affects energy flow and system dynamics. The grid size is increased from 10×10 (baseline) to 30×30, with all other parameters unchanged.

(a) Energy dynamics With a larger grid, the total energy gained and lost are both around 30 by the end of the simulation (Figure 7a). This is much lower than the baseline case (40), suggesting that feeding interactions happened less often. This is supported by the drop in total actions shown in Figure 8.

The second plot 7b shows energy lost to the environment per timestep. Between timestep 0 and 4, the loss is high, dropping sharply from about 45 to under 10 by timestep 10. This decline is steeper than in the baseline case. The reason is that agents needed more time to explore the larger space before finding each other. During this early phase, more energy was spent on movement and was lost to the environment. As a result, the total system energy drops faster and ends close to zero.



(a) Cumulative total energy gained and lost through agent interactions. Values are equal, confirming that energy was conserved during feeding and reproduction events.

(b) Energy lost to the environment per timestep. A sharp decline is seen in the first 10 steps, due to agents exploring the wider grid before interacting.

Figure 7: Scenario 1 — Overview of energy flow. The larger grid size reduces agent interactions, leading to fewer energy exchanges and more early energy lost to movement. This causes a quicker drop in total system energy compared to the baseline.

(b) Agent Behavior The goal is to observe how a larger space affects agent behavior, especially in terms of feeding, reproduction, and survival across levels.

Figure 8 shows the total number of actions over time. Compared to the baseline, feeding events stay low throughout the simulation, confirming fewer encounters. Predation is rare. In contrast, movement rises steeply, showing agents must explore more to find food. Reproduction and deaths still occur but at lower levels than in the baseline. The widening gap between feeding and deaths indicates that more agents die from energy loss than from being eaten. The larger grid slows down interactions and makes survival more dependent on movement and energy conservation.

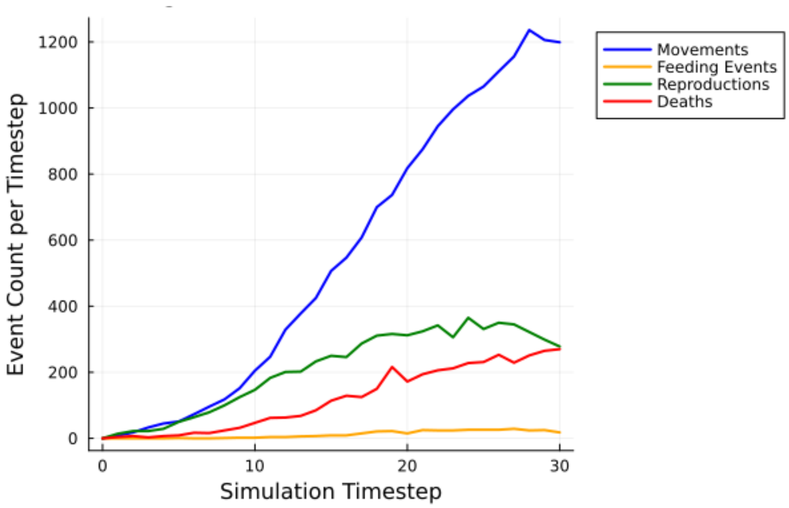
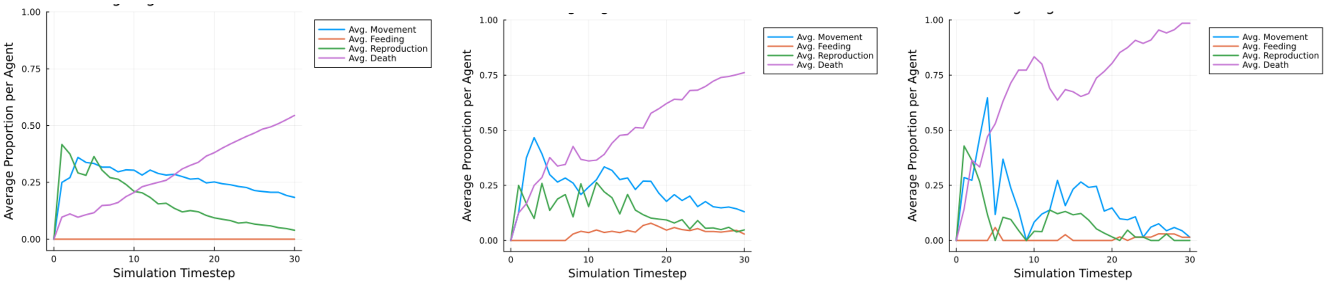


Figure 8: Scenario 1 — Increased grid size (30×30): Total number of actions taken by all agents over time, including movement, feeding, reproduction, and death. Compared to the baseline, feeding is nearly absent, while movement increases steadily due to the need for more exploration in the larger space.

Looking at individual levels in Figure 9:



(a) Level 1: reproduction follows the baseline pattern, but death rate is lower due to reduced predation. Feeding remains absent.

(b) Level 2: feeding remains near zero throughout, with low reproduction and steadily increasing deaths due to energy loss.

(c) Level 3: almost no feeding, but agents reproduce moderately and move conservatively to delay collapse despite rising deaths.

Figure 9: Scenario 1 — Increased grid size (30×30): Average number of actions per agent for each level over time. Larger grid size reduces encounter rates, leading to minimal feeding and stronger reliance on movement and reproduction to maintain survival.

- Figure 9a (Level 1): Reproduction peaks early and slowly drops off, similar to the baseline. Deaths remain lower (around 0.5 on average), which points to reduced predation. Feeding is always zero, as expected. Movement stays active but decreases gradually. Because predators struggle to locate prey, more Level 1 agents survive longer. However, reproduction still drops, which limits the overall growth of the level.
- Figure 9b (Level 2): Feeding remains close to zero for the entire run, showing that Level 2 agents rarely encounter Level 1 prey. Reproduction is lower than in the baseline and declines

steadily. Deaths rise gradually as agents lose energy over time. Movement starts high but decreases. The lack of feeding means Level 2 agents can't replace lost energy, leading to collapse even if they remain mobile.

- Figure 9c (Level 3): Feeding is nearly absent. Still, reproduction continues for most of the simulation. Movement is lower than in Level 2 but remains present. Deaths increase gradually and reach 1.0 by the end. This suggests that Level 3 agents rely on early reproduction while minimizing movement to slow down energy loss. They persist for a while but collapse once stored energy runs out.

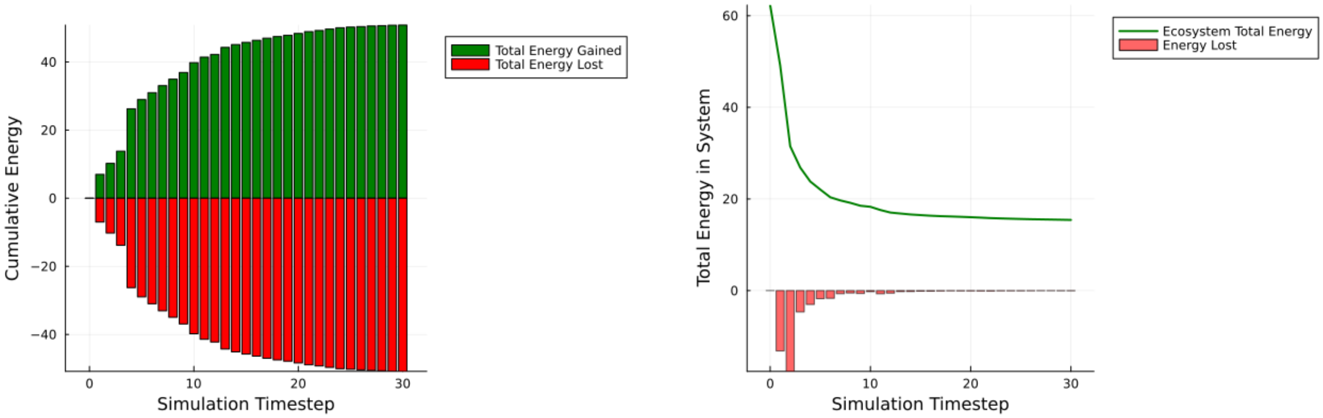
Overall, increasing the grid size makes encounters between agents less frequent, especially for predators. Feeding becomes rare, so survival depends more on early reproduction and how agents manage their energy. Level 1 benefits slightly from reduced predation, but Levels 2 and 3 decline due to the lack of food. Movement increases, but it cannot compensate for the lack of energy gain. The larger space introduces a trade-off: agents are harder to find, but that also limits the system's ability to sustain interactions.

Scenario 2 - doubling Level 3 population size

In this scenario, we extend the baseline by doubling the number of initial Level 3 agents from 4 to 8. All other parameters remain unchanged.

(a) Energy dynamics When increasing Level 3 agents, figure 10a shows a sharp jump between timestep 2 and 3 in both energy gained and lost. The values nearly double, matching the behavior patterns seen in figures 9a and 12b, where Level 1 and 2 agents show peaks in feeding and reproduction. Movement also increased sharply at this point, explaining the visible spike in environmental loss around timestep 2–3.

In figure 10b, the total energy started higher than the baseline due to the extra 4 Level 3 agents (60 instead of 40). However, the overall decline in system energy was slower than both the baseline and the larger-grid scenario, suggesting more efficient early interactions despite higher energy in circulation.



(a) Cumulative total energy gained and lost through agent interactions. The sharp rise between timestep 2 and 3 indicates intensive feeding and reproduction.

(b) Energy lost to the environment per timestep. A spike at timestep 2–3 shows increased movement and interaction.

Figure 10: Scenario 2 — Overview of energy flow. Doubling the Level 3 population increases early feeding and reproduction, resulting in a sharp rise in energy exchange around timestep 2–3. This also leads to higher movement and environmental loss early on. Despite starting with more energy, the system shows a slower overall decline compared to the baseline and Scenario 1.

(b) Agent Behavior This part examines how doubling the top predator population affects survival and interactions.

Figure 11 shows total agent actions over time. Movement steadily increases with sharper spikes before timestep 20. Deaths peak sharply around the same time, exceeding 300 events. After that, movement and reproduction decline. Feeding remains more active and lasts longer than in the larger-grid scenario. This shows that more predators increase interaction rates, especially predation, before the system starts collapsing.

