OPTIMIZING BIOENERGETIC FOOD WEB MODELS OF ECOSYSTEMS USING GAMIFICATION AND MACHINE LEARNING

A thesis presented to the faculty of
San Francisco State University
In partial fulfilment of
The Requirements for
The Degree

Master of Science
In
Computer Science

by

Benjamin Robert Saylor

San Francisco, California

May 2017

CERTIFICATION OF APPROVAL

I certify that I have read *OPTIMIZING BIOENERGETIC FOOD WEB MODELS OF ECOSYSTEMS USING GAMIFICATION AND MACHINE LEARNING* by Benjamin Robert Saylor and that in my opinion this work meets the criteria for approving a thesis submitted in partial fulfillment of the requirements for the degree: Master of Science in Computer Science at San Francisco State University.

Anagha Kulkarni
Assistant Professor of Computer Science

Ilmi Yoon
Professor of Computer Science

Pleuni Pennings
Assistant Professor of Biology

OPTIMIZING BIOENERGETIC FOOD WEB MODELS OF ECOSYSTEMS USING GAMIFICATION AND MACHINE LEARNING

Benjamin Robert Saylor
San Francisco State University
2017

Ecosystems are complex systems with many interdependent participants. Bioenergetic models of population dynamics help provide insight into specific aspects of ecosystem behavior. Due to the complex, nonlinear behavior of these models, and the large number of input parameters, it is difficult to parameterize them to correctly reflect real-world phenomena. We address this problem in the context of allometric trophic network models, a category of bioenergetic models based on food webs. Using custom simulation software to generate large numbers of simulated ecosystems, we apply machine learning to help navigate the large parameter space, revealing combinations of model parameters that result in sustaining ecosystems. Furthermore, we apply gamification to take advantage of human intuition, using the insights gained from the machine learning process to provide automatic guidance to players.

I certify that the Abstract is a correct representation of the content of this thesis.

Chair, Thesis Committee Date

ACKNOWLEDGMENTS

I would like to thank Dr. Anagha Kulkarni, my committee chair, for giving me the opportunity to embark on this journey, for her critical observations, and for her gentle pragmatism; Dr. Ilmi Yoon for introducing me to gamification and computational ecology, and for recognizing my potential; and Dr. Pleuni Pennings for stepping up at the last moment to serve a necessary and appreciated role.

A special thanks to the late Dr. S. Jonathan Stern for introducing me to the foundations of ecology and for lending an ecologist's perspective to my work. A pioneering expert on the ecology of minke whales and a kind human being, Dr. Stern will be missed by many.

I would also like to thank Dr. Neo Martinez and Dr. Rich Williams for sharing their time and expertise on allometric trophic network models, and the numerous faculty, staff, and students at SFSU who made this work a possibility.

Finally, I would like to thank my family and Zia Rauwolf for their unconditional love and support that kept me afloat during this challenging undertaking.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Ecosystems are extremely complex systems involving many entities (such as plants, animals, and nutrients) and the relationships between these entities. These entities and their relationships are characterized by numerous measurable attributes. Ecologists study ecosystems by empirical observation and analysis, and often by using sophisticated mathematical or computational models informed by empirical observation. Ecological models apply various approaches to managing the complexity of real-world ecosystems, depending on which aspects of the ecosystems are of particular interest in a given context, and at what level of detail.

Population dynamics – how the sizes of populations change over time – is one important aspect of an ecosystem. Ecological population dynamics models are concerned with how populations of species in ecosystems change over time depending on various factors in their environment and relationships to each other – for example, the relationship between predator and prey.

Allometric trophic network (ATN) models are the focus of this work. ATN models describe population dynamics in ecosystems in terms of energy flow through food webs. This family of models was developed by researchers in ecology including Yodzis & Innes (1992) [1], Williams & Martinez (2004) [2], and Williams et al. (2007) [3], and continues to form the basis of a growing body of ecological research. The word "allometric" refers to a key characteristic of these models, which is that the metabolic rates of different species are estimated based on their body size. "Trophic" refers to feeding relationships. "Network" refers to food webs.

ATN models use *food webs* as the fundamental structures by which relationships between species are defined. A food web simply describes the set of species in an ecosystem along with their feeding relationships – that is, it describes who eats whom in an ecosystem [4]. As bioenergetic population dynamics models, ATN models describe the flow of energy or biomass over time from species to species, through a food web, and the resulting changes in populations, or total species biomass, over time. Biomass is introduced into the system by plants – the primary producers. It flows along branching routes from species to species in the food web as predator consumes prey, and exits the system due to transfer losses, metabolism, and mortality. The abundance, or total biomass, of each species depends on the food web structure, the availability of its prey (for animals) and the populations of its predators. Additionally, it depends on numerous parameters describing such important factors as metabolic rate, body size, the environment's carrying capacity, food assimilation efficiency, and *functional response*, or how the preference of predators for various prey species changes based on prey abundance [3]. The values of these parameters are specific to individual species or pairwise relationships.

While this multitude of parameters enables ATN models to use food webs as a basis for rich descriptions of ecosystems, it also poses significant problems for constructing and parameterizing models. If a food web is to be accurately modeled, not only must its structure be correctly specified, but the parameters of the model must be correctly set. However, parameter values are often unknown and difficult or impractical to measure with the required degree of accuracy. Moreover, because the models are highly complex and nonlinear, the effect of changing a parameter also depends on many other parameters. A "brute force" approach to model fitting or optimization requires searching an extremely large parameter space – a combinatorial optimization problem. Only a fraction of this parameter space has been explored by researchers thus far. While the ATN model has strong theoretical support and has been successfully parameterized to model ecosystems with useful accuracy, the problem of determining parameter values that lead to self-sustaining simulated ecosystems remains a major challenge for researchers [5].

Software implementations of the ATN model are capable of producing vast quantities of simulation data, presenting an opportunity to use machine learning techniques capable of discerning patterns in large datasets. Our approach is to use machine learning techniques to reveal promising ranges of parameter values that may lead to an optimal solution.

The other angle taken in this work is the use of *gamification* to leverage human intuition to explore the ATN model parameter space within the framework of the online multiplayer ecology game *World of Balance*. Using the insights gained from the machine learning process, we augment the game-within-a-game called *Convergence*, developed by SFSU students, with hints to guide players in the search for parameter values that accurately replicate a target ecosystem as closely as possible, as well as by adding many more simulated ecosystems to the Convergence ecosystem library.

1.1 Contributions

The goal of this research is to develop methods for constructing and parameterizing simulated food webs that are self-sustaining. We approach this goal from the two perspectives of machine learning and gamification.

In this work, we have made a number of contributions toward this goal.

We developed a graph sampling algorithm for deriving smaller food webs from larger ones (Section 4.1). Smaller food webs are more manageable for gameplay and analysis. The algorithm is different from general-purpose graph sampling algorithms in that it preserves a viable food web structure.

We developed new ATN simulation code for World of Balance that corrects errors in the previous version and provides a 90% speedup. This allowed us to run hundreds of thousands of simulations for long durations, enabling analysis of slow-developing patterns. This code is now packaged as an open-source Java library called ATN Simulator.[1]

[1]https://github.com/brsaylor/atn-simulator

An important new feature of the simulation code is steady state detection (Section 4.4). This is based on our observation of both constant and oscillating steady states of the model after which the subsequent dynamics of the simulation can be inferred. This has applications in eliminating unnecessary computation and identifying self-sustaining ecosystems.

We developed a new formula for the World of Balance environment score (Section 4.2.3). This formula has several advantages over the old version, and rewards players for maintaining ecologically important characteristics in their in-game ecosystems.

We showed that standard machine learning models can predict simulation outcomes with good accuracy given the ATN model parameters as input (Section 4.3.5).

We made two contributions toward enhancing the Convergence game. We developed a method of generating sustaining and visually interpretable simulations, using it to produce a preliminary library of about 200 simulations for 14 different food webs (Section 4.9). We also developed a machine-learning-based method of generating parameter range hints for players (Section 4.3).

Finally, we developed a method of using machine learning to iteratively narrow the large parameter search space by identifying promising regions (Section 4.10).

Chapter 2

Background

2.1 Allometric trophic network models

An ATN model consists of a system of ordinary differential equations. These equations were first described by Yodzis & Innes (1992) [1], were extended to multi-species ecosystems by Williams & Martinez (2004) [2] and Williams et al. (2007) [3], and further extended by Boit et al. 2012. [6] The system of equations defines the change in biomass of each species as a function of the current biomass of each species in the food web.

At a very high level, the system can be expressed as

$$B_i' = f(\mathbf{B}) \tag{2.1}$$

where B_i' is the derivative of the biomass of species i and $\mathbf{B}$ is the vector of the biomasses of all species in the ecosystem.

The details of the equations and their parameters depend on the specific variation of ATN model in question. The equations described here are for the ATN model implemented in *ATN Simulator* and *World of Balance*, described in Section 2.2.

For producers (plants), B_i' includes terms representing growth from photosynthesis and loss from being consumed by other species. It is defined as follows:

$$B_i' = r_i B_i G_i(\mathbf{B}) - \sum_{j \in predators} (x_j y_{ji} \alpha_{ji} F_{ji} B_j / e_{ji}) \tag{2.2}$$

The first term represents the growth of producer i, and is expressed in terms of a base growth rate r_i, the producer's current biomass B_i, and its population density-dependent growth rate $G_i(\mathbf{B})$, described below. The second term, the summation, represents loss due to predation. The loss to each predator j depends on j's metabolic rate x_j, j's maximum ingestion rate y_{ji} when consuming i, j's functional response F_{ji} with respect to i (described below), j's current biomass B_j, and how efficiently j can assimilate i, e_{ji}.

For consumers, B_i' represents growth from feeding, metabolic loss, and loss from being consumed by predators. It is defined as follows:

$$B_i' = \sum_{j \in prey} (x_i y_{ij} \alpha_{ij} F_{ij} B_i) - \sum_{j \in predators} (x_j y_{ji} \alpha_{ji} F_{ji} B_j / e_{ji}) - x_i B_i \tag{2.3}$$

The first summation represents population growth from feeding. The second summation represents loss due to being consumed. The last term represents metabolic loss, or the consumption of energy to stay alive.

The growth function $G_i(\mathbf{B})$ represents how a producer's growth depends on its population density and the carrying capacity of the environment, or availability of resources, such as nutrients, required for survival. Various versions of this growth function have been proposed. The simplest, used in this work, is a logistic growth function that causes the growth rate to decrease to 0 as the producer approaches carrying capacity K_i. This function is defined as follows:

$$G_i(B) = 1 - \frac{B_i}{K_i} \tag{2.4}$$

Boit et al. (2012) [6] use a different logistic growth function that models producer competition for a shared system-wide carrying capacity K_s:

$$G_i(B) = 1 - \frac{\sum_{j \in producers} c_{ij} B_j}{K_s} \tag{2.5}$$

In this equation, c_{ij} are competition coefficients, allowing some producers an advantage over others in occupying the available carrying capacity. We implemented equation 2.5 in World of Balance and ATN Simulator, but used the simpler equation 2.4 for most of the work.

The functional response F_{ij} is one of the most complex aspects of the model. In ecology, functional response describes how the feeding rates of predators depend on the population densities of its prey [3]. Functional response encompasses factors such as preference of predators for their various potential prey, depending on prey abundance, hunting and prey handling time, and a saturation point at which the predator cannot consume the prey any faster. Of the many proposed models for functional response in the literature, we use the following:

$$F_{ij} = \frac{B_j^{1+q_{ij}}}{\sum_{m \in prey} \alpha_{im} B_m^{1+q_{im}} + B_{0ij}^{1+q_{ij}}} \tag{2.6}$$

The first new parameter here is q_{ij}, the functional response control parameter, which allows adjusting the shape of the response curve from a Holling Type II functional response ($q = 0$) to a Holling Type III functional response ($q = 1$). Other new parameters are α_{ij} and B_{0ij}, which represent the relative half saturation density, and half saturation density, respectively, controlling the saturation points of consumption rates.

Table 2.1 summarizes the model parameters used in these equations.

Parameter	Description
α_{ij}	Relative half-saturation density of predator i when consuming prey j
B_{0ij}	Half-saturation density of predator i when consuming prey j
c_{ij}	Competition between producers i and j for shared carrying capacity
e_{ij}	Assimilation efficiency of prey j by predator i
K_i	Carrying capacity of producer i
K_s	System-wide carrying capacity
q_{ij}	Functional response control parameter
r_i	Growth rate of producer i
x_i	Metabolic rate of consumer i
y_{ij}	Maximum ingestion rate of predator i when consuming prey j

Table 2.1: ATN model parameters

An ATN model, like all scientific models, is a simplification of reality that allows researchers to study particular aspects of a real-world system. As such, ATN models focus on population dynamics as driven by trophic relationships. They do not directly account for factors such as climate, migration, terrain, or seasonality. They do not explicitly model spatial relationships.

Factors such as these can be incorporated by combining multiple ATN models with different parameters, by strategically manipulating model parameters, or by modifying biomass directly over the course of a simulation. For example, Kuparinen et al. (2006) [7] model populations of fish moving through different life stages by representing each life stage as a separate guild (treated as a species in the model). They modify the biomass of each guild over time to model the aging process. They also modify model parameters over time to model the seasonal effects of fishing.

In this work, we study the behavior of individual ATN models whose parameters are fixed over the course of a simulation, with no external modification of species biomasses.

2.2 Gamification and *World of Balance*

Gamification is a way of crowdsourcing human "computational power" to solve important problems that are difficult to solve by computational methods alone. Gamification takes advantage of human intuition by presenting problems in an engaging, enjoyable game format. Players' approaches and solutions to these problems are then analyzed to guide further research.

A prominent gamification success story is *Foldit* [8], a 3D puzzle game in which players attempt to solve the problem of folding protein molecules. Players of *Foldit* have solved important protein-folding problems that were previously unsolved by computational methods. *Foldit*'s successes include an important breakthrough in AIDS research [9].

World of Balance is an online, multiplayer computer game in which players act as stewards of virtual ecosystems based on a recently compiled food web of species inhabiting the Serengeti ecosystem. The virtual ecosystems in World of Balance are simulated using an ATN model.

Yoon et al. (2013) [5] developed World of Balance in part to take a gamification approach to increasing the explored parameter space of ATN models, as well as to educate students about ecology and the Serengeti ecosystem. Yoon et al. demonstrated remarkable success in the educational goal, showing that a group of students who played the game gained significantly more knowledge than a control group who were assigned relevant reading materials. The data from the game were provided to scientists, who were then able to examine the players' intuitively guided search of the parameter space.

World of Balance consists of a lobby – a central hub where players maintain their ecosystems – and a number of different "mini-games". Players can play mini-games to earn credits with which they can purchase additional species for their lobby ecosystems.

Convergence and *Multiplayer Convergence* are two such mini-games. Convergence presents players with two graphs of simulated ecosystem biomass, one of which they can control by manipulating ATN model parameters using sliders, and a second which has fixed parameter values unknown to the player. The goal of the game is to find the parameter values that cause the player's graph to match the target graph. Multiplayer Convergence has the same overall structure and objective, but instead of being a solitaire game, players compete with each other for the best match. Players place bets in poker-style rounds. In Sections 4.3 and 4.9.1, we present proposed enhancements to the Convergence games.

Chapter 3

Literature Review

Williams et al. (2007) [3] describe the approach taken to develop the Allometric Trophic Model as an extension of prior work in population dynamics modeling by Yodzis and Innes (1992) [1]. Yodzis and Innes identified two fundamental aspects of organisms useful in modeling population dynamics: body size and metabolic type. Body size, closely correlated with surface-to-volume ratio, is an important factor in metabolism. It is also, conveniently, easy to measure. The metabolic types defined by Yodzis and Innes are plants, invertebrates, endotherm (warm-blooded) vertebrates, and ectotherm (cold-blooded) vertebrates. Once body size and metabolic type is determined for an organism, it can be modeled as undifferentiated biomass. The total biomass of a species depends on gains from net primary production (for plants), feeding (for animals), and losses from metabolism, predation, and mortality.

An important concept in population dynamics models is *functional response*, which defines the rate of consumption of a particular prey by a particular predator, depending on the prey density. Various formulations of functional response have been defined, and the exact formulation chosen for a model has a significant effect on model output.

The bioenergetic model of Yodzis and Innes (1992) [1] describes a relationship between two species, but it can be generalized to any number of species. This generalization is the contribution of Williams et al. (2007) [3], which extends the model to many species and functional responses.

Machine learning techniques have been widely applied in ecology and this intersection is well-represented in the ecological literature. Baskerville et al. (2011) [10] describe an application of Bayesian modeling to find group structure in a Serengeti food web. They cite prior work that tried to describe food web structure in terms of *compartments*. Compartments are defined in network topology as sets of nodes with many internal links and few external links. This prior work suggests that compartments promote stability by slowing the propagation of disturbances. However, Baskerville et al. argue that trophic *guilds* – "sets of species that feed on, and are fed on, by similar sets of species" – are more important to ecosystems than compartments. They focus on a more general concept, the *group*. The directional relationship between two groups i and j is the probability that a species in group i is eaten by a species in group j. Using a Bayesian network model with Markov-chain Monte Carlo sampling, they determine the most probable assignment of species to groups. They find that a detailed group structure emerges, where groups largely correspond to known trophic guilds. The high taxonomic resolution of plant species in their food web data allows the group model to reveal patterns of spatial coupling between groups, supporting ideas proposed by others.

Muttil and Chau (2007) [11] describe the use of artificial neural networks (ANN) and genetic programming (GP) to identify variables that can be used to predict harmful algal blooms (HAB). The "red tide", a type of HAB, results from eutrophication and increases in salinity and temperature, and has serious negative consequences including mariculture loss due to oxygen depletion, anoxia, and shellfish poisoning. They use data from Tolo Harbor in Hong Kong, where the red tide has been a persistent problem due to pollution including municipal and livestock waste discharge. The data include variables measuring water nutrient content, water quality, water temperature, climate, and chlorophyll-a (chl-a) as an indicator of algal biomass. Including time-lagged versions of these variables results in a total of 63 variables. They apply ANN and GP – separately, in independent experiments – to do feature selection and identify the variables which are the most significant predictors of algal biomass. Both ANN and GP learn correlated patterns between input and output variables in nonlinear and complex data, and account for interaction between variables that are not significant individually, making the techniques well-suited to ecological modeling. GP operates on parse trees describing functions of the input variables. The initial parse trees are randomly generated, and then iteratively altered and selected for "fitness" – accurate prediction of the output variables. These alterations include random mutation and crossover, or random swaps of sub-trees between selected individual trees. To use GP for feature selection, the authors define a significance measure to be the number of times an input variable is selected in one of these parse trees. To use ANN for feature selection, they create an ANN with 63 input nodes – one corresponding to each input variable – and one output node, chl-a. After training the ANN, they determine the significance of each input variable by computing the sum of the absolute values of the connection weights from the corresponding input node to all hidden nodes. They validate this methodology by applying it to feature selection of the Bernoulli's equation from fluid mechanics, supplying the ANN and GP models with random input variables in addition to true values for its actual inputs and output. Both models successfully identify the real input variables, as determined by the authors' significance measures. However, their main conclusion – that time-lagged chl-a is enough to predict the future quantity of chl-a in the harbor with reasonable accuracy – seems dubious, given the established links between human activity and HABs in the area.

Shan et al. (2006) [12] explore four machine-learning algorithms – decision trees, artificial neural networks (ANN), support vector machines (SVM), and genetic programming (GP) – to analyze data describing the habitat, abundance, and spatial distribution of the southern brown bandicoot in South Australia. Counts of "diggings" are used as an indicator of abundance by location. They attempt to identify relationships between bandicoot population and vegetation, soil, fire history, and geomorphology. They note that the small size and highly skewed nature of this dataset has made it very difficult to analyze in the past, and they perform their analyses on both the original (unbalanced) dataset and on a resampled dataset that balances class distribution. They find that while SVM and ANN perform marginally better, the output of decision trees and neural networks is more readily interpretable by humans. Given the fairly poor overall predictability of the data using any of the models, they therefore favor decision trees and ANNs for their interpretability.

Gutenkunst et al. (2007) [13] describe the common pattern in systems biology models that they call parameter "sloppiness." In many of these models, which are generally nonlinear and have many parameters, a *collective* fit to the observed data can be obtained without tightly constraining individual parameter values. "Sloppiness" refers to the ability of parameters, individually or in combination, to move through ranges of values without significantly altering model behavior. The authors test this idea by performing sensitivity analyses of 17 different published systems biology models. They first define a measure χ^2 for comparing model output under the published parameter values with model output under varied parameters. These models produce sets of time series data as output, much like ATN models, and χ^2 is essentially a continuous mean squared error calculation comparing all of the time series produced by the model under the original and altered parameters. They then define a Hessian matrix $H^{\chi^2}_{j,k}$, where the rows j and columns k correspond to model parameters θ_j and θ_k.[1] The authors argue that models can make useful predictions based on collective fits even while uncertainty remains about individual parameter values, provided that sensitivity analysis has been

[1] A Hessian matrix is a square matrix containing all of the second-order partial derivatives of a function. Each row and column corresponds to a variable in the function. The eigenvalues of a Hessian matrix evaluated at a point of the function indicate whether the function is curved up or down at that point, if the point is a saddle point, or if the curvature cannot be determined without further information.

done to quantify these uncertainties. Further, they argue that determining parameter values from collective fits is difficult due to parameter sloppiness. Finally, they point out that direct parameter measurements are often impractical for sloppy models, because the required measurement precision is determined by the smallest dimension of the region of "slop" within the parameter space.

