EPiC Series in Computing

Volume 101, 2024, Pages 93–102

Proceedings of the 16th International Conference on Bioinformatics and Computational Biology (BICOB-2024)

A simulation study on the sustainability of an ecosystem

Christopher Palazzolo, Sayed Mansour Hashemipoor and Wenying Feng*

Department of Computer Science
Trent University Durham Greater Toronto Area
Oshawa, Ontario Canada L1J 5Y1
{christopherpalazzolo, sahashemipoor, wfeng}@trentu.ca

Abstract

This paper investigates the sustainability of an ecosystem that involves the consumption and reproduction of wildlife on a day-by-day basis in addition to the growth of plants. Different from the traditional approaches such as the reinforcement learning algorithms or the predator-prey dynamical system analysis, we applied simulation techniques and developed computer programs that manage the evolution of the system. The results provide visualization for the system. Limitations and further improvements of the study are also discussed.

Keywords: Ecosystem, food chain, predator-prey, population dynamics, simulation.

1 Introduction

As an interdisciplinary research area, ecological population dynamics has attracted researchers from various fields, including Mathematics, Computer Science, Biology, Physics, Engineering, and others. When studying the stability of ecosystems involving populations of two or more species [10, 11], common approaches include modelling using machine learning [9], differential equations such as the traditional Lotka-Volterra equations [2], difference equations in the form of discrete-time systems [1] and other methodologies.

Different from the traditional approaches, we apply simulation techniques to investigate an ecosystem that involves the consumption and reproduction of wildlife on a day-by-day basis in addition to the growth of plants. Our model is simpler in development but provides visualization for the dynamical system. The results assist understanding sustainability in nature by determining the ratio of different plants and animals required to keep the system stable.

To control the evolution of the programs, we apply the Canonical Evolution Strategies (ES) algorithm introduced in [3]. At a high level, the programs start with the species of animal at the bottom of the food chain and work backward to the top. For each species, the algorithm starts with the animal who has the highest mass. They have the first pick of everything for the current day. To simulate consumption, the algorithm finds the animal of an edible species with the highest mass, but with a speed less than the animal who is looking for something to eat.

*Corresponding author.

H. Al-Mubaid, T. Aldwairi and O. Eulenstein (eds.), BICOB-2024 (EPiC Series in Computing, vol. 101), pp. 93–102

If such an animal can be found, it will be eaten. This process repeats until the animal whose turn it is to eat is full. If an animal eats and has a child in its care, the animal will give the calories to each child instead of itself, only consuming what is left over.

Next, the algorithm checks if the animal is eligible for procreation. In the simulation, an animal will either be an "alpha" or a "beta", with a 50% chance of being born as either. Both one alpha and beta animals are required to reproduce. The algorithm determines if an animal is ready for procreation by checking if the animal does not have any children in its care, and the gestation period for that animal has passed since its last procreation. If both of these conditions are true, then the animal will attempt to locate another animal of its species who has the highest mass, is ready for procreation, and is the opposite of them (in other words, if the current animal is an alpha, then it must search for a beta). If such an animal can be found, procreation can occur. If the children survive birth, they are put in the care of the alpha who is responsible for feeding them. Both parents must wait for the gestation period before procreating again.

Finally, the algorithm checks if the animal has reached its maximum age, or if the animal has not consumed a sufficient number of calories. In both cases, the animal dies. For simplicity, the carcass is removed from the simulation and cannot be used as a food source for a predator. The simulation will run until one of the species goes extinct from the ecosystem, at which point, the program will terminate as the ecosystem needs animals of every species to sustain every other species and to limit them from growing out of control and consuming all of the available prey faster than the prey can reproduce.

The simulation model was implemented in Python and is comprised of two scripts. The first script is the simulation itself. It runs the ecosystem and manages the animals within. Every 5 simulated days, it dumps the current state of the simulation to a JSON file, which contains the number of animals of each species at the end of every fifth day. The second script takes that JSON file and hosts a small web server. By visiting the home page of the web server, the contents of the JSON file are visualized. It displays the current day and line graphs of the counts per species using the ChartJS library. The web page refreshes the data every 10 seconds.

The rest of the paper is organized as follows. In Section 2, we discuss parameters and details of model development. Section 3 describes alternative configurations, experiments and performance measures. Some statistics analysis on the results is presented in Section 4. Finally, limitations and future work are discussed in Section 5.