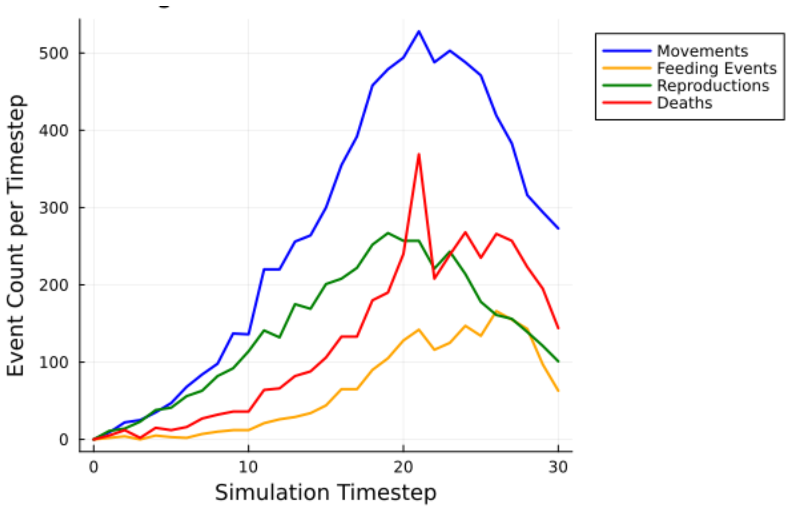
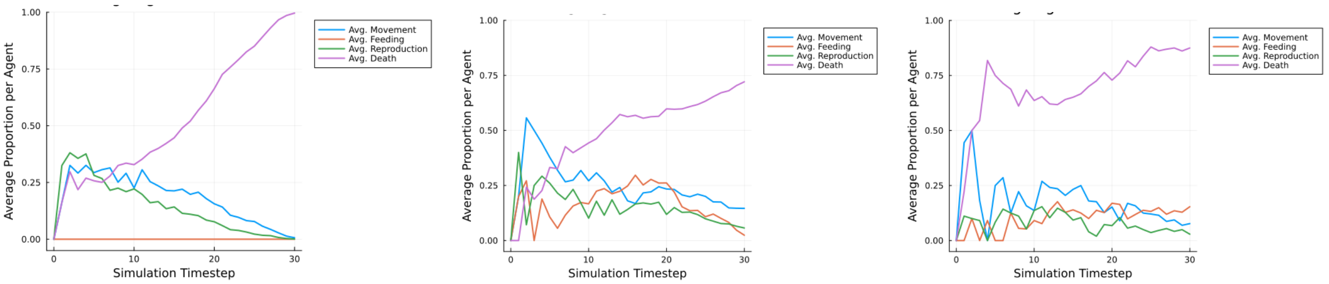


Figure 11: Scenario 2 — Doubling initial Level 3 agents (from 4 to 8): Total number of actions taken by all agents over time, including movement, feeding, reproduction, and death. Feeding is more active than in previous cases, and deaths peak around timestep 20, followed by a decline in movement and reproduction.

Figure 12 breaks down agent behavior by level:



(a) Level 1: early reproduction followed by sharp decline. Deaths reach 1.0 at the end, showing complete collapse.

(b) Level 2: feeding and reproduction stay active while Level 1 is alive, but both drop once Level 1 disappears.

(c) Level 3: stays active through the full simulation with regular feeding and reproduction, unlike the baseline case.

Figure 12: Scenario 2 — Doubling initial Level 3 agents: Average number of actions per agent for each level over time. With more Level 3 agents, feeding pressure increases, leading to faster collapse of Level 1 and 2.

- Figure 12a (Level 1): Reproduction peaks early, like the baseline, but deaths rise faster, reaching full collapse by the end. Movement and reproduction drop to zero soon after. The added pressure from Level 2 and the larger Level 3 population speeds up Level 1's decline.
- Figure 12b (Level 2): Feeding stays active longer and slightly higher than baseline. Reproduction follows a similar early peak and decline. Deaths steadily increase, passing 0.7 by the end, while movement gradually falls. This shows Level 2 feeds while Level 1 exists, but collapses after its prey disappears.

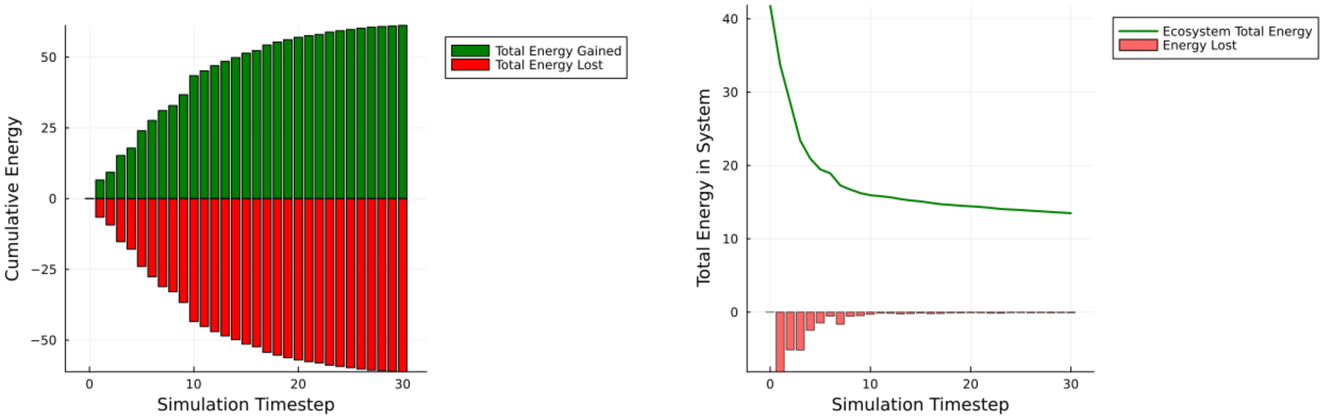
- Figure 12c (Level 3): The main difference from baseline is clear. Level 3 remains active throughout the simulation, with sustained feeding and ongoing reproduction. Movement decreases but persists. Death rate rises but never reaches extinction. The larger initial population lets Level 3 feed more and reproduce longer, delaying collapse.

Overall, increasing Level 3 agents strengthens top-down pressure. The top predators survive longer with higher feeding and reproduction. This accelerates collapse of Level 1 and eventual decline of Level 2. The system resembles baseline more than the large grid case but still collapses. The stronger predator presence helps Level 3 persist but affects overall stability.

Scenario 3 - reducing movement cost

In this scenario, we reduce the energy cost of movement by half for all agent levels. This change encourages more movement with less energy lost, allowing us to examine how increased movement affects total energy and agent behaviors.

(a) Energy dynamics When the movement cost is halved, energy lost to the environment is lower than in all previous scenarios, reflecting the reduced cost of movement(Figure 13b). This encourages more frequent movement, which in turn results in the highest total energy gained and lost across all cases, reaching around 60 by the end of the simulation.



(a) Cumulative total energy gained and lost through agent interactions. Values remain balanced, showing conservation throughout feeding and reproduction events.

(b) Energy lost to the environment per timestep. Lower movement cost reduces environmental energy loss, resulting in a gentler decline in total system energy.

Figure 13: Scenario 3 — Overview of energy flow. Halving movement cost lowers energy lost to the environment and encourages sustained agent activity, leading to higher cumulative energy exchange and a slower decrease in total system energy.

Unlike other scenarios, where energy gain and loss sharply increased only at the start and then stabilized, here the cumulative energy gain and loss rise steadily from timestep 0 to 10. This pattern indicates consistent agent activity throughout the simulation rather than a front-loaded burst. The steady increase in energy exchange suggests broader exploration and sustained feeding and reproduction, as shown in Figure 13a.

This is supported by agent behavior plots (Figure 15), which show regular spikes in feeding and reproduction across all three levels throughout the simulation, indicating ongoing interactions.

Additionally, this scenario shows the least sharp decline in total system energy. At timestep 10, the total energy remains higher than in all previous cases, confirming the effect of reduced movement cost on preserving energy within the system.

(b) Agent Behavior Figure 14 shows the overall number of actions taken. While the general shape resembles the baseline, feeding and reproduction remain slightly more elevated in the later stages. Movement rises steeply until timestep 20, then drops slightly, but not as sharply as in previous cases. Deaths increase steadily and eventually surpass reproduction, yet feeding stays more active toward the end. These patterns suggest that agents maintain interactions for longer, likely due to the reduced energy penalty of moving.

Figure 14 shows the total number of actions taken by all agents over time. The overall pattern looks similar to the baseline, but feeding and reproduction stay a bit higher in the later stages. Movement rises sharply until about timestep 20, then drops slightly, showing a more gradual decline than in earlier cases. Deaths increase steadily and eventually overtake reproduction, yet feeding remains more active toward the end. This suggests agents keep interacting longer because moving costs less energy.

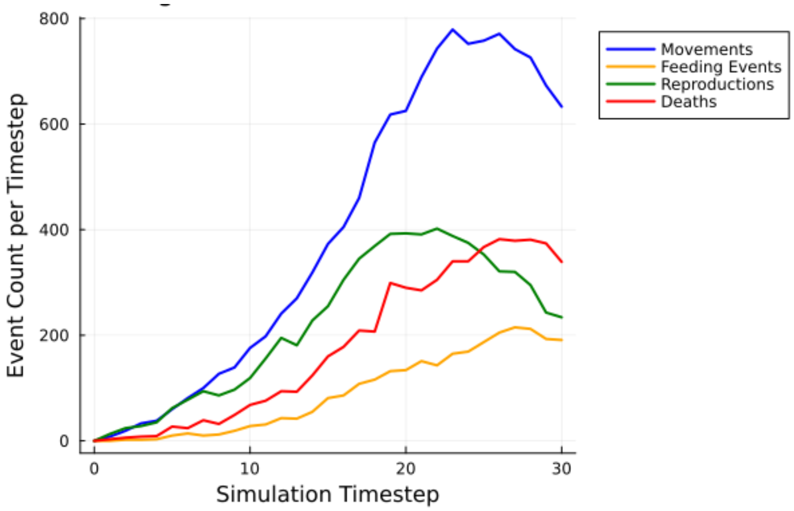
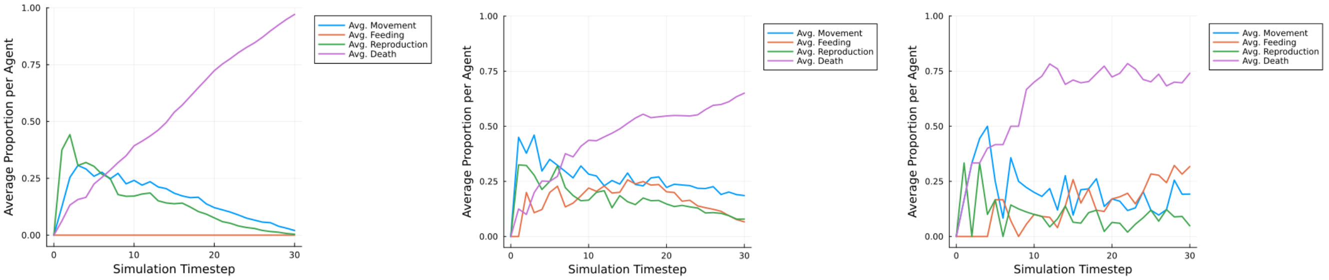


Figure 14: Scenario 3 — Halved movement cost for all levels: Total number of actions taken by all agents over time. Feeding and reproduction stay more active in later steps, and movement remains high for longer compared to the baseline.