Berlow et al. (2009) [14] apply classification and regression trees (CART) and ATN models to the problem of predicting *interaction strengths* between pairs of species in food webs. They define the interaction strength I between two species T and R as the change in time-averaged total biomass of T when R is removed from the food web. This essentially represents the extinction of species R, but the authors note that the same methods could be extended to study other phenomena such as over-harvesting. They also analyze per-capita I, which measures interaction strengths in terms of population density rather than total biomass. Using 600 food webs generated by the niche model, each with 10-30 species, they remove one species at a time and calculate the interaction effects on all other species. The resulting data, including 90 independent variables and I as the dependent variable, is used to train a CART model. The algorithm selects only 7 significant variables to build a model that explains over 86% of the variance in I. Large (positive negative) values of I are observed when both T and R have high biomass, a low degree of separation, and only one network path between them. A CART model using $\log |I|$ as the independent variable yields an even simpler model, with two variables explaining over 66% of the variance: the log of the biomass of T with R present and the log of the biomass of R. A CART model based on per-capita I produces a similarly simple model, with R's body mass and biomass and T's biomass predicting most of the variance. The authors point to the remarkable fact that seemingly important variables, such as consumer-resource body size ratio and functional response type, provide little explanatory power, and that the results hold regardless of network structure. They also argue that trophically-driven species interactions are well-predicted by ATN models, and that if model output differs from observed data, it can be inferred that nontrophic factors in the ecosystem are significant. They support these arguments using a field experiment in an intertidal ecosystem including 3 primary species: whelks, mussels, and barnacles, in which the barnacles are known to have nontrophic mediating effects, and find that the ATN model makes accurate predictions when the mussels are removed, but less accurate predictions when they are present.

Boit et al. (2012) [6] acknowledge that progress in accurately modeling empirically observed population dynamics has been slow, but describe their own successful experiment in fitting a modified ATN model to empirical biomass data. The ecosystem studied is that of Lake Constance, a large and deep lake in central Europe inhabited by a rich community of plankton and fish. Lake Constance has been well-studied, and detailed data was collected on a weekly or bi-weekly basis between 1987 and 1996. Strong seasonal effects controlling plankton growth are observed there. The researchers first defined a food web consisting of 24 guilds and 107 feeding relationships, then constructed a dataset representing an average year, and finally made iterative refinements to the

ATN model until a good fit to the observed data was achieved. These refinements are based on knowledge of factors specific to Lake Constance, but many also generalizable to other ecosystems, and include the following: lowered metabolic and growth rates for prokaryotes, deviating from strictly allometrically scaled values; separate metabolic rates for respiration vs. biomass production; prey resistance; a closed detrital loop; and abiotic forcing, which refers to the physical factors that dominate species abundance in a seasonal pattern such as irradiance, vertical mixing, and temperature changes. The authors largely achieved a good fit to the observed data based on a number of similarity measures, and their work suggests a general strategy for applying ATN models to natural ecosystems.

In her master's thesis, Cotter (2015) [15] describes her work toward fitting ATN models to empirical data. To this end, she applies and integrates two approaches, machine learning and gamification, to identify parameter changes that significantly affect species biomass in simulated food webs. Her approach to the machine learning component can be described as follows: She first generated 482 random food webs, and then generated an additional counterpart to each one, which was nearly identical except for a random parameter change. Based on the simulated data for each ecosystem and its altered counterpart, she calculated a number of species-specific dependent variables, each describing a characteristic in the data for that species that differed between the original and altered versions. These dependent variables are average biomass, linear regression slope, peak biomass, time of peak, time of extinction, and biomass curve wavelength. She then used the CART algorithm (following the example of Berlow et al. (2009) [14]) to build a model for predicting changes in these measures based on the ATN model parameters, as well as several attributes computed from the data, and an encoding of the parameter change between the original and altered ecosystems. She found that change in average biomass was predictable based on all 3 primary ATN parameters (metabolic rate, growth rate, and carrying capacity). She incorporated these findings into the Convergence game, which she developed, as ‘hints’ to guide players as they adjust parameters with the objective of visually fitting matching biomass curves of a simulated ecosystem to a target graph. In developing the Convergence game, which she built based on the World of Balance codebase, Cotter sought to use gamification to help fit ATN models to real ecosystems, and secondarily to evaluate the usefulness of the machine-learning-based predictions presented to the players as hints. Regarding directions future work, Cotter suggests improvements to the game’s user interface, including providing more visual information based on the ATN model and more actively displaying hints, incorporating species link parameters into the analysis, incorporating a local implementation of the ATN engine to mitigate latency issues, and incorporating empirical data for the target graphs, rather than relying on simulated target ecosystems.

Chapter 4

Experimental Methods, Implementation, and Analysis

In seeking to optimize ATN models through machine learning and gamification, we have developed novel algorithms, implemented software for simulation and analysis, explored how ATN models respond to changes in various parameters, made improvements to the World of Balance game software, developed understanding of various aspects of ATN model behavior, and applied machine learning to the data to various ends. This chapter describes that work in conceptual sequence, grouped by activity, presenting the background, methods, and results of each activity together.

Food webs are the ecological foundation of this work and are required input for ATN model simulations and for the World of Balance games. Therefore, we begin by describing algorithms for generating food webs for analysis using graph sampling to draw species from the larger Serengeti food web used in World of Balance (Section 4.1).

Only a subset of the ATN parameters are controlled by players and provided as data in our machine learning experiments. However, other parameters are in need of of exploration and require default values when they are held fixed. In Section 4.7, we explore the effects of the functional response control parameter q. In Section 4.6, we explore a parameter that enables an alternative model of producer carrying capacity.

In order to evaluate simulated ecosystems and give players feedback about their in-game ecosystems, we need ways of measuring and reporting ecosystem health. Section 4.2 describes existing and newly developed ecosystem health measures that are calculated from the biomass data produced by ATN simulations.

The Convergence game [15] within World of Balance applies gamification to the problem of fitting ATN model parameters. While this gamification approach shows promise, it remains difficult for players to navigate the parameter space. In Section 4.3, we describe a way of identifying the more promising regions of the parameter space and providing this information as guidance to players.

One of the overarching goals of this work is learning how to parameterize ATN models representing stable ecosystems. We find that ATN model simulations can reach steady states after which no further extinctions occur, a key observation that informed subsequent experiments. Section 4.4 provides the theoretical foundation and hypotheses about the different kinds of steady states that can occur. Section 4.5 describes how they are detected during simulation. We also find that simulations can reach persistent chaotic states that depend on the value of the q parameter, as described in 4.8.

The Convergence game's library of simulated ecosystems requires expansion to gain more information from player interaction. The biomass data for these ecosystems should be both sustaining and visually interpretable. In Section 4.9, we describe a method, based on steady-state detection, of automatically generating such ecosystems when a predetermined food web size and composition is not required.

Section 4.10 describes our most direct approach to using machine learning to search the parameter space for sustaining ecosystems, synthesizing the insights described in the previous sections. Finally, Section 4.11 evaluates a way to use this parameter space search process to help generate Convergence ecosystem data.

4.1 Graph sampling for food webs

4.1.1 Food webs as graphs

A food web can be represented as a directed graph. Each node in this graph typically represents a species, but can also represent another grouping of organisms, such an ecological guild or a general type of plant resource (e.g. seeds or fruit). Each edge represents a predator-prey relationship. The direction of the edges is from prey to predator, representing the direction of energy flow. The graph may contain self-loops (edges from a node back to itself), representing cannibalism. It is also possible for the graph to contain longer cycles with multiple nodes.

Figure 4.1 illustrates a food web as a directed graph. The node *Grass and Herbs*, in

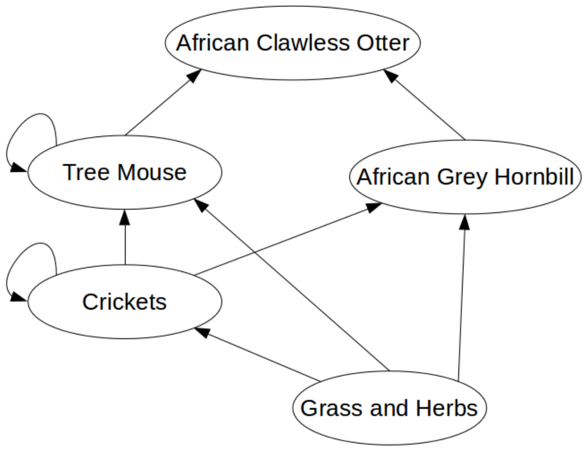


Figure 4.1: Food web consisting of a producer and four consumer species inhabiting the Serengeti ecosystem

graph theory terms, is a *source node*: a node with no in-edges. In ecological terms, it is a *producer:* it produces energy that enters the food web via photosynthesis. (If it represented a single species of grass, it could also be called a *basal species*). The other nodes are called *consumers:* they consume the energy that is provided by their prey. *Tree Mouse* and *Crickets* exhibit cannibalism, represented by self-loops.

We define a *food chain* in graph theory terms as a directed simple path from a source node to another node. In graph theory, a *simple path* is defined as a path with no repeated nodes. For example, in Figure 4.1, *Grass and Herbs* → *Crickets* → *African Grey Hornbill* is a food chain, as is *Grass and Herbs* → *Crickets* → *African Grey Hornbill* → *African Clawless Otter*. Just as there can be multiple simple paths from a source node to another node, there can be multiple food chains from a producer to a consumer, and a node can be part of multiple food chains.

4.1.2 Graph sampling and "subwebs"

The full Serengeti food web used in the World of Balance game consists of 87 different species and plant resources. For purposes of analysis, as well as for playability of the Convergence Game, we needed to generate smaller food webs based on this large set of species. Since a food web is a directed graph, what we needed were directed subgraphs.

The process of generating subgraphs that are somehow representative of a larger graph is called *graph sampling*. The sampled subgraphs must meet certain criteria, which depend on the problem domain. We call the subgraphs in our problem domain *food subwebs* or simply *subwebs*. There are three basic criteria for a directed graph to be a valid representation of a food web or a subweb as we define them.[1]

1. It must have exactly one connected component.

2. For every non-source node u, there must exist a simple path from a source node to u.

Criteria 1 means that the graph represents a single food web, not multiple food webs. Criteria 2 means that there are no incomplete food chains; that is, every species either has prey, or is a producer. Otherwise, that species will certainly go extinct.

A food subweb sampled from a larger food web must meet these criteria. Additionally, as we define it here, a subweb must contain all possible trophic links from the food web involving the species in the subweb. More formally:

[1] There exist more complex definitions of various types of food webs in graph theory terms; see the niche food webs and interval graphs of Cohen (1978) [16] for example.

3. If the subgraph contains nodes u and v, and if a directed edge (u, v) exists in the full graph, in the subgraph must also contain (u, v). Furthermore, if u and v are in the subgraph and the full graph contains a directed edge (v, u), then (v, u) must also be present in the subgraph. This means that if a pair of species in the full food web has a predator-prey relationship, and both species are included in the subweb, then that relationship must be preserved in the subweb.

This also applies to cannibal species: the formal definition holds if $u = v$.

4.1.3 Random connected induced subgraphs

Clearly, sampling the full graph by choosing nodes and edges at random is not guaranteed to result in a subgraph that meets the above criteria. As a first step, one might use a random sampling algorithm that results in one connected component. Algorithm 1 builds a subgraph starting from a randomly selected node in the full graph G. It iteratively grows outward by adding randomly selected neighbors of the current subgraph. The final subgraph S is *induced* in G by the set of nodes selected in this way, which means that for every ordered pair of vertices (u, v) in S, if the edge (u, v) is in G, then it is also in S. Thus, S satisfies criteria 3. Growing the graph outward also ensures that the subgraph is fully connected. Thus, S satisfies criteria 1.

Algorithm 1 Induce a random connected subgraph. G is the full graph, and N is the desired subgraph size. Assumes G is connected and $N \leq |G.nodes|$.

```
procedure RandomConnectedInducedSubgraph($G$, $N$)
    $S \leftarrow (\emptyset, \emptyset)$
    $S.nodes \leftarrow$ random node in $G$
    while $|S.nodes| < N$ do
        $candidates \leftarrow \emptyset$
        for all $n \in S.nodes$ do
            $candidates = candidates \cup (n.neighbors - S.nodes)$
        $S.nodes \leftarrow S.nodes \cup \{n\}$
        $S.nodes \leftarrow S.nodes \cup$ (random node from $candidates$)
    for all $(u, v) \in G.edges$ do
        if $u \in S.nodes$ and $v \in S.nodes$ then
            $S.edges \leftarrow S.edges \cup (u, v)$
    return $S$
```

4.1.4 Random successor subgraphs

Algorithm 1 does not always produce subgraphs that meet criteria 2, because there is nothing to ensure that it includes any source nodes from the full graph. Algorithm 2 is an alternative that solves this problem. It begins by selecting a specified number M of source nodes at random. On each iteration, it randomly selects a node which is a successor to a node currently in the subgraph. Like algorithm 1, it adds edges at the end by inducing the final subgraph using the selected nodes. This ensures that criteria 3 is met.

Unfortunately, the resulting subgraph is not guaranteed to meet criteria 1 if $M > 1$. It selects M source nodes at random, and builds connected components starting from each of these source nodes. These M components may or may not happen to connect in the end. Theoretically, for some values of N and the minimum path length between source nodes, it is impossible to produce a subgraph with exactly one connected component. In practice, with the 87-species Serengeti food web, this problem can be solved simply by re-running the algorithm until it produces a one-component subgraph.

It is possible for the algorithm to fail if, during some iteration, no candidate successor nodes are found. Again, this can be solved by re-running the algorithm until it succeeds. In practice, with the Serengeti food web, this does not happen often.

Algorithm 2 Induce a random successor subgraph. G is the full graph, N is the desired subgraph size, and M is the number of source nodes to include.

```
procedure RandomSuccessorSubgraph($G$, $N$, $M$)
    $S \leftarrow (\emptyset, \emptyset)$
    $S.nodes \leftarrow M$ random source nodes from $G$
    while $|S.nodes| < N$ do
        $candidates \leftarrow \emptyset$
        for all $n \in S.nodes$ do
            $candidates \leftarrow candidates \cup (n.successors - S.nodes)$
        if $candidates = \emptyset$ then
            return failure
        $S.nodes \leftarrow S.nodes \cup$ (random node from $candidates$)
    for all $(u, v) \in G.edges$ do
        if $u \in S.nodes$ and $v \in S.nodes$ then
            $S.edges \leftarrow S.edges \cup (u, v)$
    return $S$
```

4.1.5 Depth-controlled random subgraphs

In addition to the three minimal criteria described above, a subweb should be somehow representative of the full food web. Algorithm 2 can produce subwebs with many species that feed on plants but lack predators, which is not representative of the 87-species Serengeti food web, and can result in unstable population dynamics. Algorithm 3 addresses this issue. In each iteration, before selecting a random successor to add to the graph, it first tries to add a successor of a node v for which the following are true:

1. In the full graph G, v has an in-edge from a source node u which is present in the subgraph S.
2. There is no directed edge (v, w) in G for any node w in S.

In the context of food webs, v represents a species that eats plants but has no predators in S. The algorithm identifies these "predator-free plant eaters," then adds the predators of each one to a list of candidates to add to S. If no candidate predators were identified this way, the iteration proceeds similarly to algorithm 2 by adding all successors of the nodes in S to the list of candidates. It then selects a random node from the candidate list and adds it to S.

In this algorithm, *candidates* starts out as a list instead of a set – that is, it allows duplicates. If a candidate node is a predator of multiple "predator-free plant eaters", then it will be added to the list multiple times, thereby increasing the probability that it will be selected.

This algorithm is not guaranteed to produce a subgraph that has no "predator-free plant eaters". One reason is that *S* may reach size *N* before their predators can be added to the graph. Another is that some plant-eaters, such as the African elephant and the hippopotamus, have no predators in the 87-species Serengeti food web.

By virtue of including more predators of plant-eating species, algorithm 3 can result in more stable subwebs. The populations of plant-eating species can grow exponentially if there are no predators to control them, resulting in a cascade of extinctions later in the simulation.

Like algorithm 2, algorithm 3 may produce a subgraph with multiple isolated components. However, the single-component subgraphs it produces meet all three criteria, are more representative of the full food web, and are more likely to be stable.

Algorithm 3 Induce a depth-controlled random successor subgraph. G is the full graph, N is the desired subgraph size, and M is the number of source nodes to include.

```
procedure DepthControlledRandomSuccessorSubgraph($G$, $N$, $M$)
    $S \leftarrow (\emptyset, \emptyset)$
    $producers \leftarrow M$ random source nodes from $G$
    $S.nodes \leftarrow producers$
    while $|S.nodes| < N$ do
        $candidates \leftarrow$ empty list
        for all $n \in S.nodes : (n.predecessors \cap producers \neq \emptyset)$ do
            if $(n.successors - n) \cap S.nodes = \emptyset$ then
                Append $(n.successors - S.nodes)$ to $candidates$
        if $candidates$ is empty then
            $candidates \leftarrow \emptyset$
            for all $n \in S.nodes$ do
                $candidates \leftarrow candidates \cup (n.successors - S.nodes)$
            if $candidates = \emptyset$ then
                return failure
        $S.nodes \leftarrow S.nodes \cup$ (random node from $candidates$)
    for all $(u, v) \in G.edges$ do
        if $u \in S.nodes$ and $v \in S.nodes$ then
            $S.edges \leftarrow S.edges \cup (u, v)$
    return $S$
```

4.1.6 Results

Because of its ability to produce subgraphs that are valid food webs including high-trophic-level predators, we chose algorithm 3, DepthControlledRandomSuccessorSubgraph, to generate the subwebs used in most of the work described in the following sections. Figure 4.2 shows several subwebs generated by the algorithm.

Table 4.1 summarizes the performance of our implementation of the algorithm for the values of N and M used for the examples in Figure 4.2, and also for larger subwebs of size 15 and 20. It reports the average time to produce a valid subweb and the average number of retries required. The numbers are averaged across 10,000 executions for each subweb size, where each execution includes any retries required to obtain a valid subweb.

We implemented the algorithm in Python using the NetworkX graph library, version 1.11. As of that version, NetworkX itself is a pure Python library. This implementation performs acceptably well for generating the number of and size of subwebs required in our work, despite being implemented in a high-level language.

Table 4.1 shows that the number of retries required is related to the subweb size in an interesting way. For a given value of M (the number of source nodes), increasing N (the subweb size) decreases the number of retries required. This is likely due to the fact that adding nodes to the graph increases the likelihood of connecting the M initial components.

N	M	Avg. runtime (ms)	Avg. retries
5	1	0.298	0.205
6	1	0.348	0.202
7	2	0.910	1.563
8	2	0.929	1.276
9	3	1.999	3.522
10	3	1.998	2.932
15	3	2.284	1.655
20	4	5.044	2.987

Table 4.1: Performance of DepthControlledRandomSuccessorSubgraph implementation. N is the subweb size, and M is the number of source nodes. Run times and retry counts were averaged over 10,000 calls to the function, each of which returned a valid subweb. The timings were run on a 2015 MacBook Air with a 2.2 GHz Intel Core i7 processor.

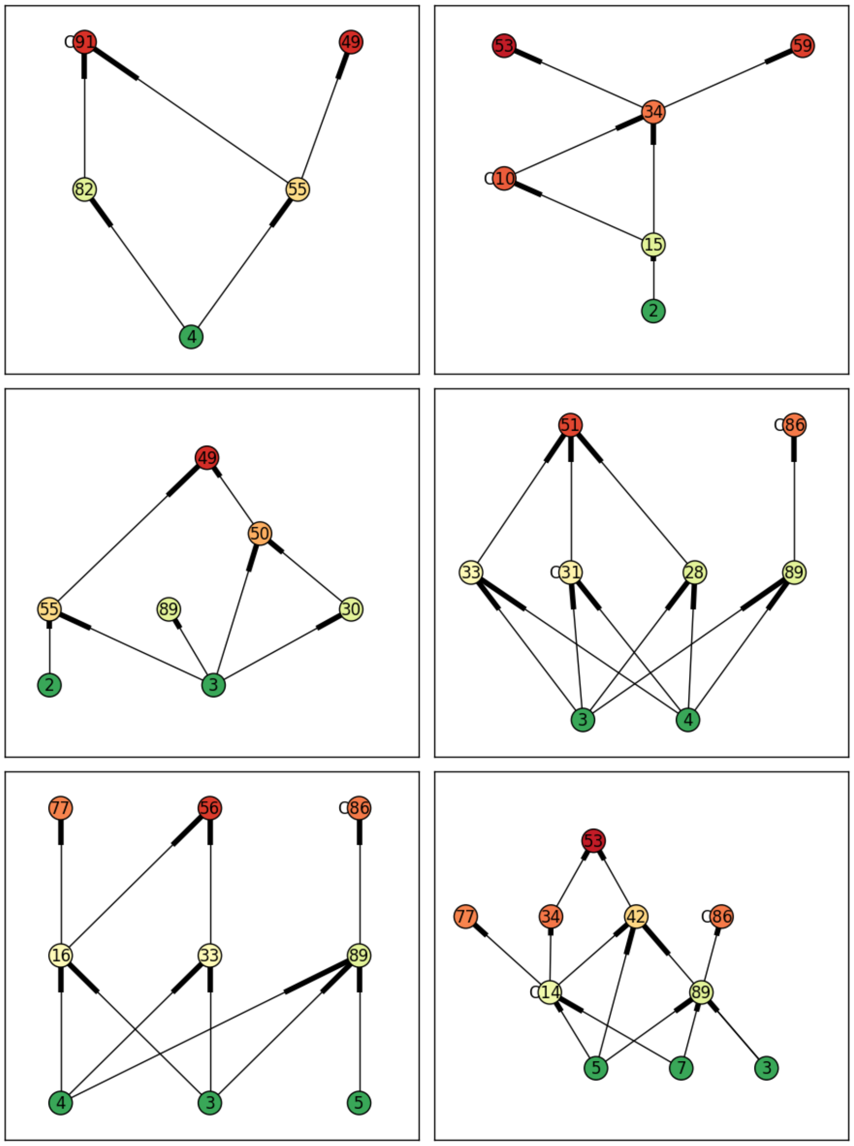


Figure 4.2: DepthControlledRandomSuccessorSubgraph examples. Node colors represent trophic levels within the larger Serengeti food web, increasing from green (1) to red (approx. 3.5). Edge direction is indicated by the thick portions of the lines, and a small “C” to the left of of a node represents cannibalism.

