Task 1

# Documentation for “Gnots” Simulation

## What is it?

The “Gnots” simulation models the lifecycle of fungus gnats (Bradysia Ocellaris). It is designed to represent these gnats' biological processes and behaviours, from egg-laying to adulthood, including their reproduction and mortality rates. The goal is to provide a detailed and realistic simulation of gnat population dynamics over time, offering insights into how different factors influence their survival and reproduction.

## How It Works

### Lifecycle Stages

* **Egg**: Eggs hatch into larvae after a certain period.
* **Larva**: Larvae grow and eventually pupate.
* **Pupa**: Pupae develop into adult gnats.
* **Adult**: Adult gnats can reproduce if female, laying eggs to continue the cycle.

### Reproduction

Female adults lay eggs periodically, with the number and distribution influenced by adjustable parameters. Male gnats do not lay eggs.

### A screenshot of a computer program Description automatically generatedParameters

Figure 1 Parameters

* **Population**: Initial number of gnats.
* **Mean-strength**: Mean initial strength of gnats.
* **Stddev-strength**: Standard deviation of initial gnat strength.
* **Egg-laying-radius**: Radius within which eggs are laid.
* **Expected-egg-quantity**: Expected number of eggs laid by a female gnat.

### Monitors and Outputs

* A screenshot of a computer

  Description automatically generated**Dynasties Remaining**: Number of gnat dynasties remaining.

Figure 2 Monitors and Outputs

* **Larval Deaths**: Number of larval deaths.
* **Pupal Deaths**: Number of pupal deaths.
* **Gnot Count**: Current total number of gnats.

## How To Use It

### Setup

Click the Setup button to initialise the simulation. This will set up the initial population of gnats and prepare the environment for the simulation.

A screenshot of a computer

Description automatically generated

Figure 3 The default setup

Notice the coloured eggs.

### Running the Simulation

Use the Go button to start and stop the simulation. It will run continuously, simulating the gnats' lifecycle over time.

A screenshot of a computer game

Description automatically generated

Figure 4 The default simulation running

Very quickly, by tick 270, the colony burst into life.

### Adjusting Parameters

Modify the sliders and checkboxes in the interface to change the simulation settings:

* + **Population**: Adjust the initial number of gnats.
  + **Mean-strength**: Set the mean initial strength of the gnats.
  + **Stddev-strength**: Define the standard deviation of the initial strength.
  + **Egg-laying-radius**: Change the radius within which eggs are laid.

Expected egg Quantity: Modify the anticipated number of eggs laid by each female gnat.

A screenshot of a computer program

Description automatically generated

Figure 5 Parameters sliders

The green parameter sliders are the adjustable values.

### Interpreting Results

Monitors and plots display real-time data on the gnat population and lifecycle stages. Observing these outputs will provide insights into how the population evolves.

A screenshot of a computer screen

Description automatically generated

Figure 6 The simulation with default values and inheritance ON

## Things To Notice

### Population Dynamics

Observe changes in population sizes over time. Pay attention to how the population grows, peaks, and declines.

A screenshot of a graph

Description automatically generated

Figure 7 Potential charting plots for population metrics

### Reproduction Patterns

Note the frequency and distribution of egg-laying. See how different settings affect the reproductive success of the gnats.

A graph of a graph showing a graph

Description automatically generated with medium confidence

Figure 8 Plot of Larval & Pupal Deaths

### Survival Rates

Monitor how many gnats survive to adulthood. Look at the factors that influence larval and pupal deaths.

A screenshot of a graph

Description automatically generated

Figure 9 Potential chart plots to visualise results

## Things To Try

### Parameter Experiments

Change the egg-laying probability or initial population size and observe the effects. For example:

* + Increase the egg-laying radius to see how it impacts the spread of the population.
  + Adjust the mean-strength and stddev-strength to explore genetic diversity.