Looking closer at each level:



(a) Level 1: similar to baseline. Reproduction and movement drop over time, and death reaches 1.0 by the end.

(b) Level 2: feeding and reproduction remain more stable than baseline, supported by higher movement throughout.

(c) Level 3: survives the full simulation. Feeding and reproduction continue due to more efficient movement.

Figure 15: Scenario 3 — Halved movement cost: Average number of actions per agent for each level. Lower cost helps Levels 2 and 3 stay active longer by keeping feeding and reproduction going.

- In Figure 15a (Level 1): Reproduction and movement follow baseline trends. Feeding stays zero, as expected. Deaths reach complete extinction by the end. There's no major change here, lower movement cost doesn't help Level 1 since they don't feed or gain energy from moving.
- In Figure 15b (Level 2): Movement remains higher for longer, and feeding stays active almost the entire run. Unlike baseline, feeding doesn't drop off near the end. Reproduction also keeps going, showing Level 2 agents maintain energy and reproduction later in the simulation. Lower movement cost helps them stay connected to prey and survive longer.
- Figure 15c (Level 3): This level shows the biggest difference. Despite starting with only 4 agents like baseline, Level 3 doesn't go extinct here. Feeding happens consistently, especially in the latter half, and reproduction continues steadily. Movement fluctuates around 20–25%, enough to find prey regularly. Lowering movement cost lets Level 3 survive and sustain itself without needing a bigger starting population.

Overall, reducing movement cost extends the system activity. It helps Levels 2 and 3 keep feeding and reproducing longer, while Level 1 stays much the same. Therefore, increased mobility mainly benefits higher levels that rely on finding and exploiting prey.

4.1.4 Scenario Comparison and Evaluation

After evaluating energy flow conservation and agent behaviors for each scenario extended from the baseline in detail, we provide here a comparison across all cases together to highlight their differences and similarities.

(a) Energy dynamics: Across all scenarios, energy availability declines over time, but the rate and cause differ. In the larger grid case, energy loss is rapid early on due to increased movement and exploration, causing more energy to be lost to the environment. The predator-dense scenario shows a sharp increase in energy gain and loss between timesteps 2 and 3, linked to peaks in feeding and reproduction. When movement cost is reduced, cumulative energy gain and loss increase steadily from the start, reflecting consistent agent activity over time. Across all cases, total energy gained equals total energy lost, confirming energy conservation within the system. These results show how grid size, initial population, and movement cost affect energy flow.

(b) Agent behavior: Across the four cases, system dynamics shift based on grid size, predator density, and movement cost. In the baseline, Level 3 collapses early, and the system declines shortly after. In the larger grid, feeding drops due to fewer encounters. Level 1 lasts longer, but Level 3 cannot survive. With more Level 3 agents, top-down pressure increases. Level 3 persists, but Levels 1 and 2 collapse faster. When movement is cheaper, feeding and reproduction stay stable longer. Level 3 persists without needing a large initial size. Level 1 declines early in all cases.

Table 3 below summarizes these insights by linking energy flow, agent behavior, and ecological outcomes. The biodiversity index is an overall measure based on a weighted combination of three common biodiversity metrics, as explained in Section 3.3.3.

Table 3: Comparison of scenarios showing effects on energy flow, agent behavior, survival rates, and biodiversity. Energy gain equals loss in all cases, confirming conservation.

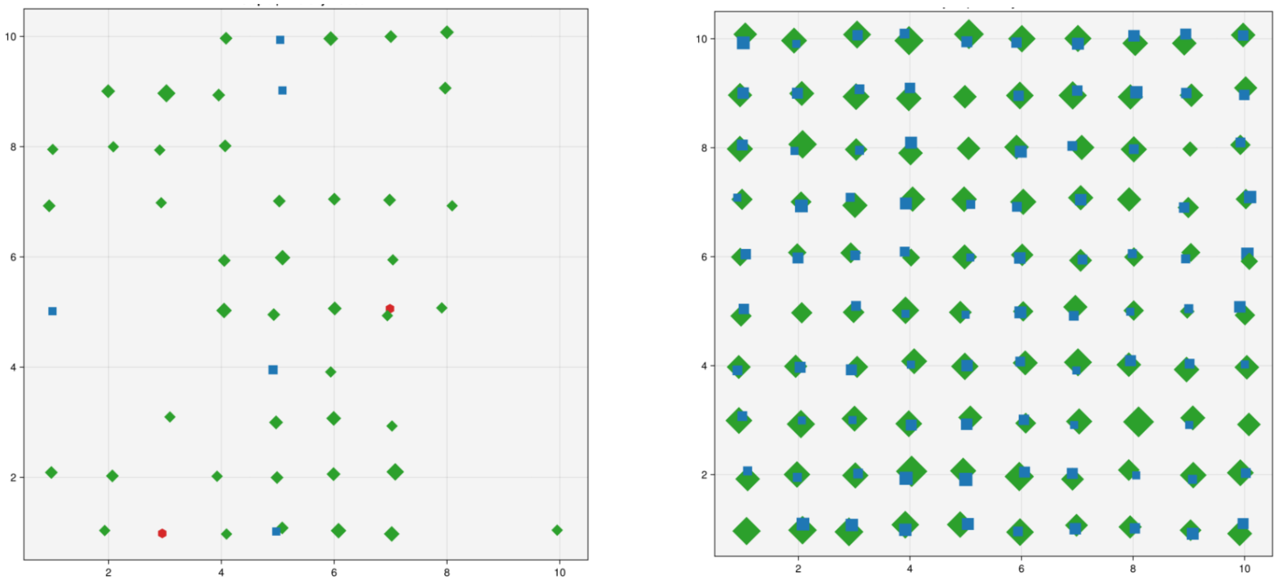
Scenario	**Parameter Change**	**Energy Gain (=Loss)**	**Behavior & Ecological Impact**	**Final Survival (L1/L2/L3)**	**Biodiversity Index**
Baseline	Default configuration	40.65	System collapses after top-level extinction and energy loss	0.26 / 0.22 / 0	0.47
Larger Grid	Increased to 30x30	29.08	Lower encounter rates reduce feeding; system collapses more slowly	0.46 / 0.24 / 0.015	0.38
High Predators	Doubled Level 3 population 4 to 8	50.85	Increased top-down pressure; early collapse of prey. Level 3 persists longer; system destabilized faster.	0.004 / 0.28 / 0.125	0.32
Reducing Movement Cost	E_move by half	61.22	Higher stability; top levels survive and stay active longer	0.03 / 0.35 / 0.26	0.48

4.1.5 Spatial Structure and Distribution

Besides energy flow and agent behavior, this section examines how agents are distributed across the grid in each scenario. Figures 16 to 19 show snapshots at timesteps 5 and 25, representing the early and later stages of the simulation. These images show how population density, species dominance, and clustering change over time.

Agent levels are shown by color: green for Level 1, blue for Level 2, and red for Level 3. The marker size corresponds to the number of agents occupying the same grid cell, making it easier to see clustering and density differences.

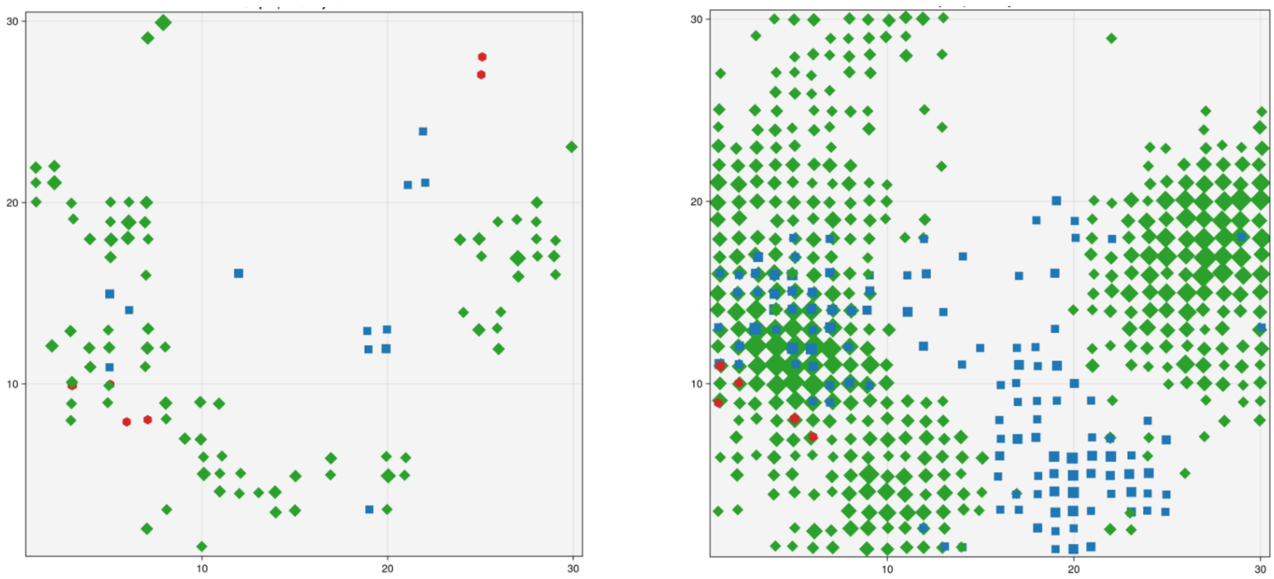
By comparing these spatial patterns, we can directly link differences in agent distribution to the energy flow and behavior dynamics discussed earlier (Table 3).



(a) Baseline – Timestep 5: Agents spread with some clusters. All levels present, higher levels beginning to decline.

(b) Baseline – Timestep 25: Level 3 agents extinct. Level 1 dominates with clusters; Level 2 persists but less dense.