4.2 Measuring the health of simulated ecosystems

One of our main goals is to learn how to generate healthy ecosystems. To this end, we wish to provide players of World of Balance with information about the health of their ecosystems. Also, we wish to identify target variables to optimize and to predict with machine learning models. To enable this, we need well-defined measures of ecosystem health that can be computed from the food web structure and the output data from ATN model simulations.

In this section, we explore two measures of ecosystem health (the *Environment Score* and the *Revised Environment Score*) that we can use for these purposes.

4.2.1 Oscillating patterns in biomass

Many simulated ecosystems exhibit oscillating patterns in species biomass, as can be seen in Figure 4.3. Based on personal communications with Neo Martinez, who has authored many papers based on ATN models, and ecologist S. Jonathan Stern, these kinds of patterns are normal and reflect the population cycles found in nature. The size of a predator population tends to follow the size the population of its prey. As the prey population increases, the predators have more to eat, leading to an increase in predator population. The growing predator population consumes the prey at an increasing rate, until that rate exceeds rate at which the prey can reproduce. This leads to a decline in the prey population, leaving the predators with less to eat, which in turn leads to a decline in the predator population. With fewer predators to limit the prey population, the prey population begins to increase again, and the cycle repeats.

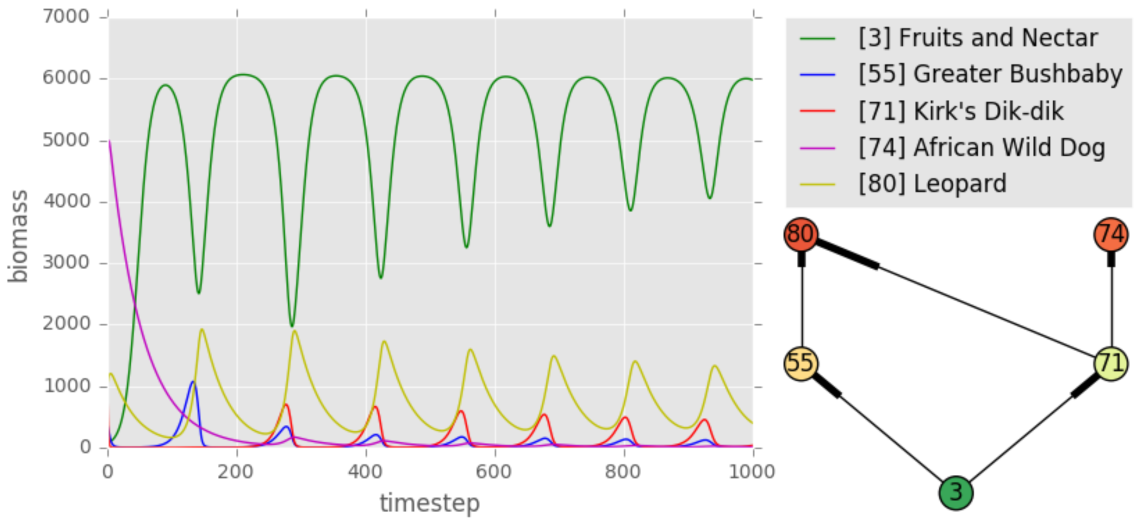


Figure 4.3: Example of simulation with oscillating biomass

Kendall et al. (1998) [17] measure the prevalence of these cyclic patterns in the populations of 220 animal species from Global Population Dynamics Database. They find that about 29% of populations are cyclic, and that 70% of mammal and fish species have at least one cyclic population worldwide. We find that the amplitudes of the oscillations in our data are consistent with the measurements of Kendall et al.

The ecosystem health measures we describe in this section tend to follow these oscillating patterns when they occur. For some purposes, we need to control for the oscillations to get a higher-level picture of ecosystem health. For the World of Balance lobby, we apply a moving average smoothing function to the *Revised Environment Score*.

When using the environment score for machine learning classification in Section 4.3, we calculate a linear time trend of the score over the duration of each simulation.

4.2.2 The original World of Balance Environment Score

The World of Balance *lobby* is the central hub of the game, and includes important gameplay elements of its own. Players purchase tiles, or regions located on a shared world map, and build ecosystems in these tiles by purchasing species. The populations of species in player ecosystems evolve over time based on ATN simulations. Players are shown a score, called the *environment score*, that represents the current health of their ecosystems.

Until recently, the formula to calculate the environment score for a player's ecosystem was as follows:

$$EnvironmentScore = \left[\left[5\log_2 \left(\sum_{i=1}^{N} b_i \left(\frac{B_i}{b_i} \right)^{T_i} \right) \right]^2 + N^2 \right] \tag{4.1}$$

where N is the number of species in the food web, b_i is the per-unit (individual) biomass of species i, B_i is the current total biomass of species i, and T_i is the trophic level of species i. Square brackets indicate rounding to the nearest integer.

This score formula is intended to reward an ecosystem including many species (diversity) as well as the presence of high trophic level species (top predators, which are recognized as indicators of ecosystem health). By raising the number of individuals $\frac{B_i}{b_i}$ of a species to the power of the trophic level of the species, it also rewards a large number of individuals.

4.2.3 Revised Environment Score

There are some potential disadvantages to the original environment score formula, because it can assign far more weight, by orders of magnitude, to some species than others. Specifically, a species with a very small body size and/or high trophic level can dwarf the score contributions of the other species. Another potential issue is that body sizes in the World of Balance database were determined by estimation or educated guesses, rather than taken from published data; therefore, their significant influence on the score is problematic.

We propose a revised environment score that avoids these issues while meeting the following objectives:

- Rewards high biomass levels
- Rewards presence of high trophic level species
- Rewards diversity
- Is significantly influenced by the biomass of each species
- Is interpretable by the player

The alternative score has two components: a biomass score and a diversity score.

Trophic-level-weighted biomass score

To reward high biomass and high trophic levels, the total biomass weighted by trophic level is calculated as follows:

$$BiomassScore = \sum_{i=1}^{N} T_i B_i \tag{4.2}$$

Diversity score

To reward species diversity, the Shannon index is incorporated into the score. The Shannon index is a standard measure of ecological diversity, originally developed by Claude E. Shannon (1948) as a measure of entropy in text strings [18]. For the score, we use a version based on species biomass instead of population density:

$$Shannon = -\sum_{i=1}^{N} p_i \log_2 p_i \tag{4.3}$$

where p_i is the biomass of species i as a proportion of the total biomass in the ecosystem:

$$p_i = \frac{B_i}{\sum_{j=1}^{N} B_j} \tag{4.4}$$

This Shannon index ranges from 0 to $\log_2 N$. It achieves its maximum value for N species when their biomasses are in equal proportion. Thus, it increases both with species count and with evenness of species proportions. However, it does not increase with the total biomass in the ecosystem as a whole. So, in order to map it to a range comparable to the biomass score, we multiply it by the weighted total biomass to calculate a *diversity score*:

$$DiversityScore = BiomassScore \times Shannon \tag{4.5}$$

Total score

The total Revised Environment Score is the sum of the biomass score and the diversity score, or equivalently:

$$RevisedEnvironmentScore = BiomassScore \times (1 + Shannon) \tag{4.6}$$

Environment score smoothing

Both the original and proposed environment score formulas follow fluctuations in biomass. It is desirable to smooth out these fluctuations so that players are rewarded for achieving a good environment score, even if species biomasses are fluctuating. We examined several methods for calculating a smoothed environment score, all of which can be parameterized to perform more or less smoothing:

- **Simple moving average**: the average of the raw score over the previous N time steps.

- **Triangle moving average**: a weighted average of the raw score over the previous N time steps, where the weights decrease linearly to zero at the oldest value.
- **Feedback filter**: a first-order feedback filter. Each output value is equal to the current input value multiplied by a coefficient, plus the previous output value times a coefficient. This is a weighted moving average with infinite history, but the weight of each previous value decreases exponentially with time.
- **Decay rate limit**: the score can increase at any rate, but the rate at which the score can decrease is limited to a fixed threshold fraction of the previous value.

Figure 4.4 illustrates the behavior of these methods on a simulated ecosystem, using the revised environment score.

Down-scaling the environment score

We observed that the alternative environment score can produce very large numbers that may be difficult for players to interpret. We considered several ways of scaling down the score. One possibility is to scale it linearly, in which case all that is needed is a suitable multiplier based on typical player-created ecosystems.

A second possibility is to use non-linear scaling by taking some root of the score. This root-based scaling would result in scores changing more quickly when small, and more slowly when large, when compared to the unscaled score or linear scaling. We find that taking the square root effectively controls the range of the score throughout a large range of ecosystem biomass.

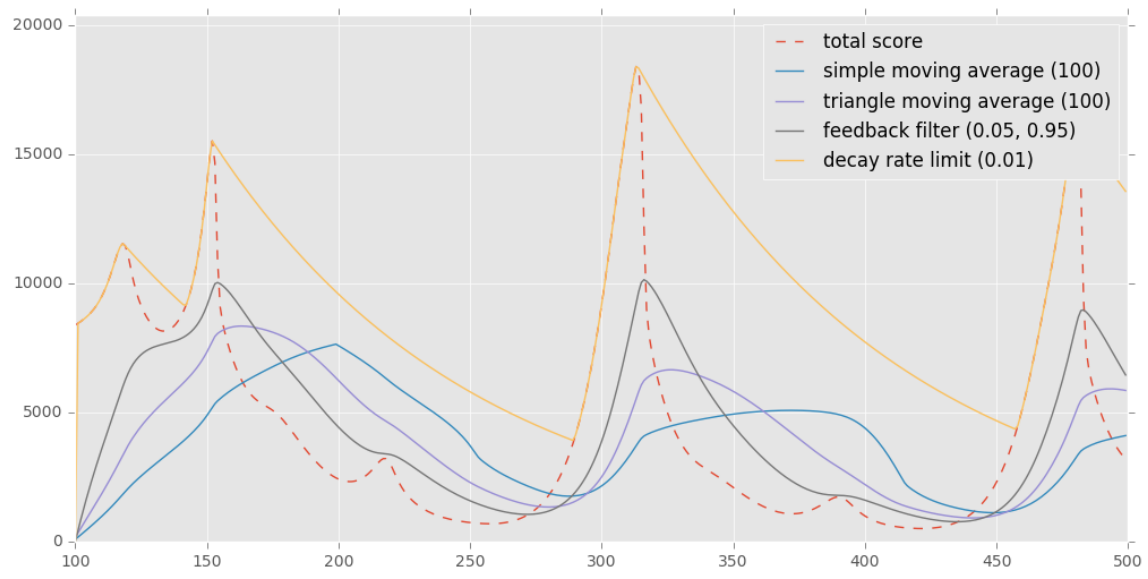


Figure 4.4: Comparison of environment score smoothing methods. “Total score” indicates the raw score, before smoothing.

As of this writing, World of Balance implements the revised environment score with triangle average smoothing and down-scaling using a square root and a scalar multiplier.

4.2.4 Comparison of environment score formulas

Figures 4.5 and 4.6 compare the original environment score with the smoothed and scaled revised environment score. For smoothing, we used the triangle moving average with a window size of 100. For down-scaling, we took the square root of the smoothed score and multiplied by 10. Both figures demonstrate the stair-step effect of rounding for the original score.

In Figure 4.5, the original environment score is influenced primarily by the greater bushbaby, because of its small body size. The leopard, despite having a higher trophic level, contributes relatively little to the original environment score due to its larger body size. In contrast, the revised environment score has a smaller variance and demonstrates a more even set of contributions from all species.

In Figure 4.6, the original environment score is largely determined by the katydids and southern ground hornbill. The katydids have a smaller trophic level than the southern ground hornbill, but have a much smaller body size, leading to a significant contribution to the original score. The effect of the grains and seeds on the original score is negligible, because it has trophic level 1. In contrast, the revised score does take the grains and seeds into account, though to a lesser degree than the animals. The influence of the grains and seeds on the score reflects the fact that producer biomass will positively influence consumer biomass, leading to a healthier ecosystem. All species have an influence on the revised score.

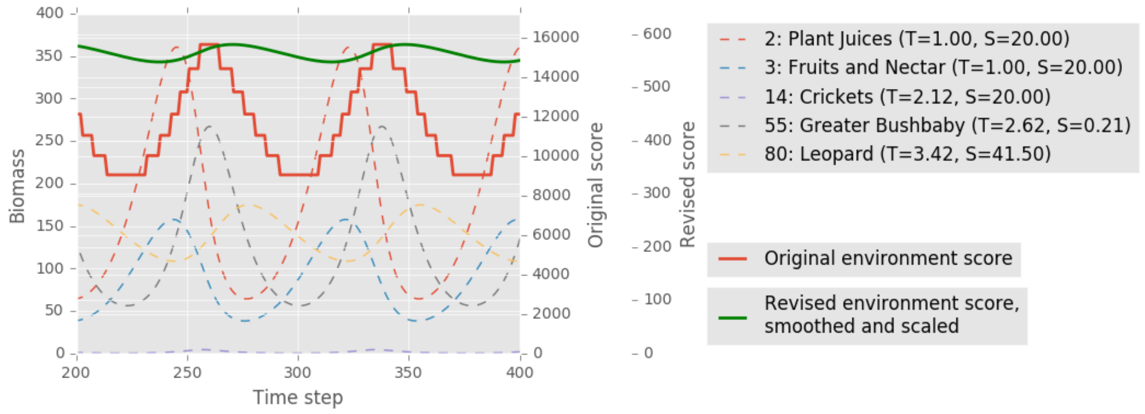


Figure 4.5: First comparison of environment score formulas. In the legend, T indicates trophic level and B indicates body size.

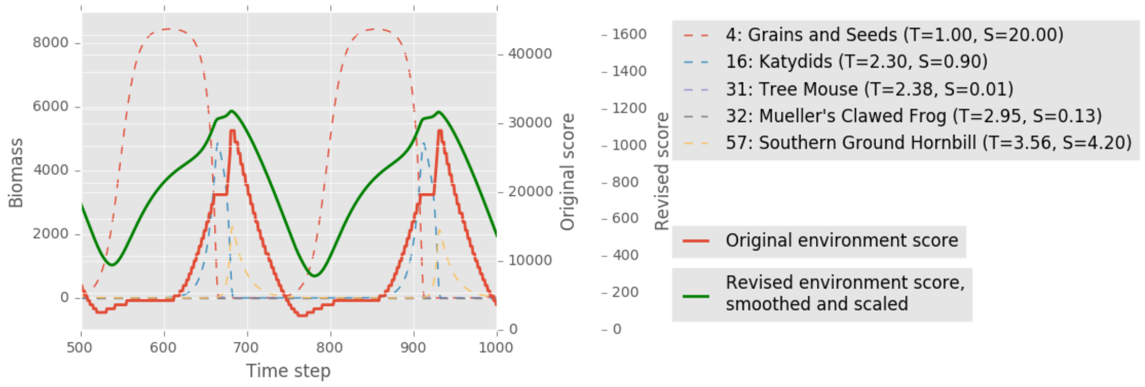


Figure 4.6: Second comparison of environment score formulas. In the legend, T indicates trophic level and B indicates body size.

4.3 Identifying parameter range hints for the Convergence game

To produce the target graphs for Multiplayer Convergence, we sought to improve the data used for the original Convergence game and to determine ranges of parameter values likely to produce desired outcomes and be displayed to players as hints. We propose a 4-step process for each target graph, consisting of (1) species selection, (2) parameter space exploration and simulation, (3) machine-learning classification of simulation results, and (4) derivation of parameter ranges to display as game hints.[2]

4.3.1 Species selection

We use algorithm 3, DepthControlledRandomSuccessorSubgraph, to select a subset of species from the Serengeti food web that form a viable food web including high-trophic-level predators. Call this food web F.

4.3.2 Parameter space exploration and simulation

We explore the parameter space of food web F by running 2,000 simulations with randomized parameter values. This step has three goals: (a) to identify the parameter values that produce the best target graph (b) to select the values for initial biomass that will be

[2] This section is an extended explanation and evaluation of the approach we presented at the 2016 Computational Sustainability Conference (CompSust-2016).

used for subsequent simulations, and (c) to produce a training set for the classification of step (3). We do this in two batches of 1,000 simulations in which parameters are randomly varied according to different rules. The first batch addresses goals (a) and (b), and the second batch addresses goal (c).

Initial batch: selecting target graph and initial biomass values

In the first batch of 1,000 simulations, we set the parameter values for each simulation F_S as follows:

- The metabolic rate for consumer i, x_i, is drawn from a uniform random distribution between 50% and 150% of its default value. The World of Balance database contains default parameter values for each species, including x_i, based on measurements and established models of metabolic rates.

- The carrying capacity for producer i, K_i, is drawn from a uniform random distribution between 1,000 and 15,000. This is the range of values made available to the player in the Convergence games.

- The initial biomass of each species is drawn from a uniform random distribution between 100 and 5,000. This is an intentionally wide range, but not so wide that it should result in biomass values that cause visibility issues when displayed on the same graph.

- Other parameter values are fixed at their defaults. The player can only manipulate a limited set of parameters in the Convergence games (x_i and K_i).

We run each simulation F_S for 1,000 time steps.

After the simulations complete, for each simulation F_S, we calculate the original World of Balance environment score (see Equation 4.1) as a time series. We then calculate the linear time trend of the score time series. When calculating this trend, we exclude the first 200 time steps (personal communications with Dr. Neo Martinez suggested that the initial 100 time steps or so are often discarded to allow the food web dynamics to settle). We interpret this trend value as a measure of ecosystem health: a greater value means that the health of the ecosystem is improving over time.

We choose the simulation with the greatest environment score trend F_S^* for the target graph. We also select the initial biomass values from F_S^* as the fixed values to use for subsequent simulations. (This is because Convergence players are not able to manipulate initial biomass values.)

Second batch: generating training set

Again, we generate a batch of 1,000 simulations. The parameter values for each simulation F_S in this batch are determined as in the first batch, *except* for the initial biomass values. These are kept fixed at the values of the target graph. This batch is used in training the classifier model in the next step.

4.3.3 Machine-learning classification of simulation results

Using the data from the second batch of simulations above, we train a machine learning classification model to predict the ecosystem health trend based on model parameters.

We first compute the quartiles of the ecosystem health trend values of all 1,000 simulations in the training set.

Each simulation F_S is assigned one of two class labels, or is left unlabeled. A label of "good" indicates that the simulation represents a relatively healthy ecosystem, and a label of "bad" indicates that the simulation represents a relatively unhealthy ecosystem. An unlabeled ecosystem is neither exceptionally healthy nor unhealthy. The label is determined based on the quartile in which the trend value of F_S lies. If it is in the top 25%, it is assigned the "good" label. If it is in the bottom 25%, it is assigned the "bad" label. Otherwise, it is left unlabeled.

This quartile-based labeling strategy was chosen for three reasons. First, we have no ecological basis on which to assign "good" and "bad" labels based on absolute values of the environment score trend – it is purely a relative measure. Second, by using quartiles, we maintain class balance – there are always 250 observations in each class. Third, by excluding the middle 50%, we obtain a wider separation between the classes which should improve classification performance.

We use the labeled data created in this fashion to train a binary classifier and to test the learned classification model. The simulation's variable input parameters (x_i for consumers and K_i for producers) provide the features needed to train the classifier. We employ a decision tree classifier, specifically, Weka's implementation of the C4.5 algorithm [19, 20] to learn an *ecosystem health classification model*.

While classification accuracy is not the focus of this experiment – we seek to identify promising parameter ranges, rather than to use the model to make predictions directly – we cannot derive good parameter ranges from an inaccurate classification model. Therefore, we test the learned model with a test set generated based third batch of 1,000 simulations in which parameter values are drawn from the same distributions as in the training set, and whose class labels are assigned based on the same quartiles computed from the training set. In Section 4.3.5, we present classifier test results for three food webs.

4.3.4 Derivation of parameter ranges to display as game hints

We use the structure of the trained decision tree classifier models to derive and evaluate parameter ranges to display as hints in the game. Each node in a binary decision tree represents a branching criterion on the input attributes that distinguishes the data instances in the left subtree from those in the right subtree. These criteria are chosen by the decision tree learning algorithm to maximize the predictive accuracy of the tree on the training data. For real-valued input attributes such as the ATN parameters, each criterion consists of a parameter name and a threshold value that separates the instances in the two subtrees. Thus, for each each parameter there is a (possibly empty) set of threshold values given by the decision tree nodes which have been chosen to effectively separate “good” from “bad” instances.