2 Parameters and model assumptions

Animals at the bottom of the food chain will eat grass. Simulating millions of individual blades of grass is impractical, but necessary for the level of detail used in this simulation. To get around this, the algorithm keeps a count of the number of blades of grass. For each blade that is eaten, this counter is decremented, and the number of blades enters a queue, and will reach the top of the queue in the amount of time it takes for grass to grow from a seed into a full blade. At that point, the number is added to the total grass available. This method allows grass to be a finite but renewable resource without a high amount of overhead.

The simulation determines how many calories each animal should consume per day using the Resting Energy Requirement (RER) [6, 8]. Thus the energy required to perform essential body functions like digestion, respiration, heart functions, brain functions, etc. The RER is multiplied by some factor to represent the energy spent on physical activities when the animal is not at rest [5]. According to Ohio State University Veterinary Medical Center, active and working dogs have a multiplier of 2-5. From this information, the following formula is applied

in our models [6]:

$$\text{Daily Calories} = \text{Multiplier} * 70(\text{weight})^{\frac{3}{4}}. \quad (1)$$

If the calories that the animal consumed over their lifetime divided by the number of calories, they should have consumed drops below 10%, the simulation considers the animal to have starved.

When a child is born from its parent, it is not an exact copy. A slight mutation will be applied to the weight, speed, RER multiplier, and minimum hunger to eat to simulate evolution and natural selection within the ecosystem. The mutation is generated from a standard normal distribution with some multiplier to have detailed control over the rate of evolution. The weight, speed, and RER multiplier uses a mutation multiplier of 0.1, and the minimum hunger to eat has a multiplier of 0.01 (1%). The idea of using a multiplier on a standard normally distributed random number was inspirited from the Canonical ES algorithm [3].

The starting animals in the ecosystem are created as adults who must wait for one gestation period before they are ready to procreate. All animals (regardless of whether they are born during the simulation or are original animals) start with one day's worth of calories. In other words, they are treated as if they are full and have no need to eat right away. This prevents them from starving immediately after birth.

If the animal is considered a child, and in the care of another animal, it will not be able to obtain food on its own and it will be unable to procreate. Instead, the child starts with a very low mass which grows at a linear rate each day until they reach their full mass. At this point, they would no longer be considered a child and would leave the care of the alpha. Children are not exempt from being eaten by a predator.

The food chain evaluation is modelled as the following: grass will act as the primary producer, which will be consumed by a grasshopper followed by a frog, a snake, and finally a hawk which will be the apex consumer. For simplicity, the simulation will not consider the sun or decomposers, and due to performance issues, the hawk will not be included in the simulation.

For each species in the simulation, the system needs to know the number of calories that would be gained for each kilogram of animal consumed, the age (in days) when the animal can be considered an adult, a list of prey, weight, speed, maximum age, gestation period, and children born per procreation. Table 1 shows parameter values. The grass is modelled after annual ryegrass, the grasshopper is modelled after the white-whiskered grasshopper, the frog is modelled after the common frog, and the snake is modelled after the brown tree snake. These animals were selected because the prey they are supposed to eat in the simulation is part of the diet of that specific type of animals.

It is known that grass has an average of "33 calories per 100 grams" [4]. To convert this to calories per blade, the average mass of a blade of grass is required. As a general case, an assumption of 0.055 grams per blade is used, which is the average of the range. Therefore, the simulation will run with the assumption that eating a single blade of grass will give the consumer 0.01815 calories.

In terms of the grass growth, depending on the type of grass "...it can take anywhere between five and 30 days for grass seed germination to begin. After that, it takes another three to four weeks before the grass is long enough to mow." [7]. The assumption will be made that long enough to mow refers to the blade being an adequate height to be eaten. Between the germination and growth periods, it will take an average of 26 to 38 days for the grass to regrow from being eaten.