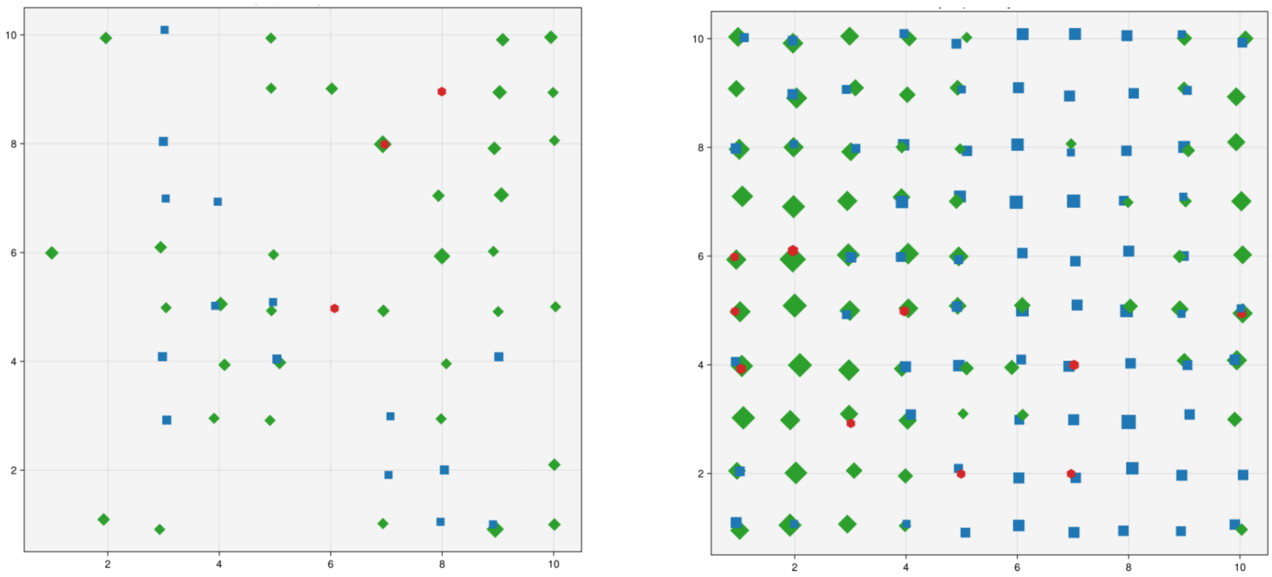
Figure 16: Agent spatial distribution in the baseline scenario at timesteps 5 and 25. Higher-level collapse occurs early; lower levels cluster unevenly.



(a) Scenario 1 – Timestep 5: Agents spread across larger grid, clustering mainly near edges. All levels present but sparse.

(b) Scenario 1 – Timestep 25: Clusters remain in same areas; Level 2 expands into empty spaces. Levels coexist but remain separated.

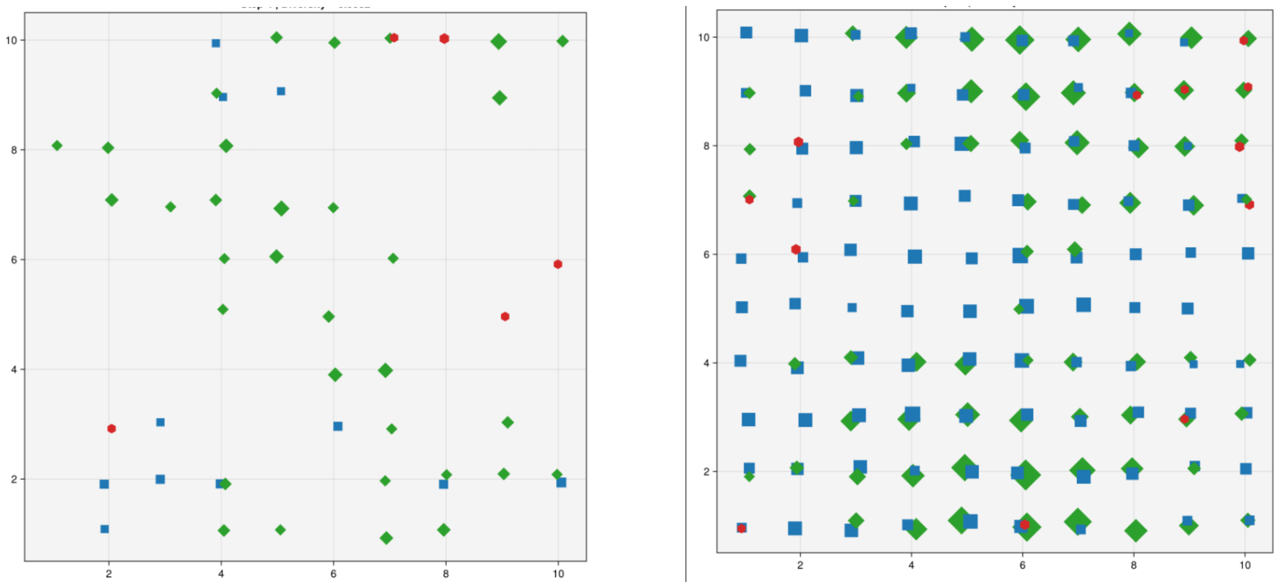
Figure 17: Spatial distribution in Scenario 1 (increased grid size) at timesteps 5 and 25. Larger grid leads to more dispersed agents with stable, separated clusters.



(a) Scenario 2 – Timestep 5: Higher density of Level 3 agents than baseline; Level 1 and 2 agents begin to spread out.

(b) Scenario 2 – Timestep 25: Increased Level 3 presence reduces Level 1 and 2 agents in some areas due to predation.

Figure 18: Spatial distribution in Scenario 2 (doubled Level 3 agents) at timesteps 5 and 25, showing stronger predator pressure and its effect on lower levels.



(a) Scenario 3 – Timestep 5: Agents are more spread out with fewer tight clusters. All levels present with some coexistence.

(b) Scenario 3 – Timestep 25: Even distribution maintained with balanced coexistence across levels, likely due to reduced movement costs.

Figure 19: Spatial distribution in Scenario 3 (halved movement cost) at timesteps 5 and 25. Lower movement cost promotes more even spread and stable coexistence among agent levels.

At timestep 5, all scenarios show agents forming small clusters but with different spacing. In the baseline, agents group around certain areas, leaving parts of the grid empty. Scenario 1, with a larger grid, shows similar clustering but agents are more spread out, especially in the center. In Scenario 2, doubling Level 3 agents creates more red clusters early on, pushing Levels 1 and 2 slightly to the sides. Scenario 3, with reduced movement cost, allows agents to spread more evenly, with fewer tight clusters.

By timestep 25, the baseline is dominated by Level 1 agents, Level 3 agents have died out, and clusters remain in their original areas. Scenario 1 remains sparse overall, but more agents appear in previously empty zones, mostly Level 2, though they mostly stay in their initial regions. Scenario 2 shows Level 3 agents surviving and dominating specific areas, while Levels 1 and 2 are clearly reduced. Scenario 3 ends with a more balanced presence across all levels and a smoother spread, with fewer dense clusters.

These plots show how spatial distribution, survival, and movement differ between scenarios and connect to the energy flow and agent behavior results presented earlier.

4.1.6 Effect of energy conservation

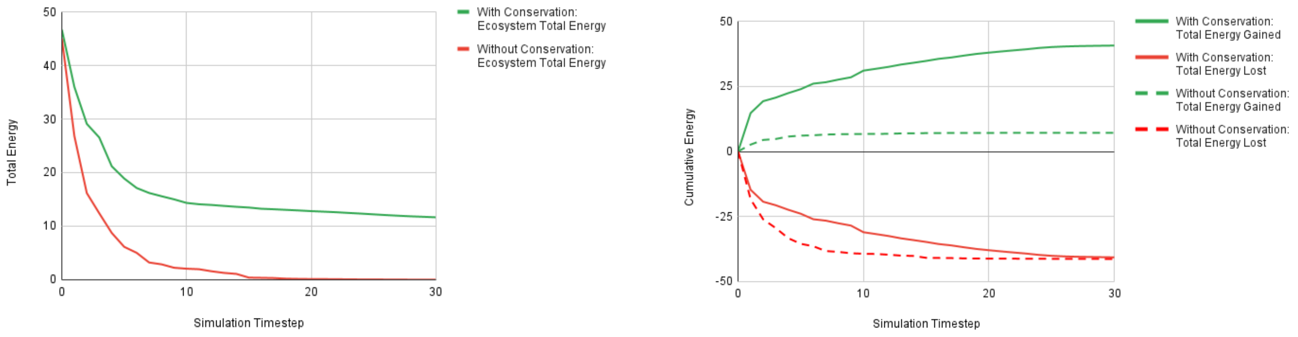
This section evaluates how energy conservation affects agent behavior and energy flow, based on the update rules defined in Section 3.2.5. We compare two simulation versions: conserving vs. non-conserving, using identical initial conditions and baseline parameters.

In the conserving system, all energy lost by agents is either transferred (e.g., through feeding or reproduction) or accounted as environmental loss. The system remains closed and balanced: total energy gained matches energy lost, aside from gradual losses to the environment. This makes it possible to track where energy moves throughout the ecosystem (see Section 3.2.5).

In the non-conserving version, energy is not preserved. Only a portion of energy is passed on during feeding or reproduction; the rest is discarded. For example, predators receive just 10% of prey energy, consistent with Lindeman's 10% Law [Lindeman, 1942], which states that only about 10% of the energy from one trophic level is transferred to the next. Offspring inherit only part of the parents' energy, and any energy held at death is lost.

Figure 20 summarizes the results.
In Figure 20a, both simulations begin with the same total energy. The conserving system shows a gradual decline due to environmental loss. In contrast, the non-conserving system experiences a rapid energy drop as non-transferred energy is removed from the system entirely.

Figure 20b tracks cumulative energy gained and lost. In the conserving version, gains and losses remain balanced throughout, confirming that energy is being correctly redistributed and tracked. In the non-conserving version, energy loss far exceeds gain. This gap grows over time, indicating that energy continuously leaves the system without return.



(a) Total system energy over time. With energy conservation, energy declines slowly due to losses to the environment. Without conservation, energy drops quickly because it is not redistributed, leading to instability.

(b) Cumulative energy gained and lost. The conserving system maintains balance between gain and loss. In contrast, the non-conserving system shows a huge gap more energy loss than gained, demonstrating how energy inefficiency weakens ecosystem sustainability.