We will use the decision tree in Figure 4.7 as an example. (This is the decision tree produced from the 5-species food web we evaluate in Section 4.3.5). There are five parameters represented in the tree: $K3$, $X28$, $X51$, $X73$, and $X86$. $K3$ is only used to split the tree once, so the set of threshold values for $K3$ contains the single value 6003.88. $X28$ splits the tree twice, and the associated threshold values are 0.601404 and 0.455708.

We use these threshold values to partition the full range of each parameter π into sub-ranges. For each parameter sub-range $(\pi_{low}, \pi_{high}]$, we count the number of “good”, “bad”, and unlabeled training simulations whose value for parameter π falls between π_{low} and π_{high}. Using Figure 4.7 again as an example, the threshold values for $X28$ define three sub-ranges: $(-\infty, 0.455708]$, $(0.455708, 0.601404]$, and $(0.601404, \infty)$. Analyzing the training data and counting the number of training simulations within these ranges, we obtain the following counts:

- 95 good, 2 bad, and 44 unlabeled simulations within $(-\infty, 0.455708]$
- 97 good, 11 bad, and 122 unlabeled simulations within $(0.455708, 0.601404]$
- 58 good, 237 bad, and 334 unlabeled simulations within $(0.601404, \infty)$

Based on these counts, we assign a score to each range $(\pi_{low}, \pi_{high}]$ as follows. We estimate the probability that a simulation drawn from within $(\pi_{low}, \pi_{high}]$ is a “good” simulation – call this $P(good)$ – as the number of “good” simulations within the range divided by the total number of simulations in the range. In our example for $X28$, for the first range, $P(good) = 95/(95 + 2 + 44) \approx 0.6738$. We estimate $P(bad)$ in the same way: in our example, for the first range, $P(bad) = 2/(95 + 2 + 44) \approx 0.0142$. The score for the range is calculated as $P(good) - P(bad)$. In our example, the score is approximately $0.6738 - 0.0142 = 0.6596$. A positive score represents a promising range – one more likely to result in "good" simulations than "bad" – while a negative score represents a non-promising range.

Figure 4.8 illustrates the parameter ranges derived from this example decision tree and the associated training set. The plot titled "X28 – simulation outcomes" shows the counts of simulations within each range. The plot to the right of that, "X28 – parameter range scores", shows the score calculated as $P(good) - P(bad)$. The leftmost bar in the latter plot shows our calculated score of 0.6596.

We select all ranges with a positive score – the promising ranges – to display to Convergence players as hints.

```
X28 <= 0.601404
|   X51 <= 0.14091
|   |   X28 <= 0.455708: good (11.0)
|   |   X28 > 0.455708: bad (10.0/1.0)
|   X51 > 0.14091
|   |   X73 <= 0.079958
|   |   |   X86 <= 0.069814: good (22.0)
|   |   |   X86 > 0.069814: bad (5.0/1.0)
|   |   X73 > 0.079958: good (157.0)
X28 > 0.601404
|   X51 <= 0.18639: bad (194.0)
|   X51 > 0.18639
|   |   K3 <= 6003.88: good (56.0/1.0)
|   |   K3 > 6003.88
|   |   |   X73 <= 0.145747: bad (41.0)
|   |   |   X73 > 0.145747: good (4.0/1.0)
```

Figure 4.7: A Weka J48 decision tree representation, as displayed by the Weka software. The root node is on the left, and the leaves are on the right. Each line of text represents an edge in the tree. The letter-number token toward the left of each line represents a feature in the training set (in our case, the letter is an ATN model parameter, and the number is a species identifier). The relational operators and decimal values represent the split criteria. "Good" and "bad" represent the predicted class labels at the leaves. The numbers inside the parentheses show the number of training instances reaching each leaf, and the number of misclassified instances, if any, is shown after the slash.

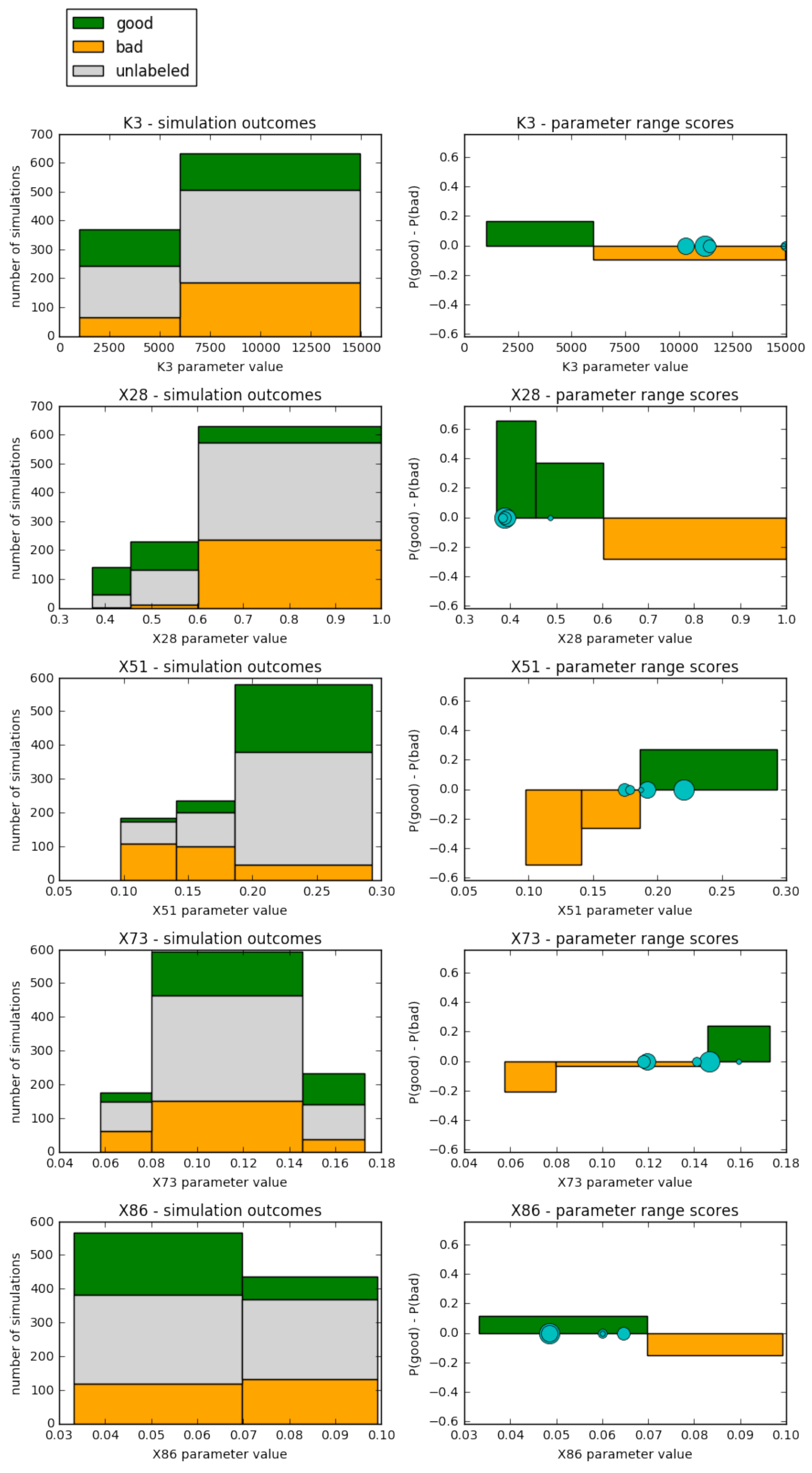


Figure 4.8: Parameter range evaluation based on the decision tree shown in figure 4.7. Left: number of "good," "bad," and unlabeled simulations within parameter ranges derived from the tree. Right: score for each parameter range. Green boxes show promising ranges. Orange boxes show non-promising ranges. Parameter value boundaries correspond to the threshold values of the decision tree nodes. Blue circles indicate the parameter values for the top 5 simulations. Their sizes descend in rank order (largest is best).

4.3.5 Evaluation

To evaluate our approach for deriving parameter range hints we followed the 4-step procedure described above, using three randomly generated food webs consisting of 5, 10, and 15 species. We evaluated the learned classification models on test data, derived the parameter range hints, and finally evaluated the potential effectiveness of these hints based on simulated data intended to mimic player behavior.

Performance of the learned classification models

We evaluated the performance of the decision tree classifiers on test data, as described in Section 4.3.3. Table 4.2 shows the results. For all three food webs, the classifiers are able to very accurately distinguish "good" from "bad" simulations based on our labeling strategy.

Food web	Class	Precision	Recall	F1 score
5 species	good	0.951	0.951	0.951
	bad	0.959	0.959	0.959
10 species	good	1.000	0.996	0.998
	bad	0.996	1.000	0.998
15 species	good	0.996	1.000	0.998
	bad	1.000	0.996	0.998

Table 4.2: Evaluation of ecosystem health classification model

Effectiveness of parameter range hints

A user study to evaluate this approach applied to Multiplayer Convergence was originally proposed for the fall of 2016. The players in the study were to be divided into two groups. The first group would be given the parameter range hints derived using our approach, while the second group would play without the hints. The performance of the two groups would be compared.

This user study did not take place as planned due to software implementation issues. However, we performed a preliminary evaluation of the parameter range hints in which we simulated player attempts based on the expected behavior of the two groups. To do this, we treat simulated players as fairly naïve in their choice of parameter values: players in the control group (who are not shown the hints) select parameter values randomly from within the entire valid range of each parameter, and players in the test group (who are shown the hints) select parameter values randomly from within the hint ranges. (For the test group, if there are no hints for a parameter, then parameter values are selected randomly from the entire valid parameter range.) In reality, Convergence players are more intelligent and use intuition gained from prior attempts to inform subsequent choices of parameter values, but this simple simulation provides a baseline.

We performed this evaluation independently for each of the three food webs from which we derived parameter ranges above.

To simulate the control group, we generated a batch of 1,000 simulations in which the x_i and K_i parameters were drawn from uniform random distributions across their entire valid range (0 to 1 for x_i, and 1,000 to 15,000 for K_i). We calculated the environment score trend for each simulation.

To simulate the test group, we generated another batch of 1,000 simulations. For these simulations, if a parameter had a hint range, its value was drawn from a uniform random distribution within that range. Otherwise, its value was drawn from the entire valid range for the parameter. Again, we calculated the environment score trend for each of these simulations.

Our analysis compares the environment score trends between the two groups. The null hypotheses is that the environment score trends are no different between the control and test groups. Table 4.3 shows the means and standard deviations of the trend values for the three food webs and two groups. As could be expected, the test group has smaller standard deviations, because its parameter values are constrained to smaller ranges by the hints. For the 5- and 15-species food webs, the test group performs better on average. Interestingly, the control group performs better on the 10-species food web.

To test for a statistically significant difference between the means for the two groups, we use Welch's t-test. Welch's t-test, in contrast to the more common Student's t-test, does not assume that the groups have equal variance (they do not, according to the results above). It does, however, assume that the populations are normally distributed, an assumption we may be violating here. However, given the very high p-values we find, we believe this is not a significant problem. Table 4.4 shows the test statistics and p-values for the three food webs.

Food web	Statistic	Control group	Test group
5 species	mean	-3.054789	5.358788
	std	9.410075	2.236706
10 species	mean	-5.187572	-6.016333
	std	8.226043	2.130220
15 species	mean	-2.250668	-0.975169
	std	5.101375	4.270391

Table 4.3: Comparison of performance of two simulated groups of Convergence players. The control group represents players who did not receive hints. The test group represents players who did receive hints. The mean and standard deviation are calculated on the environment score trend value.

Food web	t-statistic	p-value
5 species	27.508	1.772×10^{-127}
10 species	-3.0842	0.00209
15 species	6.0628	1.6038×10^{-9}

Table 4.4: Welch's t-test results for the data in Table 4.3

For all 3 food webs, based on the p-values shown, we reject the null hypotheses that the environment score trend values are the same between the two groups. The hints do indeed make a difference to player performance in this simulated experiment.

The mixed results showing *decreased* performance on one of the food webs, however, indicate that further modifications and analysis of the approach are required before the hints can be expected to improve actual player performance in the game. One likely issue is that the static ranges for each parameter do not include information about how the parameters interact with each other. As we have described, ATN models represent the complex, highly interdependent nature of ecosystems. Parameter values, therefore, do not have independent effects. Rather than parameter *ranges*, we should be examining promising *regions* of the parameter space. (We pursue this line of thought in Section 4.10). From a gameplay perspective, this would require some kind of dynamic, interactive system for displaying hints that would allow the players to navigate the parameter space with more information than static ranges can provide.

4.4 Steady states

An observation we made of ATN models based our simulations is that they often reach what could be described as steady states. A steady state, as we define it, is a state in which species biomasses remain constant, or are locked in a periodic oscillating pattern, or some combination of the two, and this state continues indefinitely. If a simulation reaches a steady state, no further extinctions will occur.

This is a useful concept for two main reasons. First, it eliminates unnecessary computation time. Continuing a simulation beyond the point at which a steady state is reached provides no new information. With simulations only running until a steady state is reached, we can set a much longer maximum duration for simulations. This allows us to gain more information about simulations with dynamics that take a long time to unfold, without wasting computation time on simulations that are generating redundant information.

Second, it allows us to evaluate simulations knowing exactly how many extinctions will occur given the model parameters. The number of extinctions resulting from a given set of parameters is a useful metric, because it quantifies to what extent a simulated ecosystem is “sustaining” (see Section 4.10.1). It provides us with a target variable to optimize, as we do in Section 4.10.

4.4.1 Hypothesis

Our intial hypothesis about steady states is as follows: If run indefinitely, the model will eventually reach a steady state. That steady state can be one of two types: (1) all species are extinct or (2) some species have nonzero biomass which has reached a steady state.

Type (2) can be further divided into two possible states: (2a) consumers are extinct and producers have reached carrying capacity or (2b) there are both producers and consumers with nonzero biomass that can be confirmed as a steady state. Based on initial observations, a graph of a simulation that has reached a steady state with nonzero biomass can consist either of straight lines, or show a periodic oscillating pattern.

Thus, in this hypothesis, there are three possible outcomes of a simulation in terms of reaching a steady state. The unambiguous, well-defined nature of these steady states provides an attractive target variable for machine learning tasks.

Figure 4.9 demonstrates the initial observations supporting this hypothesis.

4.4.2 Testing for steady states

An important fact that helps verify that the model has reached a steady state is that the model is memoryless. The change in biomass at time t depends only on the biomass at time t: $\mathbf{B_t}' = f(\mathbf{B_t})$, where $\mathbf{B_t}$ is the biomass state vector at time t, and $\mathbf{B_t}'$ is the derivative the biomass state vector. The change in biomass does not depend on what happened before time t. Given a particular biomass state vector at time t, the output that follows will always be the same, assuming the parameters of f are fixed.

Therefore, to test whether the model is in a steady state at time t, even if biomass is not constant, we can search for a matching biomass vector in a previous timestep that matches the vector at time t.

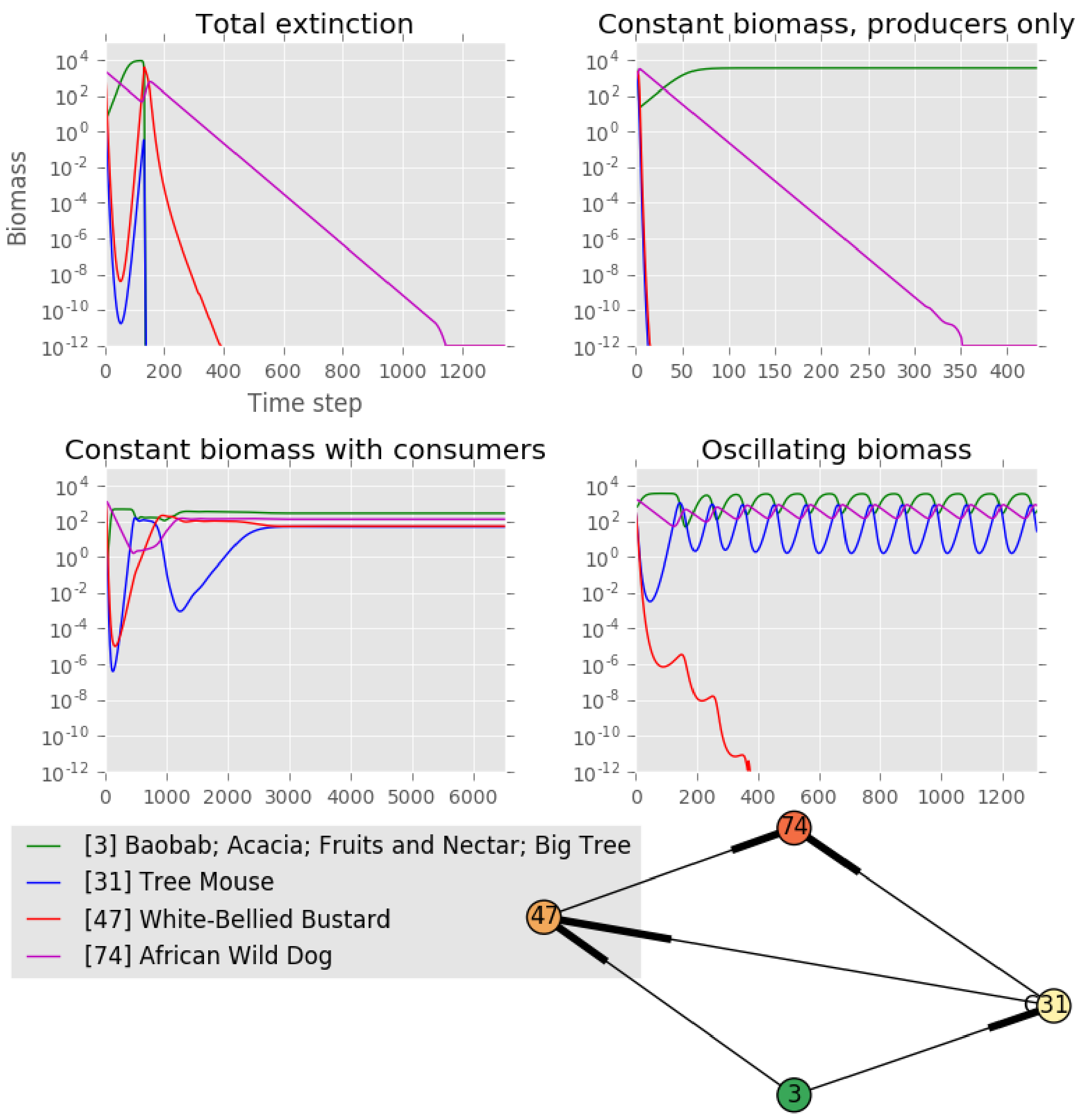


Figure 4.9: Observed steady states for a 4-species food web. The vertical axis has a logarithmic scale. 10^{-12} is the extinction threshold. The biomass curves change in shape near the extinction threshold due the tolerances of the numerical integration underlying the simulation.

4.5 Implementation of steady state detection

We implemented code in ATN Simulator to detect steady states in the model and stop the simulation when a steady state is detected. The steady states it can detect fall into two categories:

1. Biomasses of all species are constant
2. Biomasses of one or more species are oscillating together in a periodic pattern, while the biomasses of any other species are constant

4.5.1 Terminology

Simulating an ATN model consists of integrating a system of ordinary differential equations over time – that is, solving an *initial value problem.* In numerical integration, the output at a point in time is called the *state vector.* With ATN model simulations, each term in the state vector represents the biomass of a species in the food web. We call our state vector the *biomass state vector.*

4.5.2 Constant-biomass steady states

Constant-biomass steady states include states in which all species have gone extinct, or all consumers have gone extinct and producers have reached carrying capacity, or both producers and consumers survive and have constant biomass. A constant steady state is detected by continuously monitoring the derivatives of the model over the integration. When all derivatives are very close to zero, a constant steady state is confirmed and the integration is stopped. “Very close to zero” means the following:

$$\left|\frac{B_i'}{B_i}\right| \leq 10^{-10} \tag{4.7}$$

where B_i' is the derivative of the biomass of species i and B_i is the biomass of species i.

4.5.3 Oscillating steady states

Detecting oscillating steady states is more complex. Oscillating steady states are detected based on the observation that if the biomass state vector returns to any state it held previously, then it will return to that state again and again in a periodic pattern. To facilitate the comparison of the current state to a previous state, the simulator takes snapshots of the state vector at exponentially spaced intervals. It then monitors the error between the current state vector and the snapshot. When the current state matches the snapshot within a certain tolerance, and at least one species has oscillating biomass, an oscillating steady state is likely. To provide confirmation before stopping the integration, the snapshot must be matched three times.

The current state is considered to match the snapshot when the relative error for all species is within a 1% tolerance of the snapshot value. That is,

$$\left|\frac{B_i - B_{si}}{B_{si}}\right| \leq 0.01 \tag{4.8}$$

for all species i, where B_i is the biomass of species i and B_{si} is the biomass of species i in the snapshot.

A species is identified as having oscillating biomass if its biomass has had both positive and negative derivatives since the time of the snapshot.

Each snapshot interval is double the length of the previous one. This is because the duration of the period is not known in advance, and in order to detect a period, the current interval must be at least as long as the period.

4.5.4 Implementation

We implemented two steady state detectors – one for constant steady states and one for oscillating steady states – as two classes implementing the EventHandler interface in the ODE package of the Apache Commons Math library.