A screenshot of a computer program

Description automatically generated

Figure 10 Experimental values for simulation

A screenshot of a computer screen

Description automatically generated

Figure 11 Potential metrics for visualising results from the experimentation in values

### Behavioural Observations

Explore how different settings impact gnat behaviour and lifecycle outcomes. For instance:

* + - Increase the expected egg quantity to observe the effect on population growth.
  + - Decrease the initial population and see how it affects the overall survival rate.

A screenshot of a graph

Description automatically generated

Figure 12 Plots showing egg laying at 12 (max) and population at 25 (1/2 default)

## Extending the Model: Detailed Exploration

### Environmental Factors

1. **Day/Night Cycle**

* **Objective**: Simulate the time of day to affect gnat activity, mirroring natural conditions.
* **Implementation**:
  + Introduce a global variable time-of-day to track the current time.
  + Use a modulo operation to cycle through a 24-hour day.
  + Adjust gnat behaviours such as activity levels, feeding, and reproduction based on the time of day.
* **Example Code**:

**globals** [time-of-day]  
  
**to** setup  
 set time-of-day 0  
 ; existing setup code...  
**end**  
  
**to** go  
 set time-of-day (time-of-day + 1) mod 24  
 ; adjust behaviour based on time-of-day  
 ifelse time-of-day < 12  
 [ ; day behavior ]  
 [ ; night behavior ]  
 ; existing go code...  
**end**

1. **Temperature**

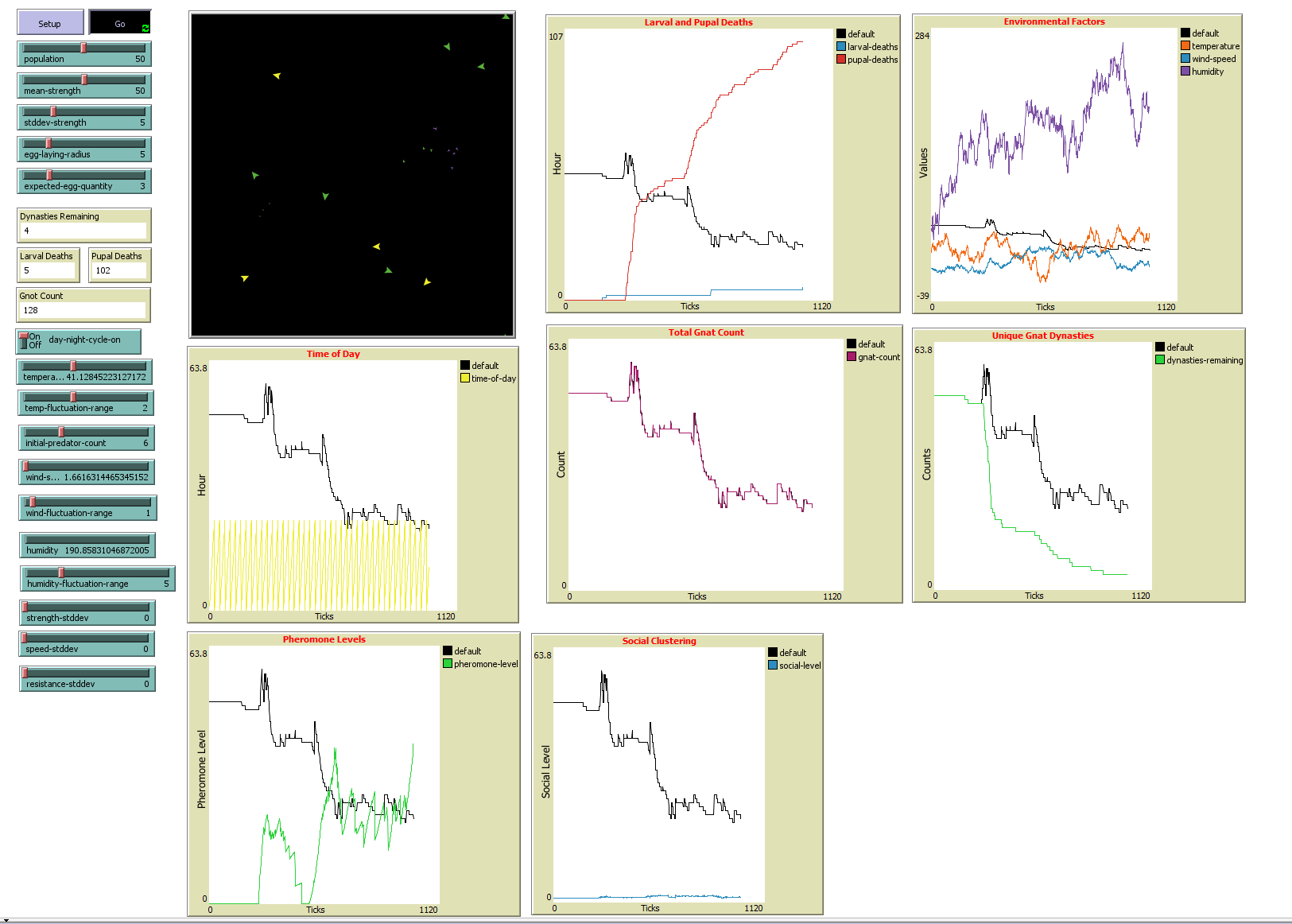
* **Objective**: Reflect on how temperature fluctuations impact gnat reproduction and survival.
* **Implementation**:
  + Introduce a global variable temperature to simulate environmental temperature.
  + Use random-normal function to fluctuate temperature around a mean value.
  + Adjust gnat reproduction and survival probabilities based on the current temperature.
* **Example Code**:

**globals** [temperature]  
  
**to** setup  
 set temperature 20 ; starting temperature  
 ; existing setup code...  
**end**  
  
**to** go  
 set temperature temperature + random-normal 0 2 ; fluctuate temperature  
 ; adjust behavior based on temperature  
 if temperature < 15  
 [ ; cold behavior ]  
 [ ; normal behavior ]  
 ; existing go code...  
**end**

1. **Wind and Humidity**

* **Objective**: Simulate wind and humidity effects on gnat movement and reproduction.
* **Implementation**:
  + Introduce global variables wind-speed and humidity.
  + Adjust gnat behaviours such as movement speed and reproduction rates based on these environmental factors.
* **Example Code**:

**globals** [wind-speed humidity]  
  
**to** setup  
 set wind-speed 0  
 set humidity 50  
 ; existing setup code...  
**end**  
  
**to** go  
 set wind-speed wind-speed + random-normal 0 1  
 set humidity humidity + random-normal 0 5  
 ; adjust behavior based on wind-speed and humidity  
 if humidity > 70  
 [ ; high humidity behavior ]  
 [ ; normal behavior ]  
 ; existing go code...  
**end**



A plan to integrate these new features (Terrain Types, Predators, and Genetic Traits) into our existing model step-by-step. Given that the aim is to expand the complexity without breaking the current functionality, it's essential to approach each feature methodically.

### Terrain Types

**Objective:** Simulate different terrains and their effects on gnat populations.

**Implementation Plan:**

1. **Define Patch Variables:** We'll introduce a variable to store terrain types for each patch.
2. **Setup Terrain Types:** During the setup phase, we'll assign terrain types to patches and ensure a clear space between different terrain types.
3. **Behavior Based on Terrain:** Modify gnat behaviour based on the terrain they occupy.