	White whiskered grasshopper	Common frog	Brown tree snake
Calories gained (/kg)	2,570	730	930
Adult age	25-30 days	3 months	2-4 years
Weight (kg)	M=0.00011 F=0.00031	0.0227	0.525
Speed (m/s)	3.048	4.4704	5.89408
Lifespan	12 months	unknown	10-15 years
Gestation period	lays eggs about every 30 days	2-3 weeks	28-45 days
Children born per procreation (eggs)	3-5	up to 4000	3-12
Probability of child surviving birth	50%	as low as 1%	assumed to be 100%

Table 1: Parameter values for each species

3 Configurations and experiments

The goal of the simulation is to find a ratio of animals in a food chain in order to create a relative stable ecosystem, thus an ecosystem where no species go extinct in a certain time period. The longer an ecosystem remains stable, the better the selected ratio. The programs use random number generators of various distributions so running the simulation with the same parameters and ratio of species can produce different results. After a considerable amount of experimentation, it was found that 10 animals of a species have enough alpha and beta animals to start the population, regardless of the lifespan, gestation period, and children per procreation (assuming there is a sufficient amount of food to sustain the animals). For that reason, the apex consumer of the ecosystem will start with a population of 10. From the ratios provided to the simulation, the number of starting animals per species can be calculated.

A. Configuration 1

Using the calculated ratios, the following simulation was run with 10 snakes, 5 frogs per snake, 359 grasshoppers per frog, and 613 blades of grass per grasshopper. Therefore, the starting counts were 10 grasshoppers, 50 frogs, 17,950 grasshoppers, and 11,003,350 blades of grass.

After only 6 days, the entire frog population died out and the simulation ended. This is because the frogs were unable to reach their first procreation cycle to average out with a stable population. To give the system a fair attempt, the initial number of frogs was multiplied by 200 (the average number of offspring per frog during each cycle) to start the simulation with the proper number of adults. Therefore, the starting counts were 5 snakes, 5000 frogs, 8975 grasshoppers, and 5,501,675 blades of grass.

This simulation ran for 298 days, at which point the frog population died out. Interestingly, all of the charts level out over time and become very consistent except for the frogs whose population fluctuates wildly and then around day 260, begins to drop (see Fig 1 and Fig. 2).