An EventHandler defines a continuous function of the state vector and current time that changes sign when some event of interest has occurred. This is called a *switching function*. For example, the switching function in the oscillating steady state detector calculates the total error between the current biomass state vector and the snapshot state vector. When this error changes sign – that is, crosses zero – a possible match between the current state and the snapshot has occurred. The integrator then calls a method that returns a value indicating whether the integration should be stopped or should continue. In the oscillating steady state detector, this method checks whether all biomasses in the state vector match the snapshot within a tolerance.

4.5.5 Qualitative evaluation

During the process of refining the steady state detection algorithms and tolerances, we examined thousands of simulations for cases where a visually obvious steady state was not detected (a false negative), or a steady state was incorrectly detected (a false positive). Qualitatively, the current implementation performs very well, though it sometimes errs on the side of continuing simulations beyond the point at which a steady state is visually apparent. This is not a serious problem, and is much better than cutting simulations short before they have reached a steady state.

Figures 4.10–4.12 show examples of simulations that were stopped when a steady state was detected.

4.5.6 Quantitative evaluation

Methods

To quantify the accuracy of the steady state detection, we performed an analysis in which we estimated the number of occurrences of false detection of a steady state for randomized simulations of several different food webs. Based on this false positive (FP) count and the true positive (TP) count among detected steady states, we calculated the precision ($TP/(TP + FP)$) of the steady state detection for each food web.

To avoid inflating our precision with easy true positives, we excluded detected steady states resulting from complete extinction. Once all species have gone extinct, it is clear that their biomass will remain at zero. We included only what we we call *nonzero steady states* – steady states with surviving species – in our analysis.

We justify the use of precision as a metric as follows.

The most important requirement with steady state detection is that it should make as few type I errors, or false positives, as possible. In other words, falsely detecting steady states is undesirable, because it means drawing incorrect conclusions about the stability of a simulated ecosystem. On the other hand, a false negative, or type II error, implies a *lack* of conclusive information about the simulation, which is not as much of a problem.

A high precision indicates that most detected steady states are true steady states. It implies that the number of false positives is relatively small. For that reason, we focus on precision here.

Another reason we focus on precision is that other metrics (such as recall) depend on counting false negatives, which is not possible here without making unreasonable assumptions. False negatives are simulations for which steady state detection failed but which would eventually reach a steady state if run indefinitely. Again, lacking ground-truth labels, we cannot identify false negatives without employing our steady state detection algorithm and assuming it works correctly – clearly not a reasonable assumption, given that this algorithm is exactly what we are testing.

Lacking a set of ground-truth labels indicating true steady states, we used the following heuristic to distinguish between true positives and false positives. We continued running each simulation for 100,000 time steps beyond the point at which a nonzero steady state was detected. If any further extinctions occurred, we flagged the simulation as a false positive (if the simulation had reached a true steady state, then no further extinctions would have occurred). Otherwise, we flagged it as a true positive.

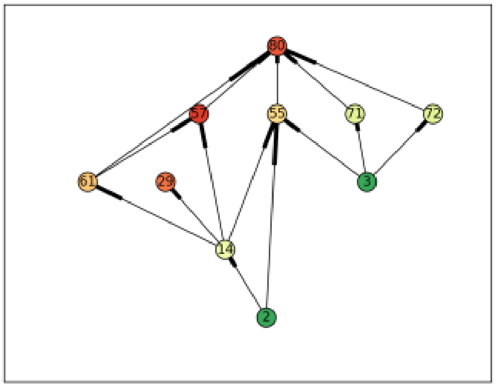
We analyzed three 5-species food webs and two 10-species food webs. We randomized the x and K parameters as well as initial biomass. We held other parameters fixed.

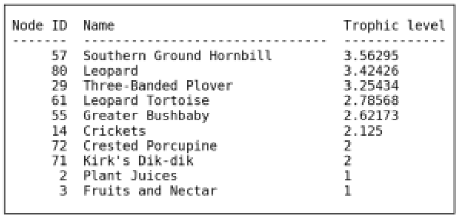
Results

Table 4.5 summarizes the results of the analysis. The total of TP and FP varies, because although we generated 1,000 simulations for each row of the table, only simulations with a detected nonzero steady state are included. The steady state detection has perfect or near-perfect precision for all food webs below.

	TP	FP	Precision
Food web			
2-8-9-26-41	319	0	1.000
3-21-55-80-85	29	1	0.967
3-30-50-69-71	17	0	1.000
2-3-5-8-9-21-22-69-71-94	718	1	0.999
4-7-14-43-47-61-69-74-80-89	836	0	1.000

Table 4.5: Steady state detection evaluation results





Node ID	Name	Trophic level
57	Southern Ground Hornbill	3.56295
80	Leopard	3.42426
29	Three-Banded Plover	3.25434
61	Leopard Tortoise	2.78568
55	Greater Bushbaby	2.62173
14	Crickets	2.125
72	Crested Porcupine	2
71	Kirk's Dik-dik	2
2	Plant Juices	1
3	Fruits and Nectar	1

Figure 4.10: A 10-species food web used in steady state detection evaluation

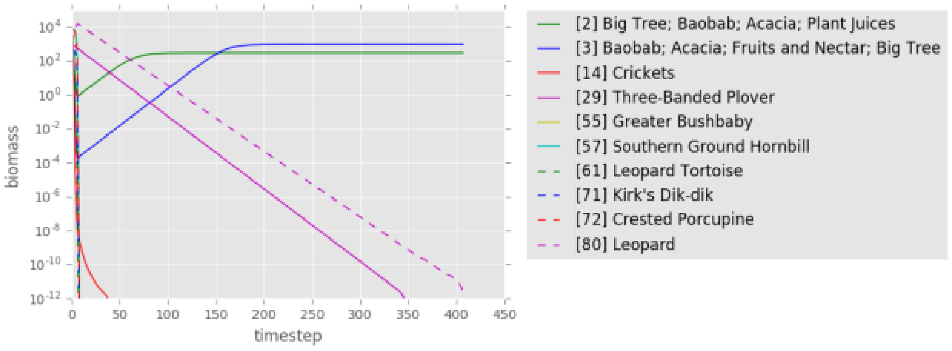


Figure 4.11: A detected constant steady state

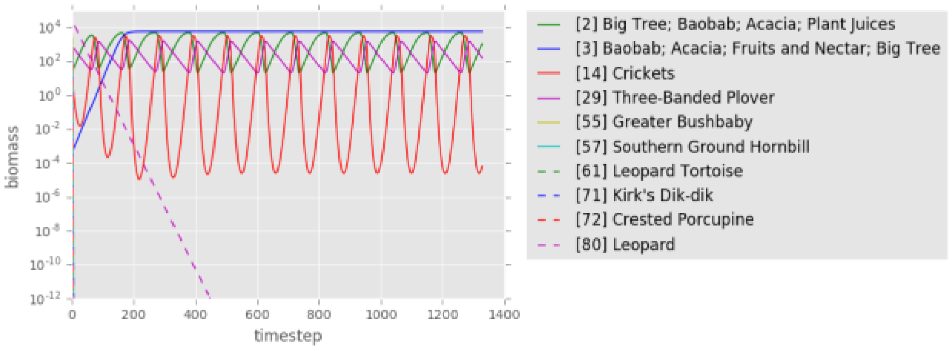


Figure 4.12: A detected oscillating steady state

4.6 System-wide carrying capacity

In this section, we compare two different ways of representing producer carrying capacity in ATN models: one in which producers have independent carrying capacity, and another in which producers share a system-wide carrying capacity. The simulations in World of Balance and Convergence currently use the first type of model. However, the second type is more richly descriptive. Our goal here is to explore the system-wide carrying capacity model as a potential alternative for World of Balance and Convergence.

For our other experiments, we continue to use the independent carrying capacity model, because it is built in to the design of the Convergence game. However, we implemented the system-wide carrying capacity model as an option in the simulation code.

For the independent carrying capacity model, we use the simple logistic growth function for producers defined by Williams et al. (2007) [3]:

$$G_i(B) = 1 - \frac{B_i}{K_i} \tag{2.4 revisited}$$

where B_i is the biomass of producer species i and K_i is the environment's carrying capacity (maximum biomass) for species i. This causes the growth of the producer to slow down and stop as its biomass approaches carrying capacity. When B_i is very small, $G_i(B)$ is very close to 1, meaning that the growth of the producer is effectively unconstrained by the carrying capacity. When B_i is equal to K_i, on the other hand, the second term becomes 1, and so $G_i(B)$ becomes 0: the producer's growth stops because its biomass has reached carrying capacity.

This growth function assumes that the environment has a completely separate and independent carrying capacity for each producer. This is not true in reality; while various species of plants do share areas of the environment without being in full competition for it, there are shared limiting resources (such as nutrients, water, sunlight, and land area). Researchers have proposed various alternative growth functions to account for these factors [3, 6, 7, 14].

The ATN models used by Boit et al. (2012) [6] and Kuparinen et al. (2016) [7] use the following growth function to model a system-wide carrying capacity shared by all producers:

$$G_i(B) = 1 - \frac{\sum_{j \in producers} c_{ij} B_j}{K_s} \tag{2.5 revisited}$$

where K_s is the system-wide carrying capacity and c_{ij} is a competition coefficient.

Boit et al. [6] fix c_{ij} at a value of 1 except for when $i = j$ (in which case, it is set to 1.8 to model higher intra-guild competition). Kuparinen et al. [7] seem to use a fixed value of 1 for c_{ij} in all cases. Accordingly, we fix c_{ij} at 1 for our experiments.

The ATN Simulator software simply sets $K_s = \sum_{i \in producers} K_i$ when the system-wide K option is selected. This is for ease of parameterization and comparison of the two growth functions.

4.6.1 Comparison of the two growth functions

The two growth functions can produce very similar results, or very different results, for different simulations. If the biomass of producers does not approach carrying capacity, the choice of growth function has little effect. On the other hand, if producers do approach carrying capacity, the differences in the behavior of the two growth functions are evident. With a system-wide carrying capacity, producers must compete, which results in coupled dynamics between them.

Figure 4.13 shows a comparison between the simple growth function (plots on the left) and the system-wide K growth function (plots on the right) for several simulations of a 5-species food web.

4.6.2 System-wide carrying capacity and steady state detection

Given the differences in simulation dynamics between the two growth models, we wanted to assess whether there was any difference in the effectiveness of our steady state detection algorithms between the two growth models. To do this, we repeated the steady state detection evaluation described in Section 4.5.6 for the system-wide carrying capacity model.

The prior evaluation used three 5-species food webs with one producer each. In order for the system-wide carrying capacity growth model to be meaningfully different from the individual carrying capacity model, we had to add a second producer to each of these food webs. We used the same two 10-species food webs from the previous evaluation.

Table 4.6 shows the results of this evaluation. The two growth models result in effectively the same precision from the steady state detection algorithms, which remains near-perfect. Interestingly, the system-wide carrying capacity model results in fewer detected nonzero steady states for all food webs, but this difference is not large.

Food web	Growth fn	TP	FP	Precision
2-3-8-9-26-41	indiv. K	800	0	1.000
	system K	739	1	0.999
2-3-21-55-80-85	indiv. K	747	1	0.999
	system K	725	0	1.000
3-4-30-50-69-71	indiv. K	807	0	1.000
	system K	737	2	0.997
2-3-5-8-9-21-22-69-71-94	indiv. K	718	1	0.999
	system K	605	1	0.998
4-7-14-43-47-61-69-74-80-89	indiv. K	836	0	1.000
	system K	830	0	1.000

Table 4.6: Steady state detection evaluation results with system-wide carrying capacity

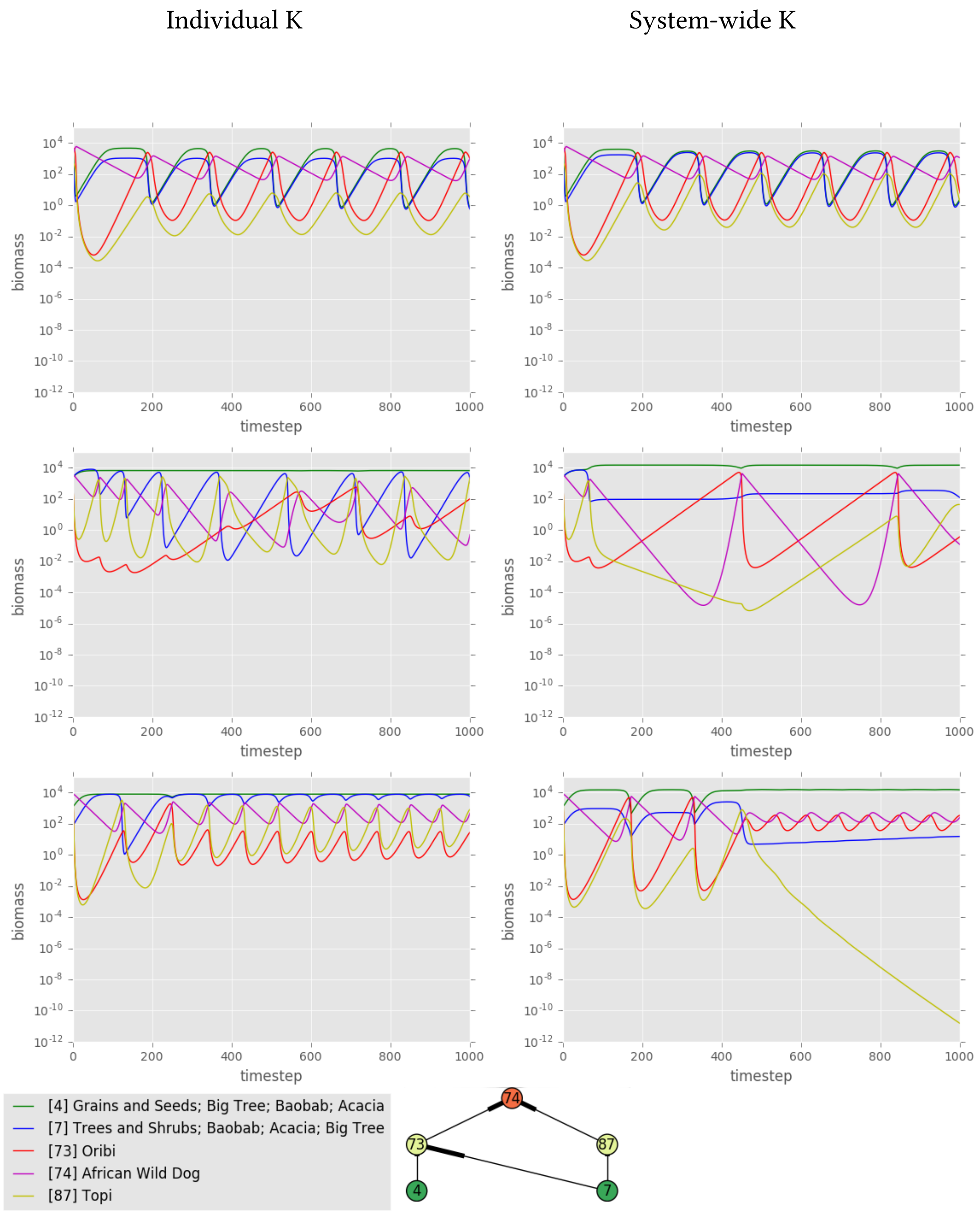


Figure 4.13: Comparison of individual K and system-wide K

4.7 Functional response control parameter q

We use equation 2.6 (shown again below) to compute the functional response F_{ij} of predator species i with respect to prey species j.

$$F_{ij} = \frac{B_j^{1+q_{ij}}}{\sum_{m \in prey} \alpha_{im} B_m^{1+q_{im}} + B_{0ij}^{1+q_{ij}}} \tag{2.6 revisited}$$

q_{ij} is a parameter that controls the shape of the functional response curve, allowing it to vary from a Holling Type II functional response ($q = 0$) to a Holling Type III functional response ($q = 1$) [3, 14]. We call it the *functional response control parameter*. Figure 4.14 shows how the behavior of the model changes as q is varied from 0 to 0.5 in increments of 0.1 for all species in a 5-species food web.

Increasing q even slightly from 0 to 0.1 has a significant stabilizing effect. The functional response equation models how predators consume more of prey species which are more abundant, and less of those which are less abundant. When $q > 0$, this shift toward more abundant prey becomes more dramatic, because the dependence on prey density becomes exponential instead of linear. As a result, fluctuations in biomass become smaller, and less abundant prey are less likely to go extinct.

The functional response control parameter is defined in at least two different ways in the literature. Williams et al. (2007) [3] define the exponents in the functional response equation to be $1 + q$, as they are in our functional response equation. Other researchers [6, 7, 14], define the exponents as q by itself (or h for *Hill exponent*). However, the descriptions of the parameter in these papers are all consistent with an exponent of 1 corresponding to a Type II functional response and an exponent of 2 corresponding to a Type III functional response.

Boit et al. (2012) [6] and Kuparinen et al. (2016) [7] chose a value of 1.2 for the parameter. According to Boit et al., this value "forms a relatively stable …Type II functional response." According to Kuparinen et al., the value creates a functional response "intermediate between the Holling Type-II and Type-III functional responses." Because our exponents are $1 + q$, this corresponds to a value of 0.2 for our q parameter.

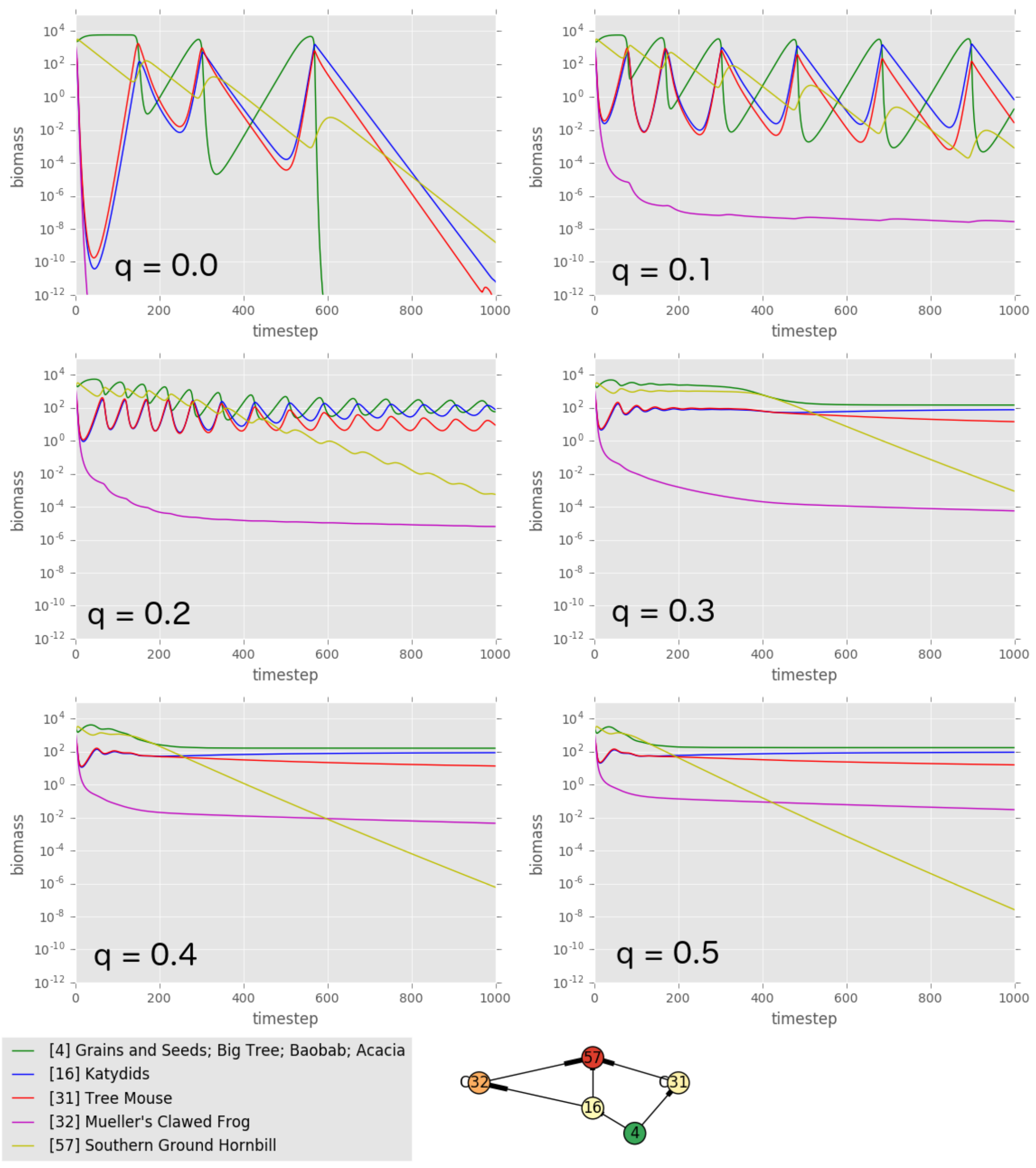


Figure 4.14: Effects of varying the *q* parameter

4.8 Effect of $q > 0$ on steady states

A value of q greater than 0 causes all the terms in the functional response (equation 2.6) to be nonlinear, because it makes their exponents greater than 1. This can result in very interesting, chaotic dynamics. Sometimes this chaos resolves into a detected constant or periodic oscillating steady state, as shown in figure 4.15.