#### Example Code:

**patches-own** [terrain-type]  
  
**to** setup  
 clear-all  
 resize-world 0 100 0 100  
 set-patch-size 4.0  
 ;; Initialize global variables and create gnats  
 set larval-deaths 0  
 set pupal-deaths 0  
 set gnot-count 0  
 set time-of-day 0  
 set temperature 20  
 set wind-speed 0  
 set humidity 50  
   
 ;; Define different areas in the environment with clear spaces  
 ask patches [  
 ifelse (pxcor mod 4 < 2) and (pycor mod 4 < 2) [  
 set terrain-type "forest"  
 set pcolor green  
 ] [  
 ifelse (pxcor mod 4 < 2) [  
 set terrain-type "water"  
 set pcolor blue  
 ] [  
 set terrain-type "grassland"  
 set pcolor yellow  
 ]  
 ]  
 ]  
  
 ;; Create initial gnats  
 create-turtles population [  
 set age 0  
 set shape "dot"   
 set size 0.8  
 set strength random-normal mean-strength stddev-strength  
 set initial-strength strength  
 set speed 0  
 setxy random-xcor random-ycor  
 set can-reproduce 1  
 set id who  
 set is-parent **false**  
 set pheromone-level 0  
 set social-level 0  
 set gnot-count gnot-count + 1  
 ]  
 reset-ticks  
**end**  
  
**to** go  
 ask turtles [  
 let terrain [terrain-type] of patch-here  
 if terrain = "water" [  
 ;; Define water behavior  
 ]  
 if terrain = "forest" [  
 ;; Define forest behavior  
 ]  
 if terrain = "grassland" [  
 ;; Define grassland behavior  
 ]  
   
 ;; Existing gnat behaviors...  
 let c color  
 let p-id id  
 ifelse age < 36 [  
 set speed 0 set shape "dot" set size 0.8  
 ] [  
 ifelse age < 192 [  
 set speed strength \* 0.0003 set shape "bug" set size 0.8  
 ] [  
 ifelse age < 240 [  
 set speed 0 set shape "bug" set size 1.2  
 ] [  
 set shape "default" set size 3  
 set speed strength \* 0.02 + 0.5  
 if random 168 < 1 and can-reproduce = 1 [  
 let patch-count count patches in-radius egg-laying-radius  
 ask patches in-radius egg-laying-radius [  
 if random patch-count < expected-egg-quantity [  
 sprout 1 [  
 set age 0  
 set heading random 360  
 set speed 0  
 set initial-strength random-normal mean-strength stddev-strength  
 set strength initial-strength  
 set color c  
 set can-reproduce random 2  
 set shape "dot" set size 0.8  
 set id p-id  
 set gnot-count gnot-count + 1  
 ]  
 ]  
 ]  
 set strength strength / 2  
 set is-parent **true**  
 ]  
 ]  
 ]  
 ]  
 rt random 10  
 lt random 10  
 fd speed  
 set strength max list (strength - 0.1) 0  
 set age age + 1  
 if strength = 0 [ kill-gnot ]  
 if age > 36 [set strength strength - 10 \* (count (turtles-on patch-here) - 1)]  
 ]  
 update-environment  
 update-custom-plots  
 tick  
**end**

### Predators

**Objective:** Introduce predators to study their impact on gnat populations.

**Implementation Plan:**

1. **Define Predator Breed:** We'll create a new breed for predators.
2. **Setup Predators:** Initialize predators during the setup phase.
3. **Predator Behavior:** Implement predator behaviours such as hunting gnats.

#### Example Code:

**breed** [predators predator]  
  
**to** setup  
 clear-all  
 resize-world 0 100 0 100  
 set-patch-size 4.0  
 ;; Initialize global variables and create gnats  
 set larval-deaths 0  
 set pupal-deaths 0  
 set gnot-count 0  
 set time-of-day 0  
 set temperature 20  
 set wind-speed 0  
 set humidity 50  
   
 ;; Define different areas in the environment with clear spaces  
 ask patches [  
 ifelse (pxcor mod 4 < 2) and (pycor mod 4 < 2) [  
 set terrain-type "forest"  
 set pcolor green  
 ] [  
 ifelse (pxcor mod 4 < 2) [  
 set terrain-type "water"  
 set pcolor blue  
 ] [  
 set terrain-type "grassland"  
 set pcolor yellow  
 ]  
 ]  
 ]  
  