B. Configuration 2

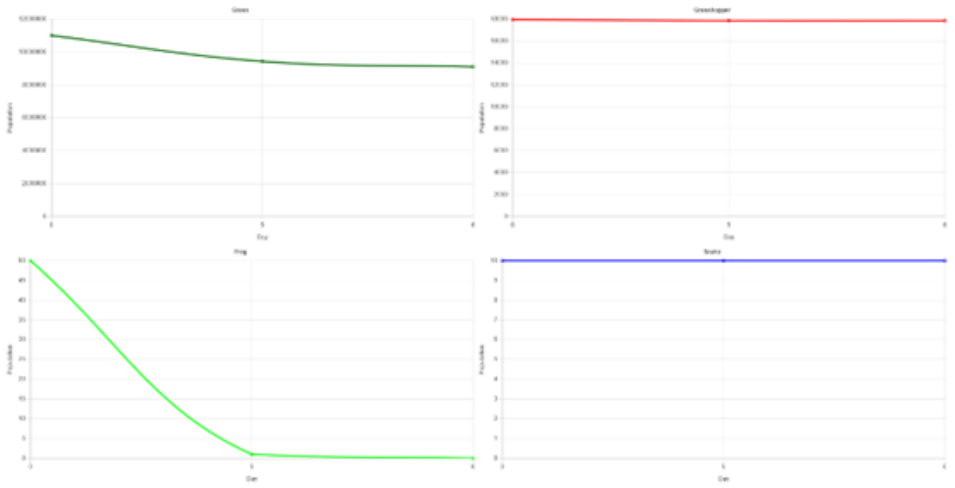

Figure 1: Day six

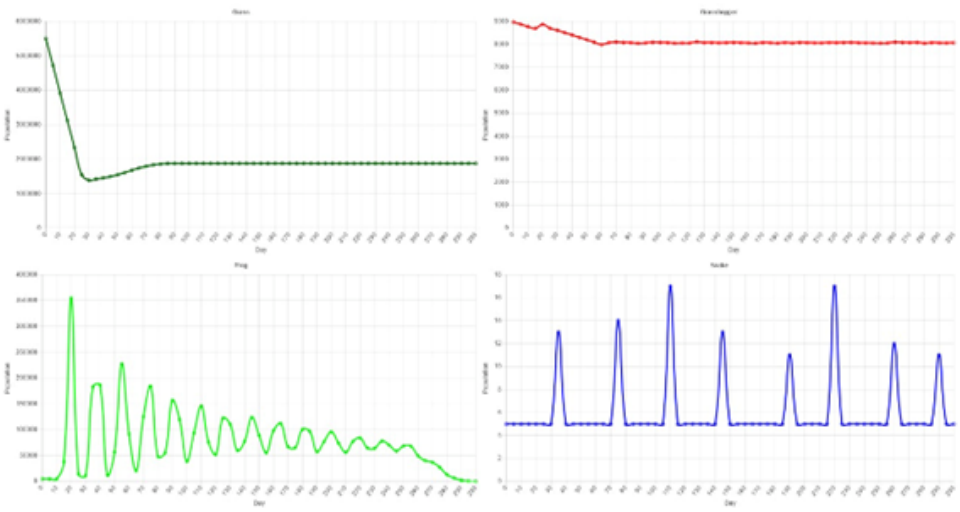

Figure 2: Day 298

The following simulation was run with 10 snakes, 40 frogs per snake, 180 grasshoppers per frog, and 10,000 blades of grass per grasshopper. Therefore, the starting counts were 10 snakes, 400 frogs, 72,000 grasshoppers, and 720,000,000 blades of grass. After 52 simulated days, the frog population went extinct. Given that as day 52 was approaching, the grasshopper population was at an all-time high, it is reasonable to say the cause of the frog extinction was because of too many predators and not due to insufficient prey Fig. 3.

C. Configuration 3

The following simulation was run with 10 snakes, 50 frogs per snake, 150 grasshoppers per frog, and 10,000 blades of grass per grasshopper. Therefore, the starting counts were 10 snakes,

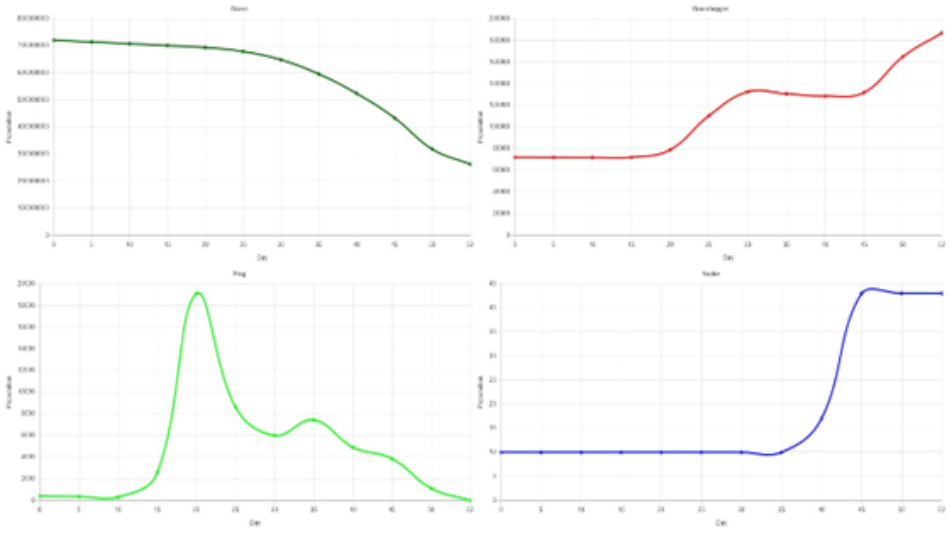

Figure 3: Day 52

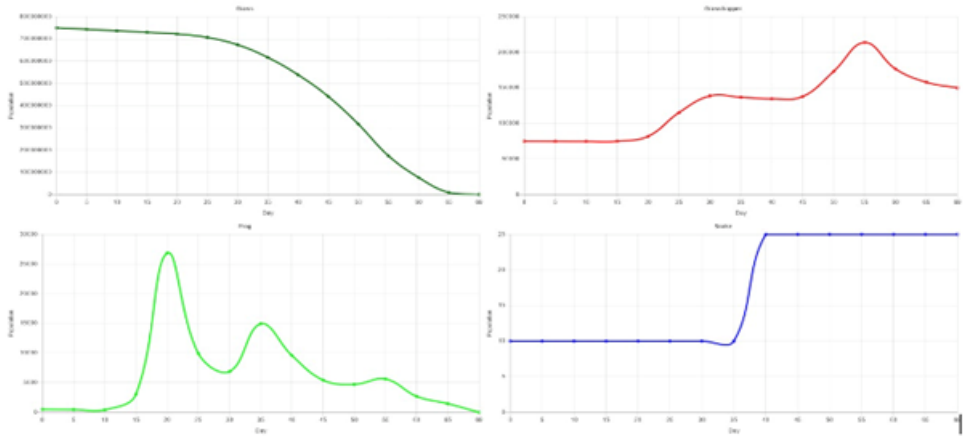

Figure 4: Day 69

500 frogs, 75,000 grasshoppers, and 750,000,000 blades of grass.