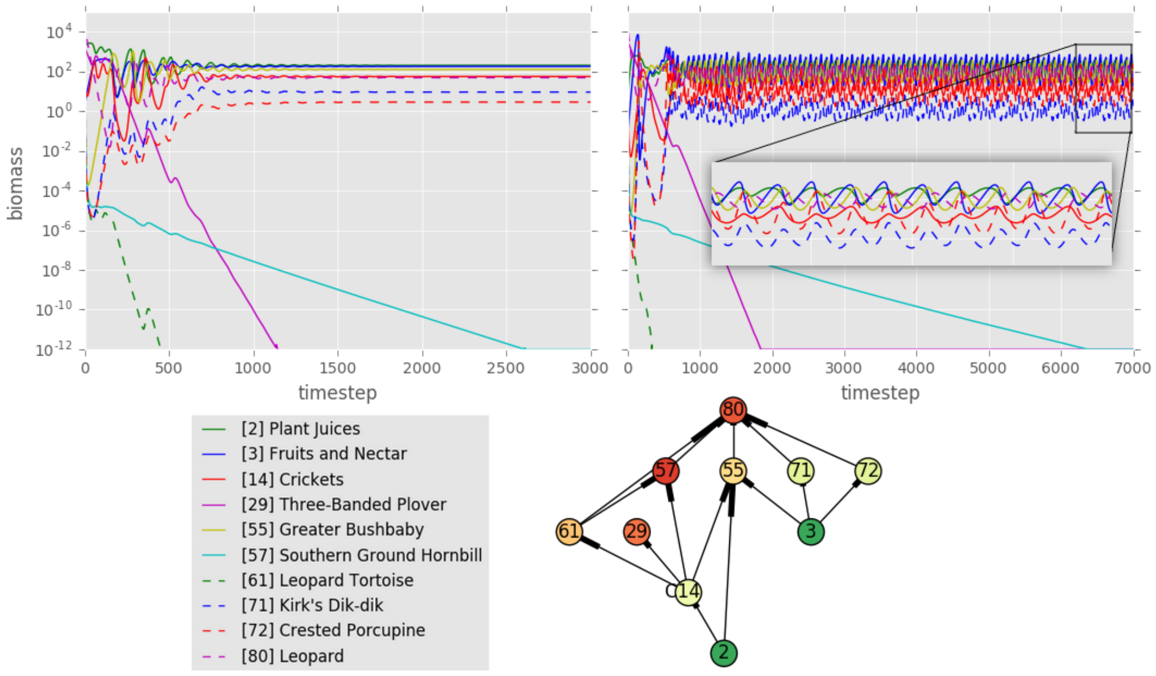


Figure 4.15: Chaotic dynamics resolving to a steady state with $q = 0.2$

Other times, the chaotic dynamics continue seemingly indefinitely, as shown in figure 4.16, and no steady state is detected.

Intuitively, there is an aspect of the dynamics in figure 4.16 that seems stable. It seems to represent some kind of steady state in that there are no apparent trends in biomass. It also seems likely that no further extinctions will occur. However, the dynamics are clearly not constant and apparently not periodic. Situations similar to this result from many different parameter configurations with $q = 0.2$. We do not observe these persistent chaotic dynamics when $q = 0$.

Is there a definable type of "chaotic steady state" that would describe these dynamics, such that the state could be detected and the simulation could be halted with confidence that the state will not change? One possible definition would require average biomasses between successive time segments at some time interval to remain constant. Additionally, variance could be required to remain constant. However, the dynamics above do not satisfy these requirements: even over segments of 10,000 time steps, both the mean biomass and variance in biomass vary significantly, as shown in figure 4.17.

For the purpose of steady-state detection, we consider the term "chaotic steady state" to be the oxymoron it sounds like, and categorize these kinds of results as inconclusive: we cannot be sure how many further extinctions will occur. This partly disproves the hypothesis in section 4.4.1. There are situations in which the dynamics may not resolve into a constant or periodic oscillating steady state. However, steady state detection is still useful, because many simulations still do resolve into steady states with $q > 0$.

4.8.1 Steady state detection performance with $q > 0$

In Section 4.5.6, we presented an evaluation of the implemented steady state detection algorithms. Up to that point in our analysis, we had been using a value of $q = 0$, and the steady state detection algorithms and tolerances were tuned based on simulations using this value. Table 4.7 repeats the evaluation for $q = 0.2$ and compares the performance with $q = 0$. The worst performance is on two of the 5-species food webs where $q = 0.2$. Inspection of the simulation data reveals the likely cause: a lone species declining so slowly to extinction that its derivative was within the steady state detection tolerance. This updated analysis reveals that further tuning of the steady state detection may be required to improve performance with positive values of q.

		TP	FP	Precision
Food web	q			
2-8-9-26-41	0.0	319	0	1.000
	0.2	954	0	1.000
3-21-55-80-85	0.0	29	1	0.967
	0.2	432	86	0.834
3-30-50-69-71	0.0	17	0	1.000
	0.2	929	50	0.949
2-3-5-8-9-21-22-69-71-94	0.0	718	1	0.999
	0.2	725	0	1.000
4-7-14-43-47-61-69-74-80-89	0.0	836	0	1.000
	0.2	850	0	1.000

Table 4.7: Steady state detection evaluation results with different q values

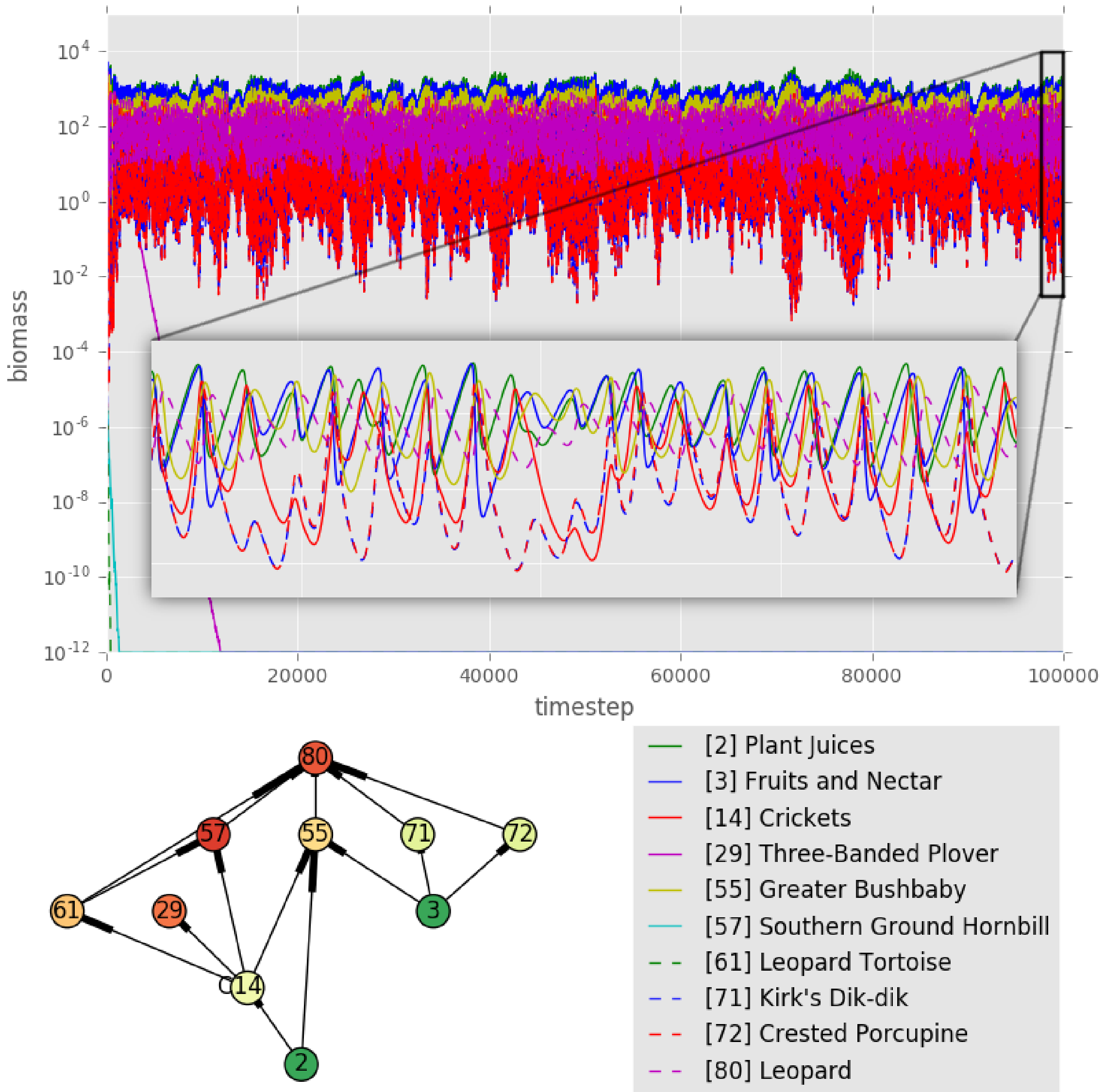
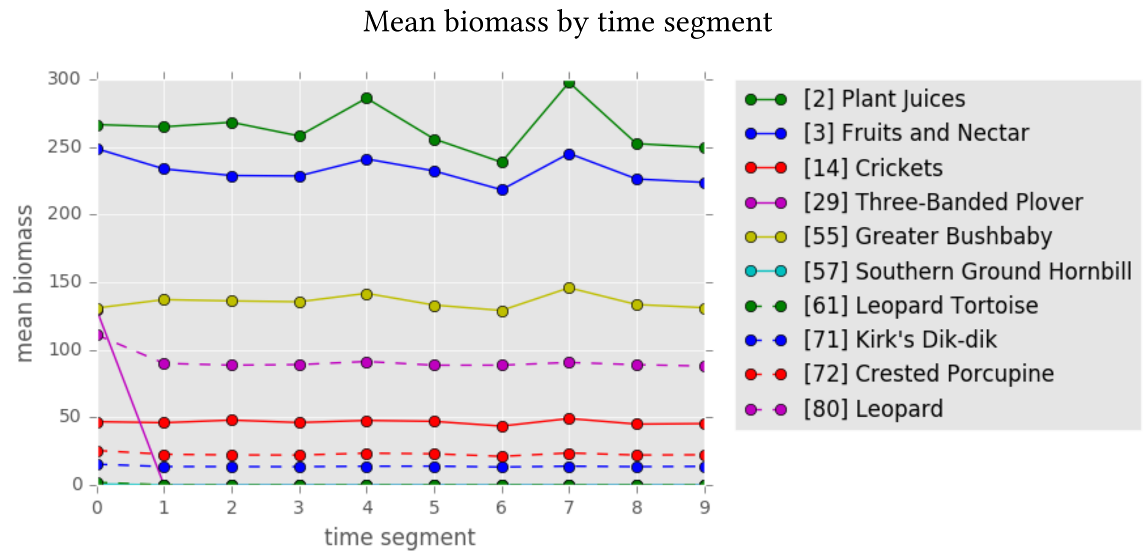


Figure 4.16: Persistent chaotic dynamics with $q = 0.2$. Inset shows zoomed-in view.



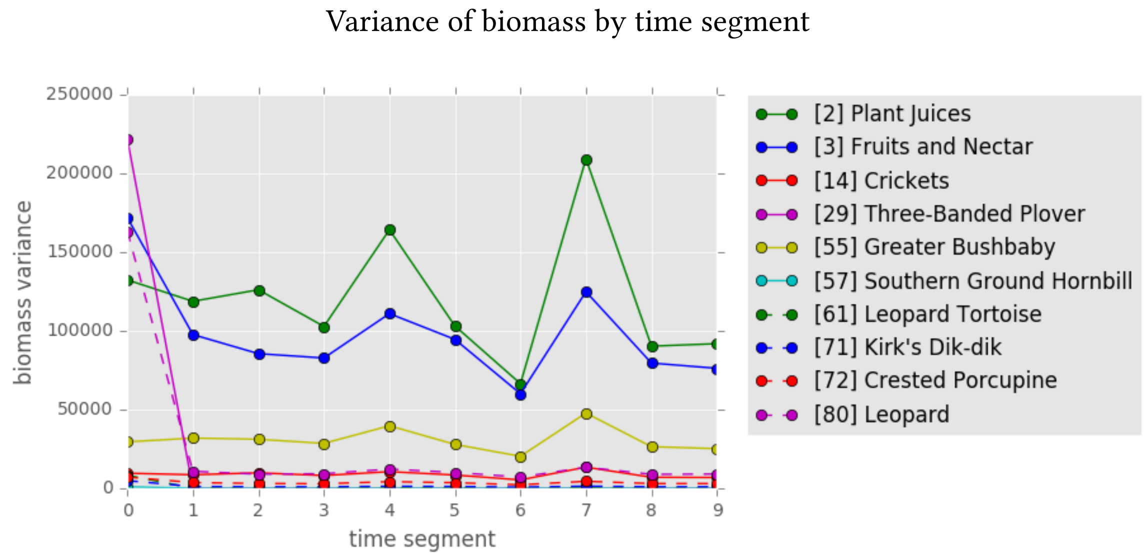


Figure 4.17: Mean and variance of biomass aggregated over 10,000-timestep segments for the persistent chaotic simulation shown in figure 4.16. Segment 0 includes time steps 0–9,999, segment 1 includes time steps 10,000-19,999, and so on.

4.9 Generating sustaining simulations from steady states

We developed the following method to generate parameter values for simulations which sustain all species in a (possibly oscillating) steady state.

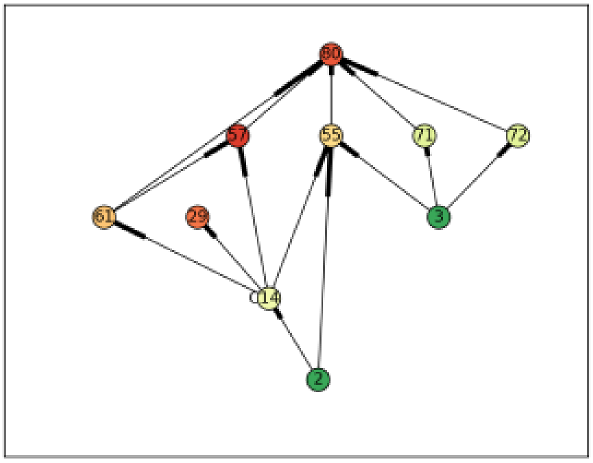
First, we generated parameters values for a set of simulations by drawing them from uniform random distributions. We ran each of these simulations until a steady state was detected, or a maximum simulation duration (100,000 time steps) was reached. The software recorded the type of steady state detected, if any, in each output file.

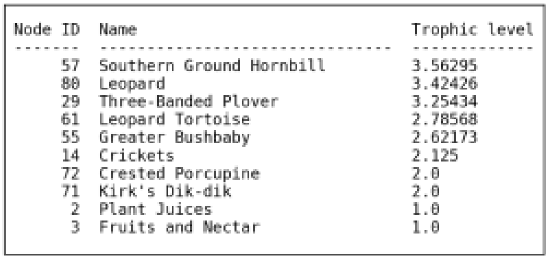
Then, we ran an automated filtering procedure on all of the simulation output files. If an oscillating steady state, or a constant biomass steady state including consumers, was detected, the simulation was identified as "sustaining." For each sustaining simulation, the procedure produced a set of parameter values for a new simulation equal to the original simulation's parameters, except with extinct species removed and the initial biomass of each species set to the final biomass of that species in the original simulation.

Running simulations from these filtered parameter values is guaranteed to produce sustaining simulations, provided that a true steady state was detected. This is because a "steady state" here is defined as a biomass state vector to which the simulation will always return. Essentially, the filtering procedure removes simulations that do not reach a detected steady state and removes the initial dynamics of those that do reach a steady state, keeping only the steady state portion.

This process helps validate the steady state detection algorithms. If a "sustaining" simulation produced by the filtering procedure results in extinctions, then a steady state was falsely detected in the original simulation. (In fact, examination of the results revealed a small number of false positives.)

Figure 4.19 shows some examples of filtered sustaining simulations based on the food web shown in figure 4.18. Note that they include subsets of the species in the food web; this is because species that went extinct in the original simulation are removed.





```
Node ID  Name                            Trophic level
-------  ------------------------------  -------------
     57  Southern Ground Hornbill        3.56295
     80  Leopard                         3.42426
     29  Three-Banded Plover             3.25434
     61  Leopard Tortoise                2.78568
     55  Greater Bushbaby                2.62173
     14  Crickets                        2.125
     72  Crested Porcupine               2.0
     71  Kirk's Dik-dik                  2.0
      2  Plant Juices                    1.0
      3  Fruits and Nectar               1.0
```

Figure 4.18: Food web from which the simulations in figure 4.19 were generated

A disadvantage to this approach is that the desired set of species for the sustaining simulation cannot be chosen in advance. There is no guarantee about which species will persist and which will go extinct. However, it is still useful when one wants to generate sustaining simulations where the exact set of persistent species is not important, such as generating ecosystems for the Convergence game.

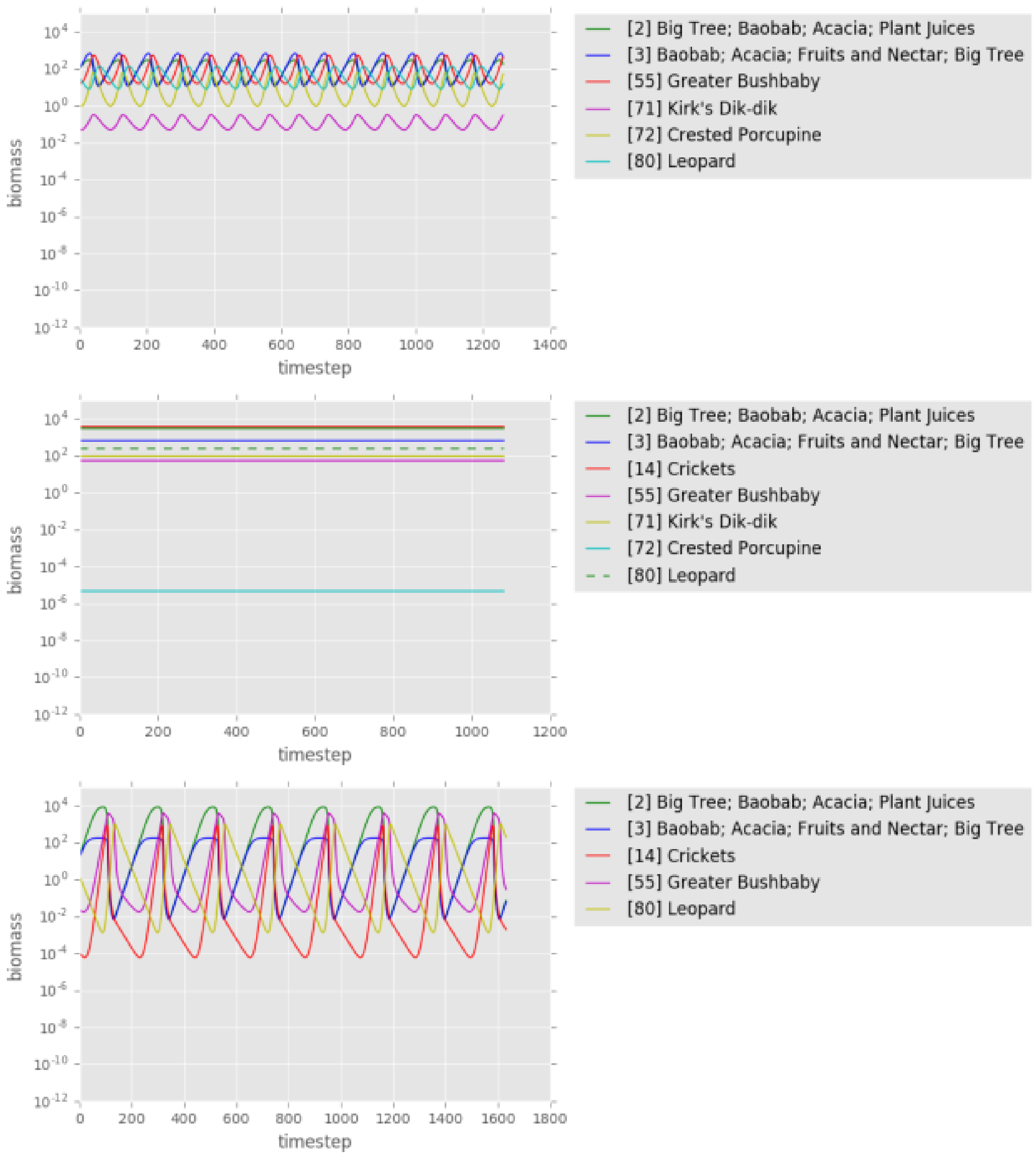


Figure 4.19: Examples of sustaining simulations generated from existing steady-state simulations

4.9.1 Generating sustaining simulations for the Convergence game

We refined the filtering procedure described above to generate simulations suitable for use as target graphs in the Convergence games. As before, the filter only passes simulations that reached an oscillating steady state or a constant steady state with consumers. Additionally, the filter allows setting a threshold on the minimum number of species (we call this parameter *min-species*), and tries to choose simulations that are likely to be visually interpretable by the player.

In some simulations, the range of species biomass will be too large for visual interpretation on the game's linearly scaled y-axis. To illustrate the problem, if one species has a peak biomass of 10,000, but another species has a peak biomass of 10, the second species will be barely visible on the graph. The vertical axis of the graph will have been scaled to accommodate the larger biomass peak. To exclude such simulations, the filter analyzes the last several hundred time steps of the simulation, calculating the maximum biomass of each species as well as the overall maximum biomass. If the ratio of a species' maximum biomass to the overall maximum biomass is below a given threshold (*min-peak-ratio*), the simulation is excluded. Both the number of time steps to analyze and the threshold are configurable parameters.