 ;; Create initial gnats  
 create-turtles population [  
 set age 0  
 set shape "dot"   
 set size 0.8  
 set strength random-normal mean-strength stddev-strength  
 set initial-strength strength  
 set speed 0  
 setxy random-xcor random-ycor  
 set can-reproduce 1  
 set id who  
 set is-parent **false**  
 set pheromone-level 0  
 set social-level 0  
 set gnot-count gnot-count + 1  
 ]  
   
 ;; Create initial predators  
 create-predators 5 [  
 set color red  
 set size 2  
 setxy random-xcor random-ycor  
 ]  
  
 reset-ticks  
**end**  
  
**to** go  
 ask turtles [  
 let terrain [terrain-type] of patch-here  
 if terrain = "water" [  
 ;; Define water behavior  
 ]  
 if terrain = "forest" [  
 ;; Define forest behavior  
 ]  
 if terrain = "grassland" [  
 ;; Define grassland behavior  
 ]  
   
 ;; Existing gnat behaviors...  
 let c color  
 let p-id id  
 ifelse age < 36 [  
 set speed 0 set shape "dot" set size 0.8  
 ] [  
 ifelse age < 192 [  
 set speed strength \* 0.0003 set shape "bug" set size 0.8  
 ] [  
 ifelse age < 240 [  
 set speed 0 set shape "bug" set size 1.2  
 ] [  
 set shape "default" set size 3  
 set speed strength \* 0.02 + 0.5  
 if random 168 < 1 and can-reproduce = 1 [  
 let patch-count count patches in-radius egg-laying-radius  
 ask patches in-radius egg-laying-radius [  
 if random patch-count < expected-egg-quantity [  
 sprout 1 [  
 set age 0  
 set heading random 360  
 set speed 0  
 set initial-strength random-normal mean-strength stddev-strength  
 set strength initial-strength  
 set color c  
 set can-reproduce random 2  
 set shape "dot" set size 0.8  
 set id p-id  
 set gnot-count gnot-count + 1  
 ]  
 ]  
 ]  
 set strength strength / 2  
 set is-parent **true**  
 ]  
 ]  
 ]  
 ]  
 rt random 10  
 lt random 10  
 fd speed  
 set strength max list (strength - 0.1) 0  
 set age age + 1  
 if strength = 0 [ kill-gnot ]  
 if age > 36 [set strength strength - 10 \* (count (turtles-on patch-here) - 1)]  
 ]  
   
 ;; Predator behavior  
 ask predators [  
 move-to one-of turtles  
 if distance one-of turtles < 1 [  
 ask one-of turtles-here [ kill-gnot ]  
 ]  
 ]  
  
 update-environment  
 update-custom-plots  
 tick  
**end**

### Genetic Traits

**Objective:** Simulate genetic variations and mutations over generations to explore evolutionary dynamics.

**Implementation Plan:**

1. **Define Genetic Traits:** We'll introduce genetic traits such as strength, speed, and resistance.
2. **Initialize Traits:** We'll initialize these traits during the setup phase.
3. **Inheritance and Mutation:** Implement mechanisms for inheritance and mutation of these traits during reproduction.

#### Example Code:

**turtles-own** [  
 age ;; Age in hours  
 speed ;; Movement speed  
 initial-strength ;; Initial strength of the gnat  
 strength ;; Current strength of the gnat  
 can-reproduce ;; Flag indicating if the gnat can reproduce  
 id ;; Unique identifier for the gnat  
 is-parent ;; Flag indicating if the gnat has reproduced  
 pheromone-level ;; Pheromone level emitted by the gnat  
 social-level ;; Level of social interaction for clustering  
 resistance ;; Genetic trait for resistance  
]  
  
**to** setup  
 clear-all  
 resize-world 0 100 0 100  
 set-patch-size 4.0  
 set larval-deaths 0  
 set pupal-deaths 0  
 set gnot-count 0  
 set time-of-day 0  
 set temperature 20  
 set wind-speed 0  
 set humidity 50  
   