After 69 simulated days, both the frogs went extinct, and the environment ran out of grass. While the grass can regrow despite having no fully grown blades left, the population can recover from the blades which have not fully regrown yet. The same can not be said about the frog population, so the simulation terminated. From these results, it seems that the frog population was unable to limit the grasshopper population in order to ensure enough grass is available to sustain the grasshoppers. Like with the first simulation, the frogs went extinct despite having more than enough food, so it is once again reasonable to say the cause of the frog extinction was because of too many predators (Fig. 4).

D. Configuration 4

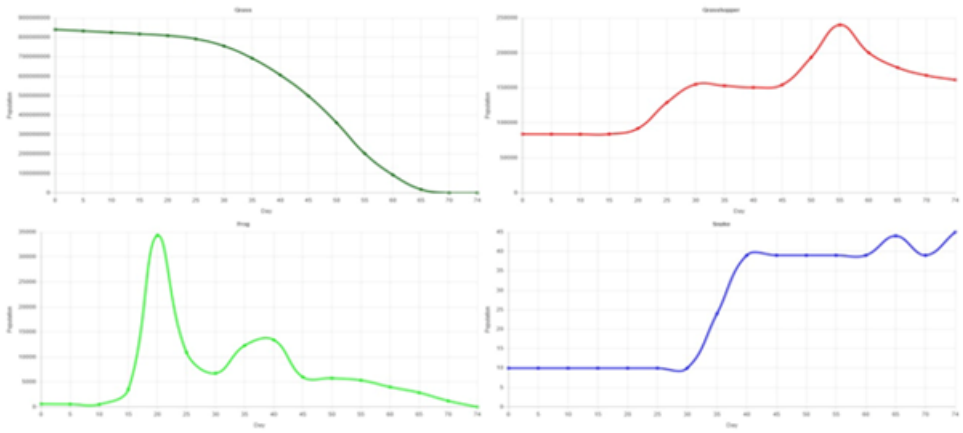

Figure 5: Day 74

The following simulation was run with 10 snakes, 60 frogs per snake, 140 grasshoppers per frog, and 10,000 blades of grass per grasshopper. Therefore, the starting counts were 10 snakes, 600 frogs, 84,000 grasshoppers, and 840,000,000 blades of grass.

Like in the previous experiment, after 74 simulated days both the frog and grass population died out. This indicates the ratio of frogs to snakes, and grass to grasshoppers needs to be increased as shown in Fig. 5.

E. Configuration 5

The following simulation was run with 10 snakes, 80 frogs per snake, 140 grasshoppers per frog, and 30,000 blades of grass per grasshopper. Therefore, the starting counts were 10 grasshoppers, 800 frogs, 112,000 grasshoppers, and 3,360,000,000 blades of grass.

After 84 simulated days, the frog population died out. With each configuration, the length of the simulation increases which indicates that the ratios selected are getting better but are not quite at the ideal values to create a stable ecosystem (Fig. 6).

Experiments using a wide range of ratios have failed due to at least one species going extinct. Interestingly, in most cases it was the frogs who died out. After a close inspection of the graphs and data, it seems that after some time, the frogs get eaten by the snakes at a faster rate than they can reproduce which causes their population to die out. The grasshopper and grass populations almost always fluctuate and then stabilize. If the snake population is able to make it to its first gestation period, that population always seems to stabilize too. It is the frog population which is almost always volatile despite the ratios that start with. This is likely due to the random nature of the simulation. If every random number generator returned its mean value each time, then the second simulation from section 3.1.1 would have been stable. Unfortunately, the generators did not. In that simulation, the generators would have generated ideal numbers but at some point, they would have stopped producing ideal numbers which leads to the decline of the frog population.

