Modeling Agonistic Character Displacement in Meadowlarks

Nahom Woldehawariat

Advisor: Dr. Kenichi Okamoto

# Introduction.

The evolution of different signaling traits, such as bird song and feather color, in similar bird species can be attributed to a variety of factors, including mate selection and territorial defense. These traits serve as signals that allow birds to communicate with each other and make important decisions about mating and territorial interactions. I have been working on extending this computational model to a specific question: How will two different signaling traits (song and feather colors) each evolve to solve two different problems meadowlarks face? How will the two species of meadowlarks evolve to use each signaling trait to determine? Who they can potentially mate with, and who can they share territories with?  So, through coding and model analyses, I will generate hypotheses about biological evolution in Eastern and western meadowlark. This model is a part of the bigger ACD model that we are working with.

A bird with its mouth open

Description automatically generated

Figure 1: Western Meadowlark (Sturnella neglecta)



Figure 2: Eastern Meadowlark (Sturnella magna)

# Methodology and Background

* 1. The computational Model

Dr Okamoto’s group has built a computer model in CUDA/C++ to explore the biological evolution of signaling traits in different bird species. We are in the process of applying this model to meadowlarks using real-world data. I have working on the part of the code base known as ACD birds. We make use of GPU parallelization in our computational models. GPU parallel computing is a type of computation in which many calculations or processes are carried out simultaneously. This is known as **parallel processing,** and it is what makes GPUs so fast and efficient. Cude is the coding language that makes use of this method. Using Nvidia GPUs we are able to make our computations much faster.

1.2 Territorial interaction and recognition functor

One part of code base is the territorial interaction functor. The functor that simulates territorial interactions between birds based on their trait values, such as feather color and song type. The functor is part of a larger program that models the evolution of competitor recognition and animal behavior. The functor takes several parameters, such as the recognition phenotypes, the recognition function mean and width, the male fighting ability, the indices of males in territory, the territory individuals hold, the species IDs, the fighting ability penalty, the fight outcome modifier, the demes, and the IDs of the birds.

The functor also uses two functions: draw\_bernoulli and bivariate\_normal. The draw\_bernoulli function returns a random binary outcome based on a probability and a random number. The bivariate\_normal function returns the probability density of a bivariate normal distribution based on two variables and their parameters.

The functor operates on a tuple that contains the territory index, the male attempting to gain access to the territory, the number of males occupying the territory, the cumulative number of males in the territory resulting from an exclusive scan, and a random number of seed. The functor performs several steps to simulate a territorial interaction:

* If the territory is unoccupied, the invading male gets to occupy it.
* If the territory is occupied, the invading male must fight with each resident male in turn until either he wins or loses.
* For each fight, the functor calculates the probability of recognition for both the resident and the invader based on their trait values and their recognition function parameters. The probability of recognition is calculated using the bivariate normal function.
* The functor also calculates the fighting ability and the win probability for both the resident and the invader based on their male fighting ability, their species IDs, their demes, and their fight outcome modifier.
* The functor then draws a random outcome for each fight based on the win probabilities using the draw Bernoulli function.

 Depending on the outcome and the recognition probabilities, there are four possible scenarios: Both birds recognize each other as competitors: The winner gets to occupy or keep the territory, and the loser becomes territory less. Scenario one: Both birds lose some fighting ability due to injury or exhaustion. Scenario two: only one bird recognizes the other as a competitor: The recognizer wins by default and gets to occupy or keep.

Table 1:Territorial interaction spreadsheet description

A table with text and numbers

Description automatically generated with medium confidence

The table above is showing how the territorial interaction is carried out in our model. Basically, acting like a spreadsheet with column of male birds with id, another column of territory and each territory with a list of occupants. Implementing this concept into Cuda and C++ and creating the model then the simulation is our goal.

1.3 Bivariate normality distribution.

Equation 1:Bivariate normal formula

A math equations on a white background

Description automatically generated

The bivariate normal distribution is a probability distribution that characterizes the joint behavior of two continuous random variables, often denoted as "x" and "y." In our case the two variables would be the song type and feather color. This distribution extends the concept of the univariate normal distribution to two dimensions. The formula you provided represents the probability density of observing specific values "x" and "y" within this distribution, considering their respective means ("Mu\_X" and "Mu\_Y"), the correlation coefficient ("p") indicating the relationship between the variables, and normalization factors. It encapsulates how the variables are interrelated, offering a mathematical framework to model complex dependencies between two measured characteristics. We have implemented the bivariate normality in Cuda/C++ using the thrust library.