Figure 20: Comparison of energy dynamics in conserving and non-conserving systems. Energy conservation helps maintain balance and supports ecosystem function, while energy loss without redistribution leads to faster collapse.

To quantify this divergence, Table 4 compares outcome metrics between the two systems.

Table 4: Comparison of selected metrics under energy-conservation and non-conservation conditions. The conservation simulation supports higher energy retention, agent activity, and population size.

Metric	**With-Conservation**	**Without-Conservation**	**Relative Difference (× times)**
Total Energy at Final Step	11.65	0.015	×782
Total Energy Gained	40.65	7.14	×5.7
Total Energy Lost	40.65	41.27	×0.99
Agents Alive – Level 1	2590	47	×55.1
Agents Alive – Level 2	616	23	×26.8
Agents Alive – Level 3	0	0	—
Avg. Feeding Actions per Timestep	44.42	8.41	×5.3
Avg. Reproduction Actions per Timestep	271.38	22.25	×12.2
Avg. Movement Actions per Timestep	619.30	57.93	×10.7

The conserving system retains significantly more energy by the end, 782 times more than the non-conserving case. Although both lose similar amounts overall, only the conserving version redistributes this energy across agents, keeping it in circulation. In contrast, the non-conserving version loses this energy permanently, leading to rapid depletion and reduced agent activity.

Population outcomes reflect this difference. By the final timestep, the conserving version retains 55 times more Level 1 agents than the non-conserving version. Level 2 shows a similar trend, with 27 times more survivors. Level 3 agents go extinct in both simulations due to limited resources at the top trophic level.

Agent behavior is also more sustained when energy is conserved. Feeding, reproduction, and movement occur more frequently, with reproduction showing the largest increase over 12 times more events on average. These activities depend on energy being available and passed on, which only happens in the conserving version.

Overall, the results show that energy conservation supports long-term ecosystem function. Without it, energy exits the system early, reducing interaction, weakening trophic links, and accelerating collapse. Even with identical starting conditions, the two versions diverge sharply.

Additional plots in Appendix C compare behaviors over time across both systems. These show that activity in the non-conserving version declines rapidly, while the conserving system maintains and even increases behavioral activity throughout the simulation.

4.2 Phase 2 Evaluation:

In this section, we evaluate the outcomes of Phase 2, which introduced the Balancer Module, to answer SQ2: *How does adaptive optimization of species parameters affect biodiversity stability compared to no optimization?* We compare simulation runs with and without feedback-based parameter updates under different settings.

The evaluation is split into three parts:

- **Case 1: Baseline Ecosystem - No ML vs. ML**
- **Case 2: ML feedback without energy conservation**
- **Case 3: ML recovery in an unbalanced ecosystem**

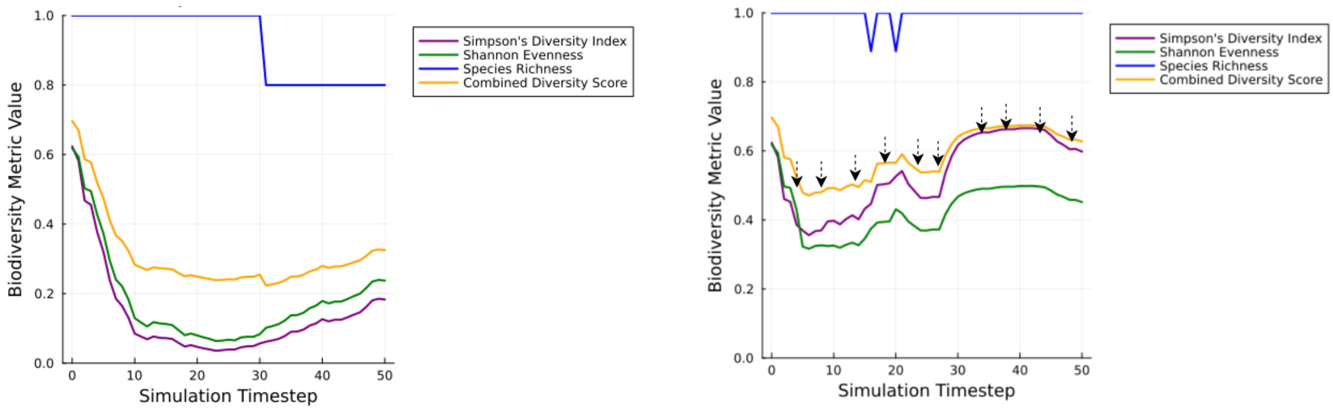
4.2.1 Case 1: Baseline Ecosystem - No ML vs. ML

This scenario compares two simulations starting from the same baseline: one without machine learning (no-ML) and one with ML-guided parameter updates, to analyze how the balancer module's feedback affects the combined diversity index over time. The baseline setup used here is detailed in Table 2, with a grid size of 30x30 and a simulation run for 50 timesteps.

The combined diversity score refers to the objective function defined earlier, which is a weighted combination of richness, Shannon evenness, and Simpson's diversity. From here on, we refer to this combined metric as the *'Combined Diversity Index'*.

In the **no-ML simulation**, the combined diversity index steadily declines and stabilizes between **0.28 and 0.35**, as shown in Figure 21a. With no parameter updates, the ecosystem remains in a degraded, low-diversity state and shows no signs of recovery.

In contrast, the **ML-guided simulation** (Figure 21b) starts similarly but begins recovering after the first intervention at **timestep 28**. The combined diversity index rises from **0.5401 at timestep 27** to **0.6411 at timestep 30**. The model predicted a value of **0.6314**, closely matching the actual outcome. This indicates that the ML model correctly identified and applied a beneficial parameter update.



(a) Biodiversity indices in the baseline simulation without ML intervention. Combined diversity index declines steadily and stabilizes at a low level, indicating no recovery.

(b) Biodiversity indices with ML-guided updates every four timesteps. Early interventions stabilize the system; recovery begins after timestep 28 and continues gradually.

Figure 21: Comparison of biodiversity trends in simulations with and without machine learning. Both use the same baseline setup; only the ML-driven case applies adaptive parameter updates to guide the system toward recovery and balance.

After timestep 30, the combined diversity index continues to improve gradually, suggesting that the system is stabilizing and moving toward a more balanced state. This positive trend contrasts with the static outcome in the no-ML case.

To see how recovery was driven, Tables 5a and 5b show the parameter values before and after the ML intervention. The changes reveal the focused adjustments made to guide the system's improvement:

Level	# Agents	E_move_id	E_rep_id	m_id	r_id	d_id
1	825	0.1624	0.4810	0.4175	0.3638	0.2615
2	208	0.0421	0.0473	0.7191	0.2515	0.0468
3	149	0.1994	0.2312	0.7293	0.6910	0.1489

(a) Before ML intervention (timestep 27). Ecosystem is declining.

Level	# Agents	E_move_id	E_rep_id	m_id	r_id	d_id
1	379	0.0550	0.6887	0.5796	0.4313	0.1615
2	183	0.3444	0.1417	0.9680	0.8912	0.0456
3	160	0.0487	0.0203	0.0727	0.6150	0.0345

(b) After ML intervention (timestep 30). Diversity and balance improved.

Table 5: Comparison of parameter values before and after ML intervention.

- **Level 1:** Movement cost was lowered, reducing energy spent on movement and improving resource access. Reproduction and survival probabilities increased. Despite this, the population decreased sharply from 825 to 379, indicating a shift from overcrowding toward a more stable, efficient population with less competition.

- **Level 2:** Movement cost increased, likely to conserve energy by limiting unnecessary movement. At the same time, reproduction probability increased to compensate. The population slightly decreased from 208 to 183, reflecting a balance between reduced mobility and sustained reproduction.

- **Level 3:** Both movement and reproduction costs were reduced, lowering energy barriers for survival and reproduction. This led to a population increase from 149 to 160, helping to restore balance by strengthening this underrepresented level.

The ML model generated parameter updates that reduced the decline in biodiversity and triggered recovery. Without these updates, the no-ML simulation showed no signs of improvement. This confirms that adaptive interventions, when matched to the ecosystem's state, are crucial for restoring biodiversity.

4.2.2 Case 2 : ML feedback without energy conservation

In this scenario, we tested how ML feedback affected the baseline ecosystem without energy conservation. Figure 22 shows how the biodiversity changes over time.
Unlike the baseline case (Figure 21b), where ML intervened every 4 timesteps, here ML triggered many times because the diversity score dropped below 0.4 often.

Between ML interventions, Shannon evenness and Simpson's diversity stayed relatively stable with minor improvements. Richness increased as mutations created new classes. However, these improvements happened only right after the ML intervened. The combined diversity index oscillated downward, starting near 0.8 and ending around 0.04, showing the system collapsed.

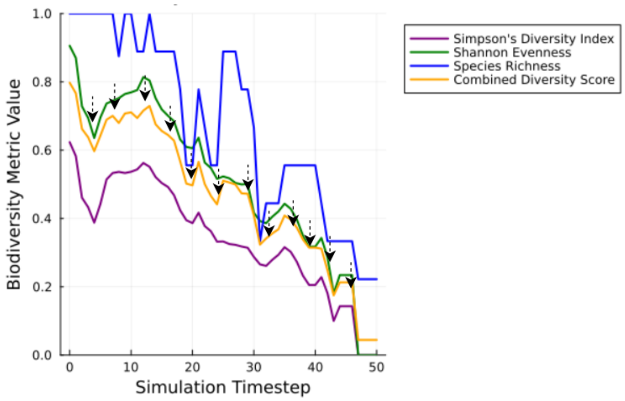


Figure 22: Biodiversity indices over time with ML feedback but no energy conservation. ML intervenes frequently while the biodiversity declines, unlike the baseline in Figure 21b

This means ML tried to improve biodiversity, but without energy conservation, the system was unstable. The improvements were only temporary, and biodiversity fell over time despite ML's efforts.

4.2.3 Case 3: ML recovery in an unbalanced ecosystem

This scenario tests the ML intervention starting from an unbalanced setup that causes a sharp decline in biodiversity indices. This lets us evaluate how effectively the ML balancer recovers the system.