4 Long-term sustainability

We can find the average number of animals required to sustain their predators for a long-term period. However, running programs using the calculated numbers may not cause a stable

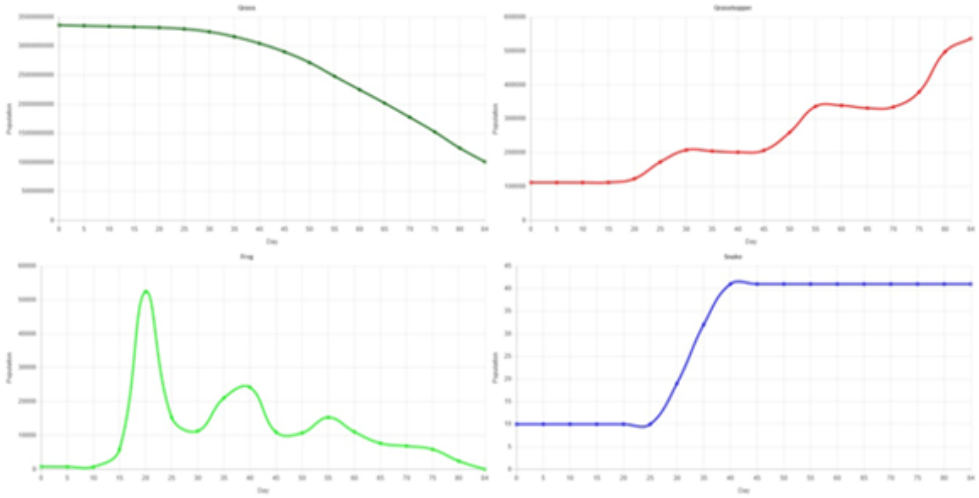

Figure 6: Day 84

ecosystem due to the length of the simulation, volatility and variance involved. Calculating the ratios also assumes all children are able to survive to adulthood. If a predator eats a parent with children in its care, then the children will be unable to feed themselves and will most likely starve. In this section, we show the steps of the calculation.

The first step is to calculate the calorie requirements of the species per day. The mass of each animal is the same in the first generation. Due to the evolutionary nature of the simulation, subsequent generations will not have this constant mass. In addition to the mass, a multiplier is also needed to use the resting energy requirement into a daily energy requirement. The simulation uses a uniformly distributed random number between 2 and 5. So the expected multiplier can therefore be calculated as 3.5. From there, the expected daily calories can be calculated as follows.

$$\text{Daily Calories} = 3.5 * 70(\text{weight})^{\frac{3}{4}} = 245(\text{weight})^{\frac{3}{4}}. \tag{2}$$

Next, the number of calories gained from eating its prey is needed. The calories gained per kilogram for the prey is constant for all animals of a species. As with the predator, the mass of the prey is constant in the first generation but evolves in subsequent generations. Equation (2) is used for the calculation. If the prey in question is grass, then the algorithm would forgo the formula and use the 0.01815 calories per blade calculated in section 2.

$$\text{Calorie Gain} = \text{Calories Per Kg} * (\text{Prey weight}). \tag{3}$$

Finally, the number of preys which must be consumed to fill the daily calorie requirement can be calculated with the following formula.

$$\text{PreyPerDay} = \frac{\text{Daily Calorie}}{\text{Calorie Gain}}. \tag{4}$$

Equation (4) implies that only 27 grasshoppers are required to sustain a frog. This simply calculates the number of grasshoppers which the frog will consume per day. To create a sustainable ecosystem, the number of grasshoppers needs to grow at a rate of 27 adults per day to

sustain the average frog. The 27 will be eaten and the rest will need to produce the 27 which will be consumed on the next day. This is where calculation becomes trickier. Assuming every alpha animal procreates as soon as its children leave its care, the average children raised per day can be calculated as the following:

$$\text{ChildrenPerDayPerAnimal} = \frac{\mu * \text{children per procreation}}{\max(\text{time to adulthood}, \mu * \text{gestation})}. \tag{5}$$

The time it takes a child to grow to adulthood is constant. Both the children per procreation and gestation periods are normally distributed random numbers, so the expected value can be used for the purposes of calculation. The maximum is used in the denominator because both the children need to grow to adulthood, and the gestation period needs to pass. To find the average number of preys to support a single predator, a number is needed to multiply Children Per Day Per Animal so that it equals Prey Per Day. This can be rearranged to isolate that multiplier to obtain:

$$\text{PreyNeeded} = \frac{\text{PreyPerDay}}{\text{ChildrenPerDayPerAnimal}}. \tag{6}$$

From this model, the ideal ratios can be calculated as shown in Table 2. All numbers have been rounded up to the nearest integer. Note that because the probability of a new grasshopper being an alpha is the same as it is a beta, both of which have a different mass, the average of both masses with the value of 0.00021 was used in the calculation.