A screenshot of a computer program

Description automatically generated

Code 1: Snapshot of Cuda implementation of Bivariate normal

1.4 How does Bivariate and Bernoulli work in our model

In our computational model, we explore the probability of successful recognition between two birds, a critical factor in their interactions. This probability, denoted as P(recognition), signifies the likelihood of recognition occurring. To represent this, we use a Bernoulli distribution - a binary model for success/failure events.

However, what's fascinating is that P(recognition) itself is influenced by various factors such as recognition phenotypes and function parameters of both birds. To capture this complexity, we utilize a bivariate normal distribution. This modeling choice signifies that P(recognition**)** is not just a single value, but a distribution accounting for the joint behavior of these factors. In mathematical terms, we express this as **P ~ bivariate(MU, Z, Sigma),** where MU is the mean vector, Z is the standardization vector, and Sigma is the covariance matrix. By embracing the bivariate normal distribution, our model effectively handles the intricate interplay between factors, enhancing our understanding of the nuanced dynamics in recognition probabilities."

In our simulation model, we employ the Bernoulli distribution to determine outcomes during the territorial interactions between birds. This approach might raise questions, especially considering our use of the bivariate normal distribution elsewhere in the model. To address any potential confusion, let's delve into the rationale behind this choice.

If you can represent this as a sequence of formulas, you might be able to avoid verbal confusion. So, for instance: P(x, MuY) = Pr(an individual with phenotype X is recognized by an individual with neurotemplate Mu\_Y) = Bernoulli[BV(X, Mu\_Y)] So the idea here is you are dealing with a nested function Bernoulli[BV(X, Mu\_y)] with BV() being the bivariate normal. Under this formulation, 0 <= P(x, MuY) <= 1, so it's kind of like the f∘g(x,y) notation mathematicians are more familiar with. Or something like h(x,y) = f(g(x,y)), in this case with g(x,y) being the bivariate normal and f( ) being the Bernoulli distribution.

1.4 Configuration File

The configuration file serves as a set of instructions and parameters that define the parameters and behavior of the simulation. Here's a summarized overview of the key components and steps discussed:

1. **Overall, Purpose of Configuration File:**
   * The configuration file provides instructions for running a simulation with various parameters and settings. It allows you to define the number of replicates, the number of teams, and the genetic and phenotypic parameters.
2. **Basic Structure:**
   * It is structured as a plain text file, often in a specific format. It consists of sections, each specifying different aspects of the simulation.
3. **Simulation Parameters:**
   * The configuration file includes parameters such as the number of replicates (simulated repetitions), the number of teams, the number of genes (loci), and other simulation-related details.
4. **Phenotype Definitions:**
   * Phenotype definitions include various characteristics or traits of the simulated organisms. Each phenotype is associated with parameters, such as a constant value, species modifiers, coefficients, and environmental noise.
5. **Genotype-Phenotype Mapping:**
   * A genotype-phenotype map defines the relationship between an organism's genotype and its resulting phenotype. Phenotype values are calculated using formulas that involve parameters, constants, coefficients, and species modifiers.
6. **Modifying Configuration:**
   * The goal is to create a configuration file generator that automates the process of creating the configuration file.
   * The generator should allow us to easily modify the number of teams, replicates, and other parameters.
7. **Running the Simulation:**
   * Once the configuration file is prepared, we can run the simulation program with the specified parameters. The simulation program reads the configuration file and conducts the simulation based on the instructions provided.

Also remembering that the specific details and syntax of the configuration file will depend on the simulation program we are using and the format it requires. It's important to ensure that your configuration file is correctly structured and contains accurate parameter values for a successful simulation run. Once we have specified the necessary parameters, we will be able to create the simulation.

Conclusion

As of right now, we don’t have a simulation as we are still working with the config file. Over the fall we be working on specifying the parameters of the config file and generating our simulation. After that, we will be able to create the result from the simulation and do ome analysis on the results.

References

1. Okamoto, K. W. Design Concepts and Details(ODD).

2. ACD Meadowlarks Dr. Kenichi Okamoto

The concept and information I was guided through by Dr. Kenichi Okamoto.Bottom of Form