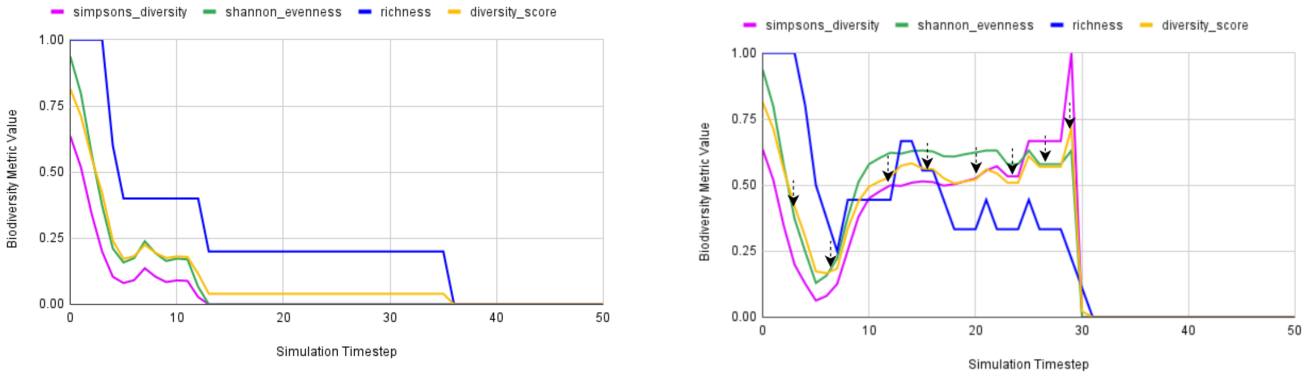
The initial parameters per level are:

Level	# Agents	E_{move}	E_{rep}	m_id	r_id	d_id
1	6	0.5	0.2	0.2	0.2	0.6
2	4	0.3	0.4	0.4	0.5	0.5
3	20	0.7	0.6	0.6	0.8	0.2

Table 6: Initial parameters for the unbalanced setup. Lower levels are disadvantaged in population, energy, and survival.

Lower levels start with fewer agents, higher death probabilities, higher movement energy costs, and lower reproduction probabilities, creating a stressed ecosystem to challenge the Balancer Module.

Figure 23 shows a comparison of the unbalanced setup with and without the intervention of ML. When no feedback mechanism is included in the simulation, the ecosystem experiences a rapid decline in biodiversity. As shown in figure 23a, the combined diversity index starts high at around 0.8, driven by high initial Shannon evenness (approximately 0.93) and Simpson diversity (0.65), based on an initial agent distribution of 12, 8, and 20 individuals across three species. By timestep 4, there is a sharp decline in all biodiversity indices. Two species go extinct, causing species richness to drop to 0.6, while Simpson diversity falls to approximately 0.1. The combined diversity score also drops to 0.24. This decline continues over the next few timesteps. At timestep 13, only one species remains, and all biodiversity indices approach zero, with the combined diversity index reaching 0.04. After timestep 36, complete extinction occurs and all indices remain at zero for the rest of the simulation. The large spatial configuration (30×30 grid) relative to the number of agents (40) may contribute to early extinctions, as agents are more dispersed and less likely to interact, feed, or reproduce efficiently. This result shows that, without feedback or adaptive intervention, the ecosystem cannot maintain biodiversity or recover from species loss when initialized under unbalanced conditions.



(a) Biodiversity indices in the unbalanced ecosystem without ML intervention. Combined diversity index rapidly declines, leading to complete collapse by timestep 36. No recovery is observed.

(b) Biodiversity indices in the same unbalanced setup with ML-driven feedback. Interventions stabilize the combined diversity index temporarily, delaying extinction compared to the no-ML case.

Figure 23: Comparison of biodiversity trends in simulations under worst-case conditions. Both simulations use the same initial parameters, but only the ML case applies adaptive updates to delay collapse and improve resilience.

With the inclusion of ML, the simulation included a feedback mechanism that adjusted agent parameters based on the current biodiversity status. ML interventions were scheduled every four timesteps but were triggered earlier if the combined diversity score fell below 0.4. Based on this dynamic threshold, interventions were executed at timesteps 4, 7, 12, 16, 20, 24, 28, and 31.

Figure 23b shows the biodiversity indices over time for this scenario. At timestep 4, the first ML intervention was applied, but all biodiversity indices further declined in the next timestep: the overall diversity score dropped from 0.3 to 0.17, and richness decreased from 0.8 to 0.5. This decline suggests that the first intervention was ineffective, likely because of insufficient data at the early stage and the near extinction of level 1 agents. The proposed parameters at this point are listed in Table 7. For example, the model increased movement and reproduction chances but also raised the energy cost of movement (E_move = 0.89), which likely encouraged more activity but used a lot of energy.

Step	Level	E_move_id	E_rep_id	m_id	r_id	d_id	Actual Diversity Score	Predicted Diversity Score	Absolute Error
4	1	0.8991	0.3762	0.5356	0.5535	0.5848	0.1738	0.5599	-0.3861
	2	0.51	0.7026	0.2116	0.1021	0.9283			
	3	0.1477	0.5411	0.4285	0.6117	0.9357			
7	1	0.2393	0.3826	0.2314	0.2625	0.9708	0.3359	0.3053	0.0306
	2	0.1304	0.474	0.7535	0.9027	0.0652			
	3	0.8264	0.0255	0.2531	0.6	0.3745			
12	1	0.5809	0.4905	0.6038	0.6603	0.5036	0.5316	0.5335	-0.0019
	2	0.6942	0.2832	0.8162	0.9552	0.8205			
	3	0.9297	0.0568	0.381	0.7898	0.2684			
16	1	0.0424	0.1291	0.8204	0.7362	0.3042	0.5607	0.5822	-0.0215
	2	0.9454	0.2708	0.9661	0.4851	0.7601			
	3	0.9762	0.5031	0.7534	0.3841	0.6724			

Table 7: Snapshot of ML-proposed parameter sets during feedback interventions at timesteps 4, 7, and 16 in the worst-case scenario. The table lists energy costs and behavioral probabilities suggested per level, alongside the model's predicted combined diversity score and the actual score observed after applying the parameters. This illustrates how early interventions suffer from limited data, while later ones (e.g., at timestep 16) begin to align better with observed outcomes.

The second intervention at timestep 7 resulted in a remarkable improvement: the combined diversity index increased to 0.33, and richness improved due to the emergence of a mutated level 3 agent. The ML's expected combined diversity index at this point closely matched the actual outcome, indicating that the model benefited from having access to more system variability. After this, biodiversity indices generally improved. By timestep 12, the score rose to $\tilde{0}.58$ and richness recovered to approximately 0.66, showing a partial recovery supported by better-informed parameter updates.

From timestep 15 to 25, the system entered a relatively stable phase. Interventions at timesteps 16 and 20 helped maintain diversity, but richness showed small oscillations. During this phase, the number of agents steadily decreased. As shown in Figure 24, by timestep 25 only four agents remained across all species levels, reducing the potential for meaningful interaction. Although the ML continued to update parameters, the population decline became irreversible. By timestep 31, the final agents had died, and all biodiversity indices dropped to zero, marking complete ecosystem collapse.

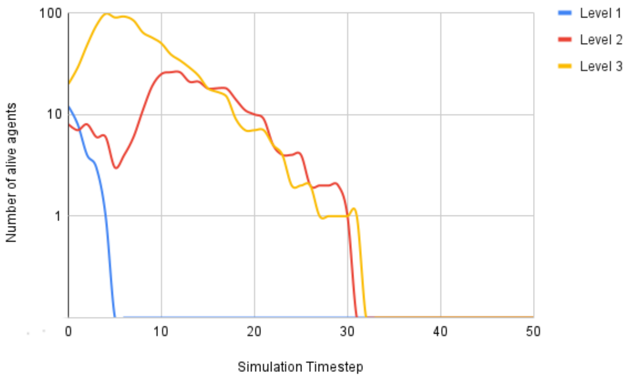


Figure 24: Number of alive agents per level over 50 timesteps in the ML-assisted simulation. The population declines gradually despite multiple interventions, with all levels going extinct by timestep 31. This trend supports the biodiversity indices shown in Figure 23b, where diversity collapses after a brief recovery phase.

One notable observation is the spike in Simpson's diversity at timestep 30, where it abruptly increased to 1.0. This occurred because only two agents remained, one from level 2 and one from level 3, resulting in perfect evenness among the surviving types. However, richness was only 0.22 and evenness was 0.63 due to the absence of level 1 agents. This emphasizes the importance of using a weighted combined diversity index, while Simpson's index may suggest high diversity, the weighted index ($\tilde{0}.7$) accounts for low species representation and better reflects the system's true state.

Therefore, the ML-enabled simulation showed improved resilience compared to the no-ML case. The system was able to partially recover and sustain diversity over a longer period. However, limitations were also evident: the model struggled to adapt once species were extinct (due to a lack of extinction-awareness), and the large 30×30 grid reduced agent interactions, increasing energy costs and accelerating population decline. These constraints ultimately led to system collapse despite repeated parameter optimizations.

5 Discussion

This section reflects on what the results of both phases show and how they relate to the main research goal. The focus is on what worked well, what was observed, and what can be taken from the experiments.

In **Phase 1**, the goal was to test whether the model could replicate core ecological dynamics in an abstract setting. The simulation integrated energy flow, trophic structure, spatial movement, and energy conservation. Results confirmed that these mechanisms worked together to produce ecologically realistic behavior. Population survival followed trophic logic: lower-level agents remained stable, while top-level agents were sensitive to prey loss and often collapsed first. This aligned with expectations from real ecosystems and showed that energy dependencies were correctly structured.

Energy conservation emerged as a key factor for long-term stability. Runs with strict energy rules produced more persistent populations, while relaxed settings led to quicker energy loss and system collapse. Spatial layout also shaped outcomes. Larger grids reduced interaction rates, which slowed energy flow and weakened the system's feedback loops. Conversely, when agents had lower movement costs or were concentrated in smaller spaces, they accessed resources more efficiently and maintained balance longer.

These findings demonstrate that the model's dynamics were not arbitrary. Agent behavior, population outcomes, and spatial clustering reflected ecological logic even in abstract simulations. This suggests that the core model captured essential patterns of ecosystem functioning, which provided a solid foundation for the adaptive optimization introduced in Phase 2.

In **Phase 2**, the aim was to test whether feedback-based parameter tuning could improve biodiversity outcomes. Results from all cases confirm that adaptive updates can help stabilize an ecosystem, but only under the right conditions. Compared to the no-ML setup, the Balancer module enabled partial recovery and reduced biodiversity loss. This directly supports SQ2 by showing that optimization influences stability not just through individual updates but through sustained, context-sensitive adjustments.