Animal	Grass	Grasshopper	Frog	Snake
Amount to sustain one predator	613	359	5	N/A

Table 2: Animal numbers to sustain one predator

5 Limitations and future work

Comparing to theoretical approaches, simulation programs take a considerable amount of time to be run. If optimizations could be made to the simulation to run faster, algorithms like Canonical ES could be used to identify the ideal ratios through evolution learning and eventually converge on the perfect set of ratios to run a long simulation. The accurate ratio is also affected by the random number generators.

Optimizations in both the algorithm and implementation will allow for more species, and the possible simulation of a complex interconnected food web where each predator can eat more than one species. These food web simulations will allow for models which more closely reflect the real world. While the current algorithm does allow for multiple food sources of a species, it would only consider the second species as a food option if it was unable to eat anything of the first species. This is also the reason why the hawk was excluded from the simulation. The ecosystem would have required a large number of snakes to sustain 10 hawks. The large number of snakes would require an even larger number of frogs, which propagates down to the individual blades of grass.

As more information about calorie requirements of animals becomes available, the simulation could be improved to reflect that and more closely model the real world.

References

[1] R. Ahmed, M. Rafaqat, I. Siddique and M. A. Arefin: Complex dynamics and chaos control of a discrete-time Predator-Prey model, Discrete Dynamics in Nature and Society, **2023**, Article ID 8873611 (2023). https://doi.org/10.1155/2023/8873611.

[2] G. Bunin: Ecological communities with Lotka-Volterra dynamics, Physical Review E. **95**, (2017) 042414, pp. 1-8, https://journals.aps.org/pre/pdf/10.1103/PhysRevE.95.042414.

[3] P. Chrabaszcz, I. Loshchilov, F. Hutter, Back to Basics: Benchmarking Canonical Evolution Strategies for Playing Atari, Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), Stockholm Sweden, July 2018. pp. 1419–1426. https://ml.informatik.uni-freiburg.de/wp-content/uploads/papers/18-IJCAI-B2B.pdf.

[4] M. Kumar, A. K. Rawat, A. Sonkar, A. Kumar, A. Azhar, M. Kumar and . M. A. Khan, Pellet Biochar: An environmental remedy, editors: V. S. Parmar, P. Malhotra and D. Mathur, Green chemistry in environmental sustainability and chemical education, Proceedings of ICGC 2016, New Delhi. pp. 73–80. Springer.

[5] E. Łuszczki, P. Jagielski, A. Bartosiewicz, K. Dereń, P. Matłosz, M. Kuchciak, Ł, Oleksy, A. Stolarczyk and A. Mazur: Development and validation of new predictive equations for resting energy expenditure in physically active boys, Sci. Rep **13**, 4527 (2023), https://doi.org/10.1038/s41598-023-31661-1.

[6] M. D. Mifflin, S. T. St Jeor, L. A. Hill, B. J. Scott, S. A. Daugherty, Y. O. Koh: A new predictive equation for resting energy expenditure in healthy individuals, Am J Clin Nutr. 1990 **51** (2):241-7. doi: 10.1093/ajcn/51.2.241. PMID: 2305711.

[7] C. O'Reilly: Grass growth rates and pasture management, Field Crop News, (2022), https://fieldcropnews.com/2022/07/grass-growth-rates-and-pasture-management/.

[8] L. D. Plank and G. L. Hill, Chapter 134 - Energy requirement and consumption in the critically ill patient, editor(s): C. Ronco, R. Bellomo, J. A. Kellum, Critical Care Nephrology (Second Edition), W.B. Saunders, 2009, pp. 697-702, ISBN 9781416042525, https://doi.org/10.1016/B978-1-4160-4252-5.50140-4.

[9] X. Wang, J. Cheng, and L. Wang: A reinforcement learning-based predator-prey model, Ecological Complexity, **42**, (2020), 100815, pp. 1-8. https://doi.org/10.1016/j.ecocom.2020.100815.

[10] L. Zhang, G. Zhang and W. Feng: Turing instability generated from discrete diffusion-migration systems, Canadian Applied Mathematics Quarterly, 20(2) (2012), 253–269.

[11] L. Zhang and W. Feng: Exponential and fractional discrete models for a two-innovation diffusion system, Dynamics of Continuous, Discrete and Impulsive Systems Series B: Applications & Algorithms 20 (2013), 103–115.