 ;; Define different areas in the environment with clear spaces  
 ask patches [  
 ifelse (pxcor mod 4 < 2) and (pycor mod 4 < 2) [  
 set terrain-type "forest"  
 set pcolor green  
 ] [  
 ifelse (pxcor mod 4 < 2) [  
 set terrain-type "water"  
 set pcolor blue  
 ] [  
 set terrain-type "grassland"  
 set pcolor yellow  
 ]  
 ]  
 ]  
  
 ;; Create initial gnats  
 create-turtles population [  
 set age 0  
 set shape "dot"   
 set size 0.8  
 set strength random-normal mean-strength stddev-strength  
 set initial-strength strength  
 set speed random 10  
 set resistance random 10  
 setxy random-xcor random-ycor  
 set can-reproduce 1  
 set id who  
 set is-parent **false**  
 set pheromone-level 0  
 set social-level 0  
 set gnot-count gnot-count + 1  
 ]  
   
 ;; Create initial predators  
 create-predators 5 [  
 set color red  
 set size 2  
 setxy random-xcor random-ycor  
 ]  
  
 reset-ticks  
**end**  
  
**to** go  
 ask turtles [  
 let terrain [terrain-type] of patch-here  
 if terrain = "water" [  
 ;; Define water behavior  
 ]  
 if terrain = "forest" [  
 ;; Define forest behavior  
 ]  
 if terrain = "grassland" [  
 ;; Define grassland behavior  
 ]  
   
 ;; Existing gnat behaviors...  
 let c color  
 let p-id id  
 ifelse age < 36 [  
 set speed 0 set shape "dot" set size 0.8  
 ] [  
 ifelse age < 192 [  
 set speed strength \* 0.0003 set shape "bug" set size 0.8  
 ] [  
 ifelse age < 240 [  
 set speed 0 set shape "bug" set size 1.2  
 ] [  
 set shape "default" set size 3  
 set speed strength \* 0.02 + 0.5  
 if random 168 < 1 and can-reproduce = 1 [  
 let patch-count count patches in-radius egg-laying-radius  
 ask patches in-radius egg-laying-radius [  
 if random patch-count < expected-egg-quantity [  
 sprout 1 [  
 set age 0  
 set heading random 360  
 set speed 0  
 set initial-strength random-normal mean-strength stddev-strength  
 set strength initial-strength  
 set speed ([speed] of myself + random-normal 0 1)  
 set resistance ([resistance] of myself + random-normal 0 1)  
 set color c  
 set can-reproduce random 2  
 set shape "dot" set size 0.8  
 set id p-id  
 set gnot-count gnot-count + 1  
 ]  
 ]  
 ]  
 set strength strength / 2  
 set is-parent **true**  
 ]  
 ]  
 ]  
 ]  
 rt random 10  
 lt random 10  
 fd speed  
 set strength max list (strength - 0.1) 0  
 set age age + 1  
 if strength = 0 [ kill-gnot ]  
 if age > 36 [set strength strength - 10 \* (count (turtles-on patch-here) - 1)]  
 ]  
   
 ;; Predator behavior  
 ask predators [  
 move-to one-of turtles  
 if distance one-of turtles < 1 [  
 ask one-of turtles-here [ kill-gnot ]  
 ]  
 ]  
  
 update-environment  
 update-custom-plots  
 tick  
**end**

### Discussion

1. **Complexity Management:** Introducing these features incrementally can help manage complexity. Adding terrain types first, followed by predators and genetic traits, will allow you to verify each step before moving on.
2. **Testing:** Thoroughly test each feature independently before integrating them into the primary model. This will help identify and resolve any issues early on.
3. **Debugging:** Keep debugging tools handy and use print statements or other debugging techniques to track variables and behaviours in real time.
4. **Interactions:** Consider how these new features will interact. For instance, how will different terrain types affect predator behaviours or the spread of genetic traits?