At the same time, the experiments show that feedback alone is not sufficient. In the setup without energy conservation, diversity collapsed despite frequent updates. This suggests that the optimizer requires structural support—such as energy constraints—to function effectively. Without such constraints, the system became unstable, and ML interventions had only short-term impact.

The unbalanced case highlights both the strengths and limits of the approach. The Balancer slowed the collapse and supported temporary recovery, but once extinction occurred, it could not reverse the loss. The model lacked awareness of irreversible dynamics such as species extinction. This underscores a key limitation: adaptive optimization depends on the system's remaining capacity to respond. It cannot compensate when critical thresholds have already been crossed.

Another observation is that early interventions were less effective. When the system was still unstable or had limited data, parameter updates did not lead to meaningful improvements. Over time, as more information became available, the quality of the interventions improved. This suggests the need for either more robust early-stage strategies or improved handling of uncertainty during initial phases.

Using multiple biodiversity metrics also helped in guiding the adaptation process. While Richness gave a quick overview of how many classes survived, it didn't capture how balanced the levels were. Adding Shannon and Simpson indices gave more insight into how the population was distributed. Since the Balancer used all three metrics in a weighted sum, the choice of weights clearly influenced the outcomes. In this thesis, weights were selected to reduce the dominance of richness, put more focus on evenness across levels, and produce an overall diversity score that captures a broader range of diversity aspects.

Another point to mention is the effect of randomness during the simulation. Even when using the same parameter settings, different results were seen between runs. This is expected in agent-based models, since agents interact based on their local environment and stochastic decision-making. While a fixed seed was used in some cases to reduce this variability, it still played a role. This makes it harder to compare results from single runs, but doesn't take away from the broader trends that were observed.

6 Further Work

This study introduced a two-phase approach: designing an abstract agent-based model to simulate ecosystem dynamics, and integrating an adaptive optimization module (the Balancer) to improve biodiversity stability. The results showed that this method can capture important ecological patterns and adapt parameters over time, but there are several directions worth exploring further.

The model was developed as an abstract simulation to test adaptive biodiversity management without grounding it in specific ecosystems. While abstraction has advantages in generality and flexibility, applying the approach to real-world ecological data could improve its relevance. Future research could incorporate field data to either parameterize the simulation or validate model outcomes against empirical benchmarks.

The combined biodiversity score relied on fixed weights for Richness, Shannon, and Simpson indices. Changing these weights or using alternative indices would likely affect optimization. Future work could explore adaptive weighting strategies based on ecosystem state or optimization feedback.

The ML model was trained offline. Implementing online learning could allow the model to update continuously during simulation, making predictions more accurate and better aligned with recent trends in the ecosystem.

Data collection involved simulation forking, but the number of forks was limited due to computational constraints. Parallelizing forks and recursive simulation forking, where each fork can recursively branch into new simulations, could improve coverage of the parameter space and create richer datasets without restarting from scratch each time.

Currently, mutations affect species traits but not their trophic level. In real ecosystems, this is part of what's known as eco-evolutionary dynamics[9], where changes in traits can shift a species' role in the food web. Future work could investigate how trait evolution could influence the food web structure, potentially allowing species to move "up" or "down" trophic levels. This would more accurately reflect natural evolutionary dynamics and could introduce new feedback loops in the simulation.

As the model becomes more complex, the parameter space grows rapidly. To reduce computational cost, future work could explore AutoML techniques or Bayesian optimization as alternatives to evolutionary strategies, particularly for managing high-dimensional, noisy search spaces.

[9] https://en.wikipedia.org/wiki/Eco-evolutionary_dynamic

7 Conclusion

To answer the main research question — ‘How can adaptive optimization of species traits help maintain biodiversity stability in agent-based ecosystem simulations?’ — a two-phase methodology was designed and implemented, with each phase addressing a specific sub-question.

In Phase 1, we focused on building an abstract agent-based simulation to mirror key ecological dynamics. This addressed the first sub-question: ‘How can an abstract agent-based ecosystem be designed to simulate biological dynamics such as energy flow and species behavior?’ We defined agent- and class-level parameters, implemented core simulation actions (move, eat, reproduce, die), and structured an interaction network based on trophic levels. The energy model ensured a balance between energy gained and lost across the system. Evaluation of the baseline model and its variations under different ecosystem conditions confirmed that the simulation behaved consistently with expected ecological dynamics.

Phase 2 introduced the Balancer module, which adaptively optimized agent parameters during the simulation using machine learning. This addressed the second sub-question: how adaptive optimization of species parameters affects biodiversity stability compared to no optimization. The Balancer used an XGBoost model to predict parameter combinations likely to improve biodiversity, guided by an objective function based on a weighted combination of Richness, Simpson’s, and Shannon’s diversity indices. Differential Evolution (DE) was then applied to efficiently explore and optimize the parameter space. The best parameter sets were applied as mutations within each trophic level. The optimization process was tested under different scenarios, including a baseline ecosystem with and without ML feedback, ML feedback without energy conservation, and an unbalanced initial setup to assess the Balancer’s recovery ability. In all cases, adaptive feedback contributed to improved or stabilized biodiversity outcomes, even when starting conditions were unfavorable. This demonstrates the Balancer’s potential to dynamically tune parameters and support ecosystem stability during simulation.

References

[Akiba et al., 2019] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[Austin and Cook, 1974] Austin, M. and Cook, B. (1974). Ecosystem stability: A result from an abstract simulation. *Journal of Theoretical Biology*, 45(2):435–458.

[Bazghandi, 2012] Bazghandi, A. (2012). Techniques, advantages and problems of agent based modeling for traffic simulation. *IJCSI International Journal of Computer Science Issues*, 9(1, No 3). School of Computer Engineering, Shahrood University.

[Begon et al., 2006] Begon, M., Townsend, C. R., and Harper, J. L. (2006). *Ecology: From Individuals to Ecosystems*. Blackwell Publishing.

[Borrelli and Ginzburg, 2014] Borrelli, J. J. and Ginzburg, L. R. (2014). Why there are so few trophic levels: Selection against instability explains the pattern. *Food Webs*, 1(1):10–17.

[Borshchev, 2013] Borshchev, A. (2013). *The Big Book of Simulation Modeling: Multimethod Modeling with AnyLogic 6*. AnyLogic North America, Chicago, IL.

[Broniec et al., 2021] Broniec, W., An, S., Rugaber, S., and Goel, A. K. (2021). Guiding parameter estimation of agent-based modeling through knowledge-based function approximation. In *Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021)*, Palo Alto, CA, USA.

[Calvez and Hutzler, 2005] Calvez, B. and Hutzler, G. (2005). Parameter space exploration of agent-based models. In Khosla, R., Howlett, R. J., and Jain, L. C., editors, *Knowledge-Based Intelligent Information and Engineering Systems*, pages 633–639, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Carnus et al., 2015] Carnus, T., Finn, J., Kirwan, L., and Connolly, J. (2015). Assessing the relationship between biodiversity and stability of ecosystem function–is the coefficient of variation always the best metric? *Ideas in Ecology and Evolution*, 7.

[Carson and Maria, 2000] Carson, Y. and Maria, A. (2000). Simulation optimization: Methods and applications. *Winter Simulation Conference Proceedings*.

[Centeno and Carrillo, 2001] Centeno, M. and Carrillo, M. (2001). Challenges of introducing simulation as a decision making tool. volume 1, pages 17–21 vol.1.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

[Cleveland Clinic, 2025] Cleveland Clinic (2025). Genetic mutations in humans. https://my.clevelandclinic.org/health/body/23095-genetic-mutations-in-humans. Accessed 4 july 2025.

[Colon et al., 2015] Colon, C., Claessen, D., and Ghil, M. (2015). Bifurcation analysis of an agent-based model for predator–prey interactions. *Ecological Modelling*, 317:93–106.

[Contributors, 2025] Contributors, J. (2025). Pycall.jl: Calling python functions from julia. https://github.com/JuliaPy/PyCall.jl. Accessed: 2025-07-06.

[Datseris et al., 2021] Datseris, G., Rezaee Vahdati, A., and DuBois, T. (2021). Agents.jl: A performant and feature-full agent based modelling software of minimal code complexity.

[Hussein, 2013] Hussein, S. (2013). Predator-prey modeling. *Undergraduate Journal of Mathematical Modeling: One + Two*, 3.

[Jones and Laughlin, 2009] Jones, T. and Laughlin, T. (2009). Learning to measure biodiversity: Two agent-based models that simulate sampling methods & provide data for calculating diversity indices. *The American Biology Teacher*, 71(7):406–410.

[Karterakis et al., 2007] Karterakis, S., Karatzas, G., Nikolos, I., and Papadopoulou, M. (2007). Application of linear programming and differential evolutionary optimization methodologies for the solution of coastal subsurface water management problems subject to environmental criteria. *Journal of Hydrology*, 342:270–282.

[Keller, 2012] Keller, A. A. (2012). Predator-prey evolution strategy for solving multi-objective optimization problems. In *Proceedings of the International Conference on Evolutionary Computation*, Paris, France.

[Kitikidou et al., 2024] Kitikidou, K., Milios, E., Stampoulidis, A., Pipinis, E., and Radoglou, K. (2024). Using biodiversity indices effectively: Considerations for forest management. *Ecologies*, 5(1):42–51.

[Korobeinikov, 2009] Korobeinikov, A. (2009). Stability of ecosystem: Global properties of a general predator-prey model. *Mathematical medicine and biology : a journal of the IMA*, 26:309–21.

[(Letcher et al., 2013] (Letcher, R., Jakeman, A., Barreteau, O., Borsuk, M., Elsawah, S., Hamilton, S., Henriksen, H., Kuikka, S., Maier, H., Rizzoli, A.-E., van Delden, H., and Voinov, A. (2013). Selecting among five common modelling approaches for integrated environmental assessment and management. *Environmental Modelling Software*, 47:159–181.

[Lindeman, 1942] Lindeman, R. L. (1942). The trophic-dynamic aspect of ecology. *Ecology*, 23(4):399–417.

[Luan, 2024] Luan, T. (2024). A comprehensive review of simulation technology: Development, methods, applications, challenges and future trends. *International Journal of Emerging Technologies and Advanced Applications*, 1:9–14.

[Luke et al., 2005] Luke, S., Cioffi, C., Panait, L., Sullivan, K., and Balan, G. (2005). Mason: A multiagent simulation environment. *Simulation*, 81:517–527.

[Macal and North, 2011] Macal, C. and North, M. (2011). Introductory tutorial: Agent-based modeling and simulation. *Proceedings - Winter Simulation Conference*, 2015:1451–1464.