Another desirable feature is visible variation in biomass. To filter out constant biomass simulations (or those that appear constant), the filter calculates the range of biomass values for each species within the configured time step window. It then calculates the ratio of each range to the overall maximum biomass. If this ratio is below a given threshold (*min-range-ratio*), for all species, the simulation is excluded – that is, a simulation is included only if at least one species' biomass has a large enough range.

4.9.2 Results

To test this method of producing simulated ecosystems for Convergence, we generated 1,000 random simulations for each of four different food webs. These input food webs consisted of 5, 10, 12, and 15 species, respectively. We configured the filter with a *min-peak-ratio* of 0.05 a *min-range-ratio* of 0.05 and set *min-species* to 3. We configured the filter to analyze and keep 600 time steps of steady-state biomass data.

The filter accepted 198 of these 4,000 simulations, or approximately 5%. The surviving species among the these 198 simulations formed 14 distinct food webs ranging in size from 3 to 9 species.

Visual inspection of the biomass data showed that the simulations passing the filter were good candidates for inclusion in the game. Figure 4.20 shows one example. Manual curation would still be a useful step to ensure variety.

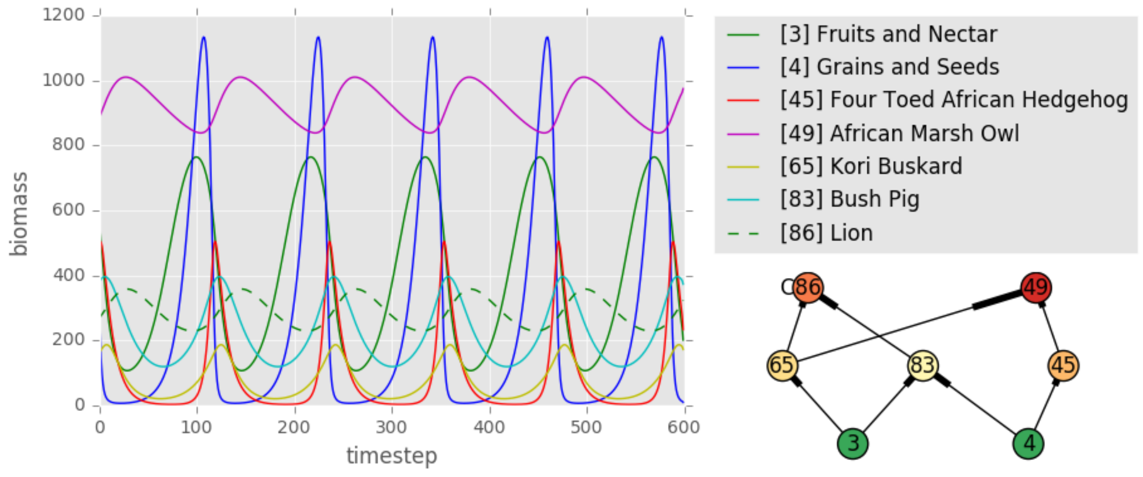


Figure 4.20: A simulated ecosystem selected by the Convergence steady-state filter

4.10 Using decision trees to narrow the parameter search space

One of the objectives when choosing parameter values for an ATN model is to produce sustaining simulated ecosystems. The large parameter space and complex behavior of the model makes this a difficult task. A brute-force search of the parameter space is infeasible. In Section 4.3, we described a method of creating parameter range hints in the Convergence game based on decision trees, thereby helping players to navigate the parameter space. In this section, we use some similar methods. Here, however, we incorporate the insights about steady states described in Sections 4.4, 4.5, 4.8, and 4.9.

The approach in this section also differs from from the one described in section 4.9.

While both methods are for discovering parameters that lead to sustaining ecosystems, the previously described approach does not allow specifying which species should be included in the many different resulting food webs. In this approach, a food web is specified, and we try to find the parameters that keep the most species alive in that specific food web.

We propose a method of iteratively refining the search space by identifying promising regions of the parameter space and focusing the search within those regions. This method uses decision tree classifiers to separate nonpromising regions from promising regions.

4.10.1 Defining "sustaining" simulated ecosystems

To clearly define the objective of producing sustaining simulated ecosystems, we must first define what it means for a simulated ecosystem to be "sustaining."

We define "sustaining" in a simple way: if all species survive indefinitely, no matter how long the simulation runs, then the ecosystem is perfectly sustaining. It is also useful to consider how close an ecosystem is to being a perfectly sustaining one, based on the number of species surviving indefinitely. That is, we can consider an ecosystem with a greater proportion of its species surviving indefinitely to be "more sustaining" than an ecosystem with a smaller proportion of its species surviving indefinitely.

Based on this definition, a simulated ecosystem's "sustainability" can be measured as the inverse of the number of extinctions that occur before the simulation reaches a steady state. After a steady state is reached, we know that no more extinctions can occur, so any surviving species will continue to survive. An ecosystem with zero extinctions at the time a steady state occurs is perfectly sustaining. Ecosystems with more extinctions before the steady state occurs are less sustaining.

4.10.2 Simulations whose sustainability cannot be measured

Not all simulations reach a detected steady state. This can happen for two reasons. First, it is necessary to put an upper limit on the amount of time a simulation can run, and sometimes this is not enough time for the model to reach a steady state. In some cases, visual inspection of the simulated data strongly suggests that a steady state would eventually occur, but this intuition is not enough to draw conclusions. Second, the dynamics can sometimes become seemingly chaotic, with no discernible patterns, as described in section 4.8. These chaotic dynamics can continue for a very long time with no extinctions occurring nor visibly inevitable. Again, however, it cannot be concluded that no extinctions would occur if the simulation were to be run longer.

The sustainability of a simulation that does not reach a detected steady state cannot be measured by the definition above. More extinctions may or may not occur if the simulation is run longer.

These inconclusive simulations arguably provide less information relevant to the search for sustaining ecosystems, compared to simulations that do reach a steady state. Fortunately, they are in the minority, so it is not too great a loss to exclude them from the data driving the search process.

4.10.3 Problem definition

The problem of parameterizing ATN models to result in sustaining ecosystems can be defined as follows: For a given food web, maximize the number of species surviving when the simulation reaches a steady state.

4.10.4 Proposed solution

General approach

This proposed solution takes an approach of iteratively refining the parameter search space. Each iteration, a set of simulations is generated using parameter values randomly drawn from the currently most promising regions of the parameter space. A decision tree is trained on the results, and then used to identify the most promising regions of the parameter space for the next iteration. Over time, this guides the search toward regions of the parameter space that result in more sustaining ecosystems.

This approach uses a classification model "backward" to produce more data similar to the data it was trained on (or, more specifically, one class of data it was trained on). Prior work has described methods of using decision trees [21] [22] and random forests [23] to generate synthetic data that preserves the properties of an original dataset. Of these, the approach in Eno & Thompson (2008) [21] seems most similar to the one described here, though the details of the algorithm are not included.

Reiter (2005) [22] and Caiola & Reiter (2010) [23] specifically describe imputation of fields of existing U.S. Census data records for protecting privacy, rather than generating entirely synthetic records.

In the description of this algorithm, the word “parameter” refers to an ATN model parameter for an individual species, such as “parameter x for species 85,” rather than simply “parameter x.”

Algorithm

The algorithm implemented for the experiment is described as follows.

Algorithm 4 Decision tree-based parameter space search algorithm

Inputs:

- *food-web*, the food web whose parameter space is to be searched
- *batch-size*, the number of simulations in each training and test set
- *seed-region*, a broad region of the parameter subspace to be searched. This region defines the outer bounds of the search space.

Outputs for each iteration:

- Distribution of extinction counts, as relative frequencies
- *regions*, the most promising regions found in this iteration
- Values of intermediate variables and classifier evaluation metrics

Let regions = *seed-region*.

Repeat the following steps until the desired distribution of extinction counts is reached:

1. Generate *batch-size* training simulations of *food-web*, where each simulation's parameters are determined as follows:
    (a) Randomly select a *region* from *regions*.
    (b) Randomly generate the parameter values from independent uniform distributions whose bounds are defined by *region*.
2. Discard simulations that did not reach a steady state.
3. Let *extinction-count-threshold* be the median number of extinctions among all remaining simulations.
4. Assemble a training set as follows:
    (a) One row per simulation
    (b) Features: All parameters in the subspace being searched
    (c) Assign class labels:
        - 0 if *extinction-count* $\geq$ *extinction-count-threshold* ("bad" simulations)
        - 1 if *extinction-count* $<$ *extinction-count-threshold* ("good" simulations)
    (d) If class 1 contains no simulations, increase *extinction-count-threshold* by 1 and reassign class labels.
5. Train a decision tree classifier using this training set.
6. Generate another *batch-size* simulations in the same way as the training set, again discarding simulations that did not reach a steady state.
7. Assemble a test set in the same way the training set was assembled.
8. Evaluate the classifier on the test set and report the results.
9. Combine the training and test sets.
10. Train a decision tree classifier using the combined dataset.
11. Derive the promising regions of the parameter space from the decision tree using the GetPromisingRegions procedure, and assign the results to *regions* for the next iteration.

procedure GetPromisingRegions

Inputs:

- *tree*, a trained decision tree classifier
- *input-regions*, the parameter space regions used to generate the training data for the *tree*

Output:

- *promising-regions*, the promising regions derived from the *tree*

1. Identify *promising-leaves*, the set of leaves in the tree for which the majority class label is 1 (“good”).
2. Compute *root-bounds*, the outermost bounds of *input-regions*, as follows:
    - For each parameter *p*,
        - set the lower bound for *p* to the smallest lower bound among all *input-regions*, and
        - set the upper bound for *p* to the largest upper bound among all *input-regions*.
3. Compute *node-regions*, the regions of the parameter space defined by each node in the decision tree, as follows:
    (a) Set $\textit{node-regions}_{\textit{root}}$ to *root-bounds*.
    (b) In a depth-first traversal, compute $\textit{node-regions}_{\textit{left-child}}$ and $\textit{node-regions}_{\textit{right-child}}$ as follows:
        i. Base case: *current-node* is a leaf (return without further computation)
        ii. Let *p* be the parameter on which *current-node* splits.
        iii. Let threshold be the value of *p* on which *current-node* splits.
        iv. Set $\textit{node-regions}_{\textit{left-child}}$ to $\textit{node-regions}_{\textit{current-node}}$.
        v. Update the upper bound for *p* in $\textit{node-regions}_{\textit{left-child}}$ to the minimum of its current value and *threshold*.
        vi. Set $\textit{node-regions}_{\textit{right-child}}$ to $\textit{node-regions}_{\textit{current-node}}$.

    vii. Update the lower bound for p in *node-regions*$_{\textit{right-child}}$ to the maximum of its current value and *threshold*.

4. Return *promising-regions*, the subset of *node-regions* corresponding to *promising-leaves*.

Discussion of algorithm

The critical idea underpinning the GetPromisingRegions procedure is that each node in a decision tree is associated with a region of the feature space. An input sample can be described as a point in the feature space. When a binary decision tree makes a prediction for an input sample, that sample is routed down the tree, going left or right depending on the split criteria of the current node, until it reaches a leaf. The region corresponding to a node in the decision tree encompasses all possible input samples that would be routed through that node. The region corresponding to a leaf, then, encompasses all possible samples that would be routed to that leaf.

4.10.5 Experiment

Hypothesis

The hypothesis of this experiment is that the distribution of extinction counts among the generated simulations will shift closer to zero each iteration as the search process proceeds. In other words, each iteration will result in a greater proportion of simulations with fewer extinctions.

Setup

We used the DepthControlledRandomSuccessorSubgraph algorithm, sampling from the World of Balance Serengeti food web, to sample three random food webs: a five-species food web with one basal species, a ten-species food web with two basal species, and a fifteen-species food web with three basal species, shown in figure 4.21.

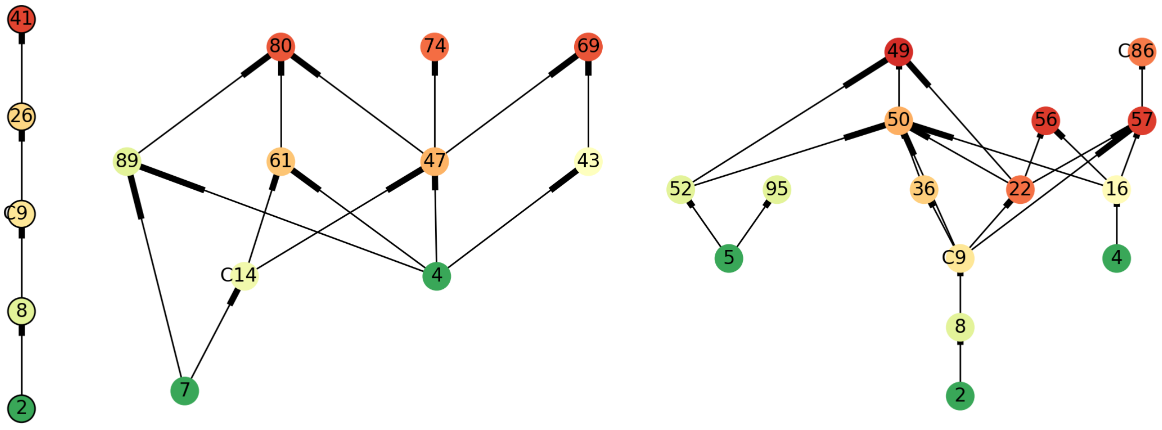


Figure 4.21: Three food webs used in the decision tree search experiment

We used ATN Simulator to run batches of 1000 simulations in which we only varied x (metabolic rate), K (carrying capacity), and initial biomass. We fixed other parameters at their default values. The parameter ranges defining the initial region in which to start the search were:

- x: 0..1
- K: 100..10,000

- initial biomass: 100..5,000

Since x only applies to animals and K only applies to plants, and initial biomass applies to both, the parameter spaces for these three food webs include 10, 20, and 30 parameters, respectively. We used the decision tree classifier from the scikit-learn[3] software package [24]. This implements a version of the CART (Classification and Regression Trees) algorithm [25]. We configured the classifier to use Gini impurity as the split criterion. To reduce class bias, we configured the classifier to balance the classes automatically by weighting samples proportionally to class size. To reduce overfitting, we constrained leaves to contain a minimum of 1% of the total sample size.

4.10.6 Results

We ran each of the three food webs for 10 search iterations and collected various metrics for each iteration. These metrics include the distribution of extinction counts, the *extinction-threshold* used to separate the classes, the number of simulations in each class, the F1 score on the test set, and the size of the decision tree.

Figure 4.22 shows the distributions of extinction counts for each of the 10 search iterations. The distributions show a general, if inconsistent, trend toward fewer extinctions. This shows that the search process is guiding the search toward more promising regions of the parameter space with some degree of success. For the 5-species food web,

[3] We switched from Weka to scikit-learn late in the project after learning that it met our needs better than Weka. Scikit-learn, a Python library, provides better integration into our Python-based data pipeline than does Weka, which is Java-based.

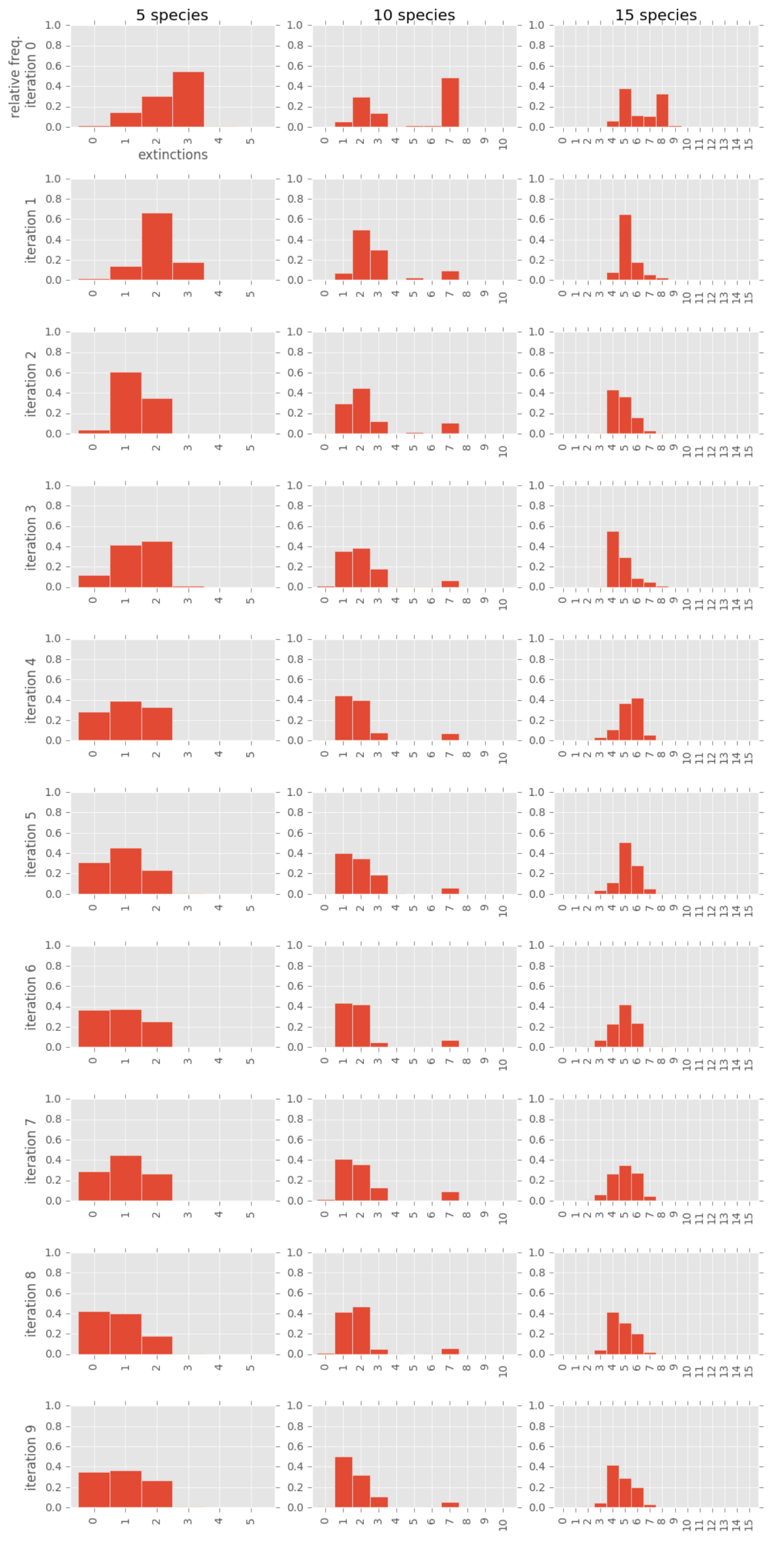


Figure 4.22: Distributions of extinction counts by iteration of the decision tree search experiment

the number of perfectly sustaining ecosystems increases from almost none to about 40%. For the 10-species food web, the number of perfectly sustaining ecosystems remains near zero, but the number of ecosystems with a large number of extinctions decreases dramatically, and the number of ecosystems with only one extinction reaches about 50%. The results for the 15-species show the least consistent trend and include very few ecosystems with fewer than 4 extinctions. However, the number of ecosystems with no more than 4 extinctions out of 15 species exceeds 40%. Unfortunately, no perfectly sustaining ecosystems are found for the 15-species food web.

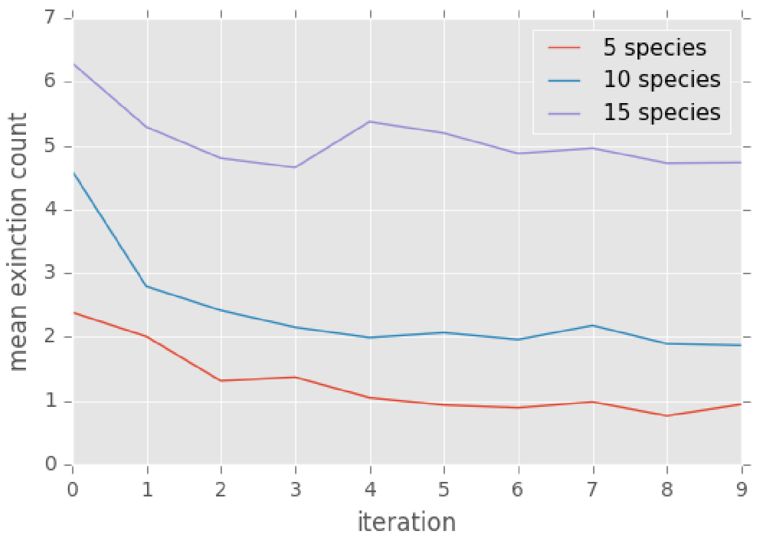


Figure 4.23: Mean extinction count by iteration of the decision tree search experiment

Figure 4.23 simplifies the picture, showing only the average extinction count by iteration of the search process. For all three food webs, the mean extinction count shows a clear decrease in the first 3-4 iterations. The trend levels off in subsequent iterations, and it does not appear that continuing the process beyond 10 iterations would yield much improvement with the current algorithm. By iteration 3, the 15-species food web has reached its lowest mean extinction count. However, as shown in figure 4.22, it does gain some ecosystems with only three extinctions in iteration 4.