[Majic et al., 2025] Majic, I., Scholz, J., Röbl, D., Bulbul, R., Lampoltshammer, T., Kleinlehner, M., and Neubauer, P. (2025). Agent based modeling (abm) and ai integration for smart tourism simulations.

[Malishev and Kramer-Schadt, 2021] Malishev, M. and Kramer-Schadt, S. (2021). Movement, models, and metabolism: Individual-based energy budget models as next-generation extensions for predicting animal movement outcomes across scales. *Ecological Modelling*, 441:109413.

[Margalef, 1957] Margalef, R. (1957). Information theory in ecology. *General Systems*, 3:36–71.

[Masad and Kazil, 2015] Masad, D. and Kazil, J. (2015). Mesa: An agent-based modeling framework. pages 51–58.

[McLane et al., 2011] McLane, A. J., Semeniuk, C., McDermid, G. J., and Marceau, D. J. (2011). The role of agent-based models in wildlife ecology and management. *Ecological Modelling*, 222(8):1544–1556.

[Montevechi et al., 2015] Montevechi, J. A. B., Fernandes Pereira, T., Silva, C., Miranda, R., and Galvao Scheidegger, A. P. (2015). Identification of the main methods used in simulation projects. pages 3469–3480.

[Murphy et al., 2020] Murphy, K., Ciuti, S., and Kane, A. (2020). An introduction to agent-based models as an accessible surrogate to field based research and teaching. *Ecology and Evolution*, 10:1–17.

[Neil et al., 2025] Neil, E., Carrella, E., and Bailey, R. (2025). Integrating agent-based models into the ensemble ecosystem modelling framework: A rewilding case study at the knepp estate, uk. *Ecological Solutions and Evidence*, 6(1):e70022. e70022 ESO-24-10-186.R1.

[Ngo-Hoang, 2020] Ngo-Hoang, D.-L. (2020). The three methods in simulation modeling [chapter 2. the three methods in simulation modeling].

[Niemann et al., 2021] Niemann, J.-H., Klus, S., and Schütte, C. (2021). Data-driven model reduction of agent-based systems using the koopman generator. *PLOS ONE*, 16:e0250970.

[Nugroho and Uehara, 2023] Nugroho, S. and Uehara, T. (2023). Systematic review of agent-based and system dynamics models for social-ecological system case studies. *Systems*, 11(11).

[Ozkal et al., 2025] Ozkal, S., Bertone, E., and Stewart, R. A. (2025). A systematic review of agent-based modelling in agricultural water trading. *Water*, 17(6).

[Pielou, 1966] Pielou, E. (1966). The measurement of diversity in different types of biological collections. *Journal of Theoretical Biology*, 13:131–144.

[Purathekandy et al., 2024] Purathekandy, A., Oommen, M. A., Wikelski, M., and Subramani, D. N. (2024). An agent-based model of elephant crop raid dynamics in the periyar-agasthyamalai complex, india.

[SciPy Developers, 2025] SciPy Developers (2025). scipy.optimize.differential_evolution — scipy documentation. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html. Accessed: 2025-07-06.

[Semeniuk et al., 2012] Semeniuk, C., Musiani, M., Hebblewhite, M., Grindal, S., and Marceau, D. (2012). Incorporating behavioral–ecological strategies in pattern-oriented modeling of caribou habitat use in a highly industrialized landscape. *Ecological Modelling*, 243:18–32.

[Shannon, 1948] Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423 and 623–656.

[Sherwin and Prat i Fornells, 2019] Sherwin, W. B. and Prat i Fornells, N. (2019). The introduction of entropy and information methods to ecology by ramon margalef. *Entropy*, 21(8):794.

[Simpson, 1949] Simpson, E. H. (1949). Measurement of diversity. *Nature*, 163:688.

[Su et al., 2024] Su, H., Wang, Z., Ma, L., Qin, R., Chang, T., Zhang, Z., Yao, J., Li, X., Li, S., Hu, X., Wei, J., Yuan, F., Adi, H., Shi, Z., Li, H., and Zhou, H. (2024). Multitrophic diversity of the biotic community drives ecosystem multifunctionality in alpine grasslands. *Ecology and Evolution*, 14(11):e70511. e70511 ECE-2024-07-01492.R1.

[Testolina et al., 2019] Testolina, P., Lecci, M., Rebato, M., Testolin, A., Gambini, J., Flamini, R., Mazzucco, C., and Zorzi, M. (2019). Enabling simulation-based optimization through machine learning: A case study on antenna design. pages 1–6.

[Vahdati, 2019] Vahdati, A. (2019). Agents.jl: agent-based modeling framework in julia. *Journal of Open Source Software*, 4(42):1611.

[Vlad et al., 2024] Vlad, A. I., Romanyukha, A. A., and Sannikova, T. E. (2024). Parameter tuning of agent-based models: Metaheuristic algorithms. *Mathematics*, 12(14):2208.

[Wilensky, 1997] Wilensky, U. (1997). Netlogo wolf sheep predation model. http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

A Energy Distribution Function Implementation

Listing 4 Energy distribution function

```
# Distribute energy of a dying agent to nearby agents of other classes
function distribute_energy!(agent, model, energy_to_split)
    nearby = nearby_agents(agent, model, 0)
    receivers = [a for a in nearby if a.class_id != agent.class_id]

    # No receivers -->  energy lost to environment
    if isempty(receivers)
        agent.E_loss_env += energy_to_split
        finalize_death!(agent, model)
        return
    end

    # Compute total weight by class level
    total_weight = sum(class_level.level[a.class_id] for a in receivers)
    assigned_energy = 0.0

    # Distribute energy based on weight
    for other in receivers
        weight = class_level.level[other.class_id]
        share = round((weight / total_weight) * energy_to_split, digits=4)
        other.E_i_id += share
        other.E_gain_id += share
        assigned_energy += share
    end

    # Track losses to other agents and environment
    agent.E_loss_id += assigned_energy
    agent.E_loss_env += (energy_to_split - assigned_energy)
    finalize_death!(agent, model)
end
```

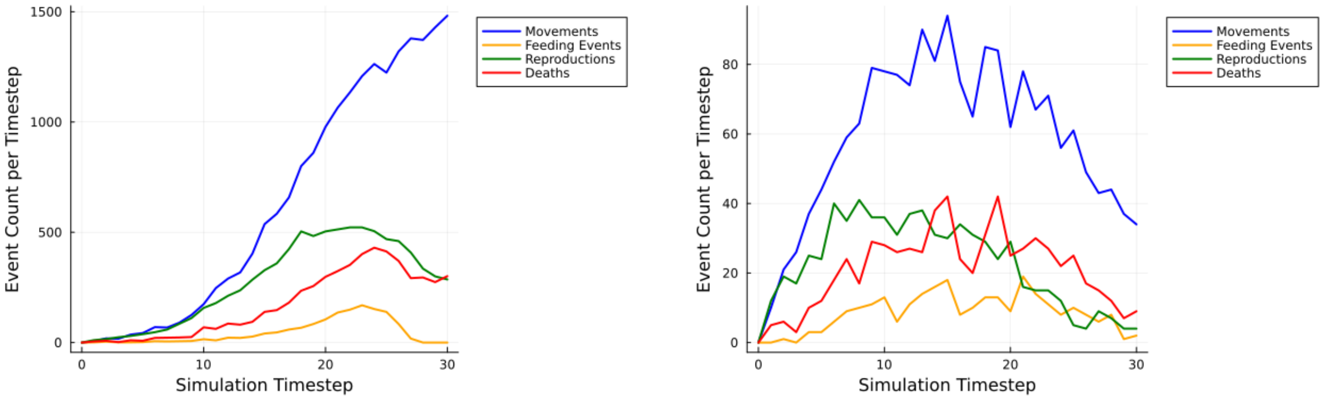
B Energy Balance Check Data of the Baseline Simulation

Table 8: Energy balance data over 30 timesteps for the baseline simulation. The table includes total system energy, total energy gained and lost through actions, and energy lost to the environment. These values were used in the plots shown in Section 4.1.2.

Time	Total Energy	Total Gain	Total Loss	Lost to Env.
0	46.6943	0	0	0
1	36.0016	14.7213	14.7213	10.6927
2	29.0985	19.299	19.299	6.9031
3	26.5525	20.6502	20.6502	2.546
4	21.1833	22.3885	22.3885	5.3692
5	18.8978	23.9161	23.9161	2.2855
6	17.1263	26.0364	26.0364	1.7715
7	16.1975	26.5797	26.5797	0.9288
8	15.5897	27.5781	27.5781	0.6078
9	15.0104	28.4641	28.4641	0.5793
10	14.3454	31.0251	31.0251	0.665
11	14.095	31.7381	31.7381	0.2504
12	13.9647	32.4761	32.4761	0.1303
13	13.7857	33.3727	33.3727	0.179
14	13.6192	34.0431	34.0431	0.1665
15	13.4805	34.7505	34.7505	0.1387
16	13.2544	35.5545	35.5545	0.2261
17	13.1596	36.0748	36.0748	0.0948
18	13.0465	36.7878	36.7878	0.1131
19	12.9351	37.4382	37.4382	0.1114
20	12.8108	37.9283	37.9283	0.1243
21	12.7124	38.389	38.389	0.0984
22	12.607	38.82	38.82	0.1054
23	12.4727	39.2294	39.2294	0.1343
24	12.3481	39.7357	39.7357	0.1246
25	12.2078	40.0783	40.0783	0.1403
26	12.0706	40.2951	40.2951	0.1372
27	11.9482	40.4318	40.4318	0.1224
28	11.8355	40.5146	40.5146	0.1127
29	11.7476	40.5869	40.5869	0.0879
30	11.6548	40.6503	40.6503	0.0928

C Additional Plots Supporting Energy Conservation Effects

To support the analysis in Section 4.1.6, this appendix presents a plot comparing agent behaviors, including feeding, reproduction, movement, and death, over time in both cases. The figure highlights the sharp decline in activity for the non-conserving system and the sustained or increasing activity in the conserving system. These patterns demonstrate the impact of energy conservation on ecosystem stability and agent survival.



(a) Agent behaviors in the energy-conserving system. Feeding, reproduction, and movement actions remain frequent and stable over time.

(b) Agent behaviors in the non-conserving system. Activity declines quickly, reflecting loss of energy and population instability.

Figure 25: Comparison of agent behavior patterns over time between the energy-conserving (a) and non-conserving (b) cases.