The classification performance of the decision trees is one factor in the overall performance of the search process. We evaluate the performance of the decision tree in each iteration by generating a test set of simulations. We calculate the F1 score for both classes, and average the two F1 scores. To give performance both classes equal weight, we use an unweighted average. Figure 4.24 shows this average score by iteration for each food web.

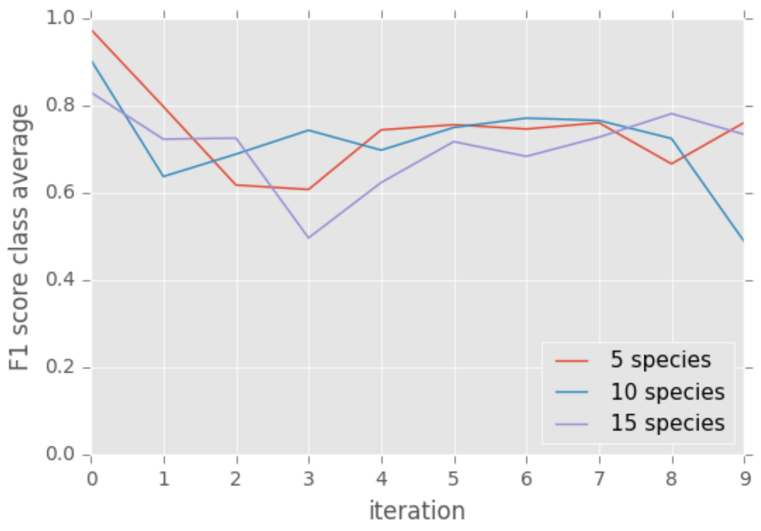


Figure 4.24: Average F1 score by iteration of the decision tree search experiment

The performance of the decision trees is good in the first iteration, when they are trained on simulations encompassing broad, contiguous regions of the parameter space. However, it drops significantly in subsequent iterations. In part, the F1 scores are limited by the minimum leaf weight of the decision tree. This minimum is set to combat overfitting, but tends to produce a tree that does not perfectly fit the training data, which inherently limits its performance on the test data.

The complexity of the decision trees, measured as the number of nodes (figure 4.25), does not have a trend that is consistent among the three food webs.

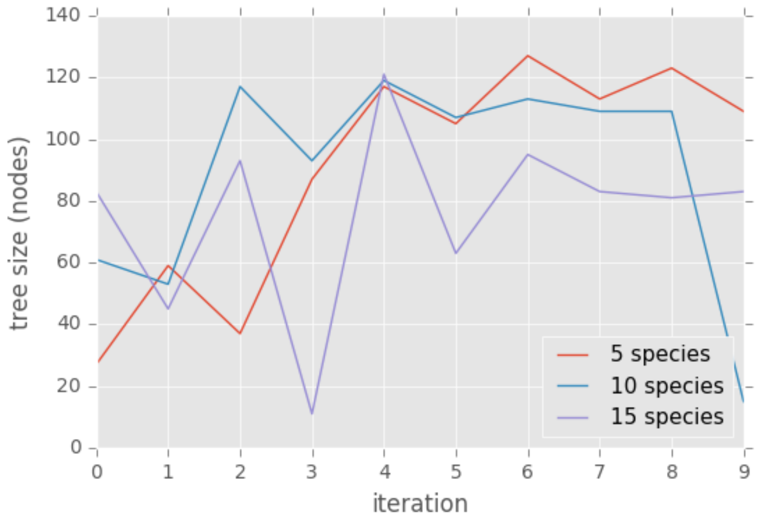


Figure 4.25: Average tree size by iteration of the decision tree search experiment

The very small tree in iteration 3 for the 15-species food web occurs when the classes become extremely skewed, with only 5 instances of the "good" class in the training and test sets combined. A similar situation of class imbalance occurs in iteration 9 for the 10-species food web. This suggests a need for additional measures to keep classes balanced.

Figure 4.26 shows the tree from iteration 3 of the 5-species food web, which is typical in size. Blue nodes have a “good” class majority, while orange nodes have a “bad” class majority. Blue leaves represent the promising regions of the parameter space.

4.10.7 Conclusions

The results are mostly consistent with the hypothesis that the distribution of extinction counts would shift closer toward zero each iteration. However, the improvements begin to level off after a small number of iterations, and do not reach a place where most ecosystems are sustaining.

There are some areas of the process that could be improved to achieve better performance.

Class balance could be improved, as it seems that weighting the classes during the training process does not adequately correct for instances of extreme class imbalance. There simply is not enough data in the minority class. Because more data can always be generated, undersampling the majority class is tempting, but when the minority class is tiny, this would require generating and discarding large numbers of time-consuming simulations.

Classification accuracy could be improved by generating additional training data in each iteration, until performance on the test set reaches a minimum threshold.

Computational performance is a clear area for improvement. Running the simulations takes the vast majority of the computation time. For this experiment, the entire

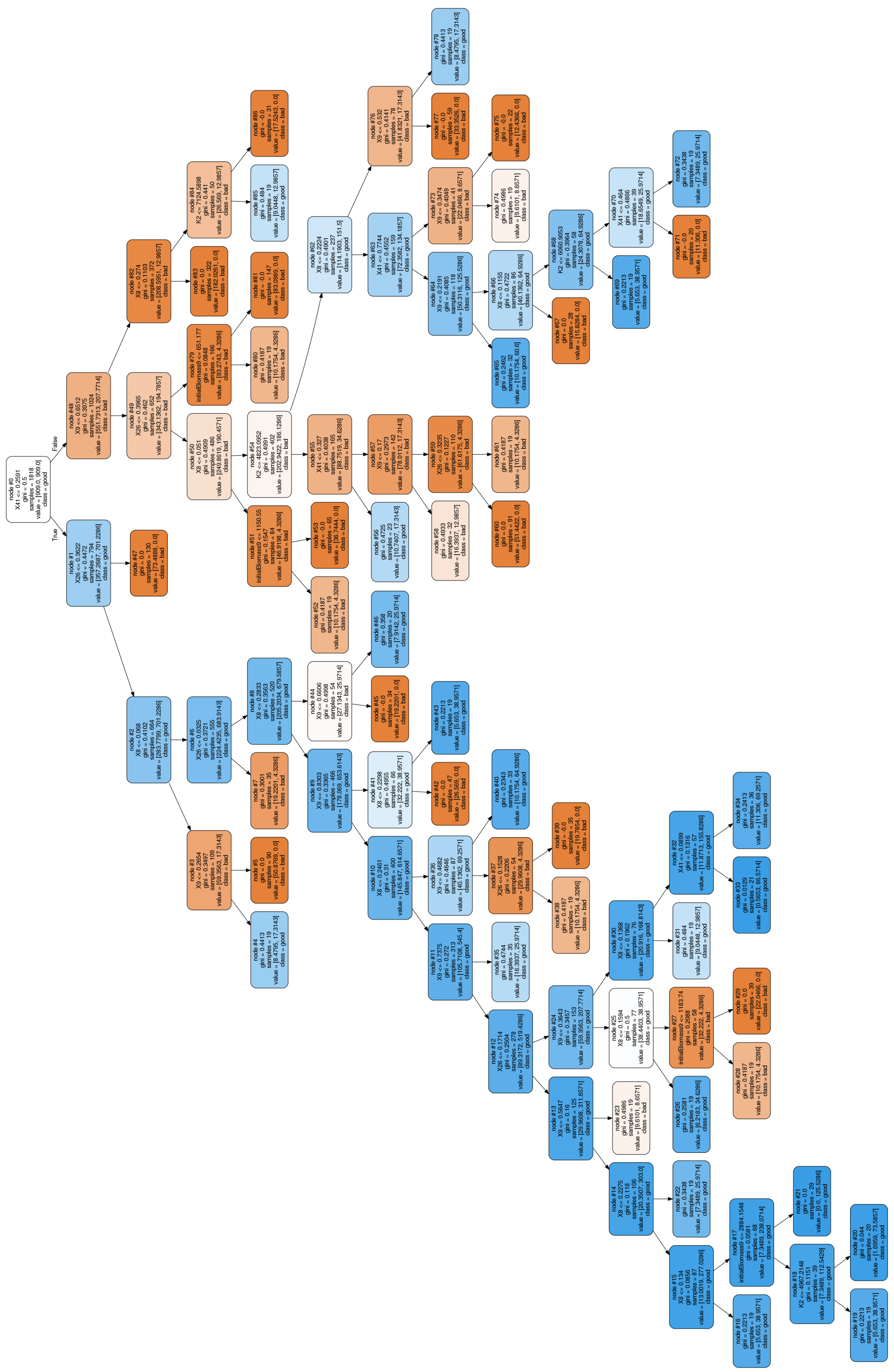


Figure 4.26: Example decision tree from decision tree search experiment

10-iteration process with 20,000 simulations per food web took 1 hour for the 5-species food web, 7.5 hours for the 10-species food web, and nearly 19 hours for the 15-species food web. We ran the software on an Amazon Web Services c3.large instance (a dual-core Intel Xeon E5-2680 v2 processor at 2.80GHz). The largest reductions in computation time could be achieved by reducing the number of simulations required. Optimization of the simulation code in ATN Simulator would clearly also improve the situation.

Despite leaving room for improvement, this experiment is a proof of concept of the general approach of using a machine learning model to iteratively refine the parameter search space. With further work, a more sophisticated algorithm, and perhaps more sophisticated classification models than decision trees, we believe this could be a successful approach.

4.11 Using the decision tree search to generate Convergence simulations

One potential application of the decision-tree-based parameter space search described in Section 4.10 is generating sustaining simulation data for the Convergence game. In Section 4.9.1, we described an approach to generating these Convergence simulations by running many random simulations to a steady state, removing the pre-steady-state portions, and filtering the resulting simulations for visual suitability. To generate the random simulations, parameter values are drawn from broad uniform distributions.

Here, we evaluate the results of replacing these broad uniform distributions with the promising regions discovered by the decision tree search process. In other words, we evaluate using the output of the decision tree search process as input to the Convergence simulation filtering process. We wish to see if this can yield improved results over using the Convergence filtering process alone, in terms of the number of Convergence simulations produced or the size of food webs in these simulations.

For our control simulations, we followed the procedure described in Section 4.9.1 using the three food webs described in Section 4.10.

For our test simulations, instead of using the broad uniform distributions, we drew parameter values from the promising regions from the final iteration of the evaluation in Section 4.10.

We generated 1,000 simulations for each of the three food webs and two groups.

4.11.1 Results

Table 4.8 compares the results for the control and test groups of simulations. The table shows the number of simulations at two different stages in the filtering process. *Sustaining sims* counts the number of steady-state simulations including consumers.[4] *Convergence sims* counts the number of *Sustaining sims* that were accepted by the Convergence filter. In all cases, the Convergence filter removes a significant portion of the candidate

[4] Note that this number is greater than 1,000 for the 15-species control group. This is because the first stage of the filter identifies food webs that have become disconnected due to extinctions and separates them into independent simulations, thereby potentially increasing the total number of simulations.

steady-state simulations. The proportion varies by food web, and is neither consistently larger nor smaller for the test group.

Input food web		Control	Test
5 species	Sustaining sims	946	854
	Convergence sims	180	353
	Output food webs	3	3
	Avg. food web size	3.31	4.30
10 species	Sustaining sims	829	689
	Convergence sims	23	0
	Output food webs	2	0
	Avg. food web size	3.87	-
15 species	Sustaining sims	1611	568
	Convergence sims	28	6
	Output food webs	2	2
	Avg. food web size	3.14	6.33

Table 4.8: Results of applying decision tree search regions to Convergence simulation filter. *Sustaining sims* is the number of simulations including both producers and consumers in a steady state after separating disconnected food webs into separate simulations. *Convergence sims* is the number of these simulations that were accepted by the Convergence filter for visual suitability for the game. *Output food webs* is the number of distinct food webs among the accepted simulations. *Avg. food web size* is the average size of the food webs included in the output, weighted by the number of simulations for each food web.

For the 5- and 10-species food webs, using the promising regions from the decision tree search as input to the Convergence filter results in simulations including more surviving species, on average (see *Avg. food web size* in the table). This shows a potential benefit to connecting the two processes in this way.

For the 10-species food web, no simulations in the test group passed the Convergence filter, despite the fact that many sustaining simulations were extracted from the input dataset. Examination of individual simulation data reveals that the sustaining simulations are rejected by the Convergence filter for one of two reasons. First, many of them have constant biomass, so the *min-range-ratio* of 0.05 excludes them. Second, many of them include one or two species that persist with very low biomass compared to the other species, resulting in exclusion due to the *min-peak-ratio* of 0.05.

4.11.2 Conclusions

Our experiment shows mixed results from this method of combining the decision tree parameter search with the Convergence filtering process. For the 5-species food web, we see improvements in the both the number of Convergence simulations and the average food web size. For the 15-species food web, we obtain much larger food webs on average, but also far fewer simulations. For the 10-species food web, combining the processes in this way gains us nothing: no Convergence simulations are produced due to the way the decision tree parameter search produces simulations with either constant biomass or species that persist with very low biomass – too low to be usable in the game.

It bears mentioning that the Convergence filtering process is designed for a linear-scale y axis. Many of the sustaining simulations in the 10-species test group are visually interpretable with a log-scale y axis. If Convergence used a log scale rather than a linear scale for the y axis, simulations in which some species persisted at very low biomass could be included.

One way in which the decision tree search might be adapted to producing Convergence ecosystems would be to classify “good” vs. “bad” simulations based not only on the number of extinctions, but also incorporating the same visual suitability criteria used by the Convergence filter.

Chapter 5

Conclusions and Future Work

Allometric trophic network (ATN) models provide useful ways of examining ecological phenomena. However, they are difficult to parameterize to accurately represent real ecosystems, due to the the models' internal interdependencies, complex dynamics, and many parameters. We have sought ways of parameterizing ATN models to produce self-sustaining simulated ecosystems using machine learning-based approaches and gamification within the framework of the World of Balance game.

The motivation of this work is to improve our understanding of ATN models and their parameters so that they can be used more effectively in the study of ecosystems. We also wish to help educate people through the World of Balance game, inspiring and encouraging game players to learn about ecology.

Our contributions include a graph sampling algorithm for food webs, a performant and reusable software package for ATN simulation, an algorithm for detecting steady states in ATN simulations, an improved environment score for World of Balance, enhancements to the Convergence game, and a machine-learning-based method for ATN model optimization.

5.1 Future work

The concept and implementation of parameter range hints for Convergence game should be evaluated and improved by conducting a user study. This study could compare the performance of players who receive the hints with a control group who do not receive the hints. Two suggested areas for improvement of the hints are the ability to show multiple, discontiguous ranges, and some representation of parameter interdependence. The second point refers to the fact that multiple, separate regions of the parameter space may lead to a reasonable solution, and in such a situation, optimal parameter ranges are not independent. This may require an more dynamic representation of the hints that responds to user input.

The precision of the steady state detection algorithm could be improved for values of $q > 0$, as described in Section 4.5.6. A more rigorous evaluation incorporating more food webs and parameter variations would help with this improvement.

There is much room for improvement in the parameter space search method described in section 4.10. Issues of class balance should be addressed. This may require determining another basis on which to assign classes other than a threshold on the number of extinctions, which may be too coarse a measure. Classification accuracy could be improved. Computational performance could be improved by finding ways of running fewer simulations, running shorter simulations, or optimizing the simulation code.

Simulations sometimes exhibit persistent chaotic dynamics. This is an interesting phenomenon that warrants further study. It would be useful if it could be determined whether all species in a chaotic system will survive.

The system-wide carrying capacity model described in Section 4.6 is more richly descriptive than the growth model currently used in World of Balance and Convergence. It is worth considering switching to this model, which is already implemented in the core simulation code. This would require changing the controllable parameters in Convergence, which currently uses K as a parameter for each producer. For example, the growth rate r could be substituted for K, and a separate slider could be added for the system carrying capacity K_s.

Resiliency is an important factor in ecological sustainability. For each individual simulation included in our analyses, the model parameters were fixed for its entire duration, and no new species or biomass were introduced into the system. By introducing parameter and biomass changes into simulations that have stabilized, some of the machine learning experiments here could be extended to predict resiliency following a similar approach to that of Berlow et al. (2009) [14].

The applications of machine learning we have described have been based on data structured a particular way. The datasets are at the simulation level – that is, each data point corresponds to a simulation. The features consist solely of ATN model parameters and the initial biomass of each species. Most importantly, each machine learning model is trained on data generated from a single food web, which means that it cannot make predictions about other food webs. Thus, generalizability across different food webs is limited. An approach that allows machine learning models to make predictions about arbitrary food webs would be useful. This would entail extracting features about food web structure. Williams & Martinez (2000) [26] define 13 such structural properties, and these would be excellent candidates for features. It would also require encoding ATN model parameters as features in a way that is uniform across food webs.

Bibliography

[1] P. Yodzis and S. Innes, "Body size and consumer-resource dynamics," *American Naturalist*, pp. 1151–1175, 1992.

[2] R. J. Williams and N. D. Martinez, "Stabilization of chaotic and non-permanent food-web dynamics," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 38, no. 2, pp. 297–303, 2004.

[3] R. J. Williams, U. Brose, and N. D. Martinez, "Homage to yodzis and innes 1992: scaling up feeding-based population dynamics to complex ecological networks," in *From energetics to ecosystems: the dynamics and structure of ecological systems*, pp. 37–51, Springer, 2007.

[4] J. E. Cohen, F. Briand, and C. M. Newman, "Community food webs: data and theory," *Biomathematics*, vol. 20, 1990.

[5] I. Yoon, G. Ng, H. Rodrigues, T. Nguyen, J. H. Paik, S. Yoon, R. Williams, and N. D. Martinez, "Iterative design and development of the 'World of Balance' game: from ecosystem education to scientific discovery," in *Games Innovation Conference (IGIC), 2013 IEEE International*, pp. 283–290, IEEE, 2013.

[6] A. Boit, N. D. Martinez, R. J. Williams, and U. Gaedke, "Mechanistic theory and modelling of complex food-web dynamics in lake constance," *Ecology letters*, vol. 15, no. 6, pp. 594–602, 2012.

[7] A. Kuparinen, A. Boit, F. S. Valdovinos, H. Lassaux, and N. D. Martinez, "Fishing-induced life-history changes degrade and destabilize harvested ecosystems," *Scientific reports*, vol. 6, p. 22245, 2016.

[8] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, *et al.*, “Predicting protein structures with a multiplayer online game,” *Nature*, vol. 466, no. 7307, pp. 756–760, 2010.

[9] F. Khatib, F. DiMaio, S. Cooper, M. Kazmierczyk, M. Gilski, S. Krzywda, H. Zabranska, I. Pichova, J. Thompson, Z. Popović, *et al.*, “Crystal structure of a monomeric retroviral protease solved by protein folding game players,” *Nature structural & molecular biology*, vol. 18, no. 10, pp. 1175–1177, 2011.

[10] E. B. Baskerville, A. P. Dobson, T. Bedford, S. Allesina, T. M. Anderson, and M. Pascual, “Spatial guilds in the serengeti food web revealed by a bayesian group model,” *PLoS Comput Biol*, vol. 7, no. 12, p. e1002321, 2011.

[11] N. Muttil and K.-W. Chau, “Machine-learning paradigms for selecting ecologically significant input variables,” *Engineering Applications of Artificial Intelligence*, vol. 20, no. 6, pp. 735–744, 2007.

[12] Y. Shan, D. Paull, and R. McKay, “Machine learning of poorly predictable ecological data,” *Ecological modelling*, vol. 195, no. 1, pp. 129–138, 2006.

[13] R. N. Gutenkunst, J. J. Waterfall, F. P. Casey, K. S. Brown, C. R. Myers, and J. P. Sethna, “Universally sloppy parameter sensitivities in systems biology models,” *PLoS Comput Biol*, vol. 3, no. 10, p. e189, 2007.

[14] E. L. Berlow, J. A. Dunne, N. D. Martinez, P. B. Stark, R. J. Williams, and U. Brose, “Simple prediction of interaction strengths in complex food webs,” *Proceedings of the National Academy of Sciences*, vol. 106, no. 1, pp. 187–191, 2009.

[15] J. Cotter, “Predicting effects of modifying species’ parameters in an allometric trophic network model,” Master’s thesis, San Francisco State University, 2015.

[16] J. E. Cohen, *Food webs and niche space.* No. 11, Princeton University Press, 1978.

[17] B. Kendall, “The macroecology of population dynamics: taxonomic and biogeographic patterns of population cycles,” *Ecol Lett*, vol. 1, pp. 160–164, 1998.

[18] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal*, vol. 27, pp. 379–423, July 1948.

[19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[20] J. R. Quinlan, *C4. 5: programs for machine learning.* Elsevier, 2014.

[21] J. Eno and C. W. Thompson, "Generating synthetic data to match data mining patterns," *IEEE Internet Computing*, vol. 12, no. 3, 2008.

[22] J. P. Reiter, "Using cart to generate partially synthetic public use microdata," *Journal of Official Statistics*, vol. 21, no. 3, p. 441, 2005.

[23] G. Caiola and J. P. Reiter, "Random forests for generating partially synthetic, categorical data.," *Trans. Data Privacy*, vol. 3, no. 1, pp. 27–42, 2010.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[25] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees.* CRC press, 1984.

[26] R. J. Williams and N. D. Martinez, "Simple rules yield complex food webs," *Nature*, vol. 404, no. 6774, pp. 180–183, 2000.