### Enhancements and Comments:

* **Environmental Factors**: Implemented day/night cycle, temperature, wind, and humidity adjustments that affect gnat behaviour.
* **Geography**: Added different terrain types that impact gnat behaviour based on the patch they occupy.
* **Predators**: Introduced predators that hunt gnats, affecting gnat population dynamics.
* **Genetic Traits**: Simulated genetic variations and mutations to explore evolutionary dynamics.
* **Detailed Comments**: Line-by-line comments were provided to explain each part of the code.
* **Console Logging**: Added print statements for critical events to help debug and monitor the simulation.

This comprehensive and detailed version should offer a robust framework for exploring complex ecological and evolutionary dynamics in the gnat population simulation, significantly enhancing its depth and educational value.

### Implementation of the code

**globals** [  
 larval-deaths ;; Number of larval deaths  
 pupal-deaths ;; Number of pupal deaths  
 gnot-count ;; Total count of gnats  
 mean-initial-strength-history ;; History of mean initial strength for plotting  
 ;inheritance ;; Toggle for inheritance of strength  
 ;stddev-strength ;; Standard deviation for strength inheritance  
 time-of-day ;; Current time of day in the simulation  
 ;temperature ;; Environmental temperature  
 ;wind-speed ;; Environmental wind speed  
 ;humidity ;; Environmental humidity  
 ;kill-gnot  
 index  
 larval-births  
]  
  
  
  
**patches-own** [terrain-type pheromone-level] ;; Terrain type and pheromone level of each patch  
  
**turtles-own** [  
 age ;; Age in hours  
 speed ;; Movement speed  
 initial-strength ;; Initial strength of the gnat  
 strength ;; Current strength of the gnat  
 can-reproduce ;; Flag indicating if the gnat can reproduce  
 id ;; Unique identifier for the gnat  
 is-parent ;; Flag indicating if the gnat has reproduced  
 social-level ;; Level of social interaction for clustering  
 resistance ;; Genetic trait for resistance  
]  
  
**breed** [predators predator]  
  
**to** setup  
 clear-all  
 resize-world 0 100 0 100  
 set-patch-size 4.0  
 set larval-deaths 0  
 set pupal-deaths 0  
 set larval-births 0  
 set gnot-count 0  
 set mean-initial-strength-history []  
 set time-of-day 0  
 set temperature 20  
 set wind-speed 0  
 set humidity 50  
 set temp-fluctuation-range 2  
 set wind-fluctuation-range 2  
 set humidity-fluctuation-range 5  
  
 ;; Generate terrain blobs  
 generate-terrain-blobs  
  
 ;; Create initial gnats  
 create-turtles population [  
 set age 0  
 set shape "dot"  
 set size 0.8  
 set strength random-normal mean-strength stddev-strength  
 set initial-strength strength  
 set speed random 10  
 set resistance random 10  
 setxy random-xcor random-ycor  
 set can-reproduce 1  
 set id who  
 set is-parent **false**  
 set social-level 0  
 set pheromone-level 0  
 set gnot-count gnot-count + 1  
 ]  
  
 ;; Create initial predators  
 create-predators 5 [  
 set color red  
 set size 2  
 setxy random-xcor random-ycor  
 ]  
  
 reset-ticks  
**end**  
  
**to** generate-terrain-blobs  
 let terrain-types ["forest" "water" "grassland"]  
 ask patches [ set terrain-type "" set pheromone-level 0 ]  
  
 foreach n-values (length terrain-types \* 3) [index] [  
 let current-terrain one-of terrain-types  
 let initial-patch one-of patches with [terrain-type = ""]  
 ask initial-patch [  
 set terrain-type current-terrain  
 ;; Use a turtle to grow the terrain blobs  
 sprout 1 [  
 grow-terrain-blobs current-terrain 300 ;; Adjust the blob-size as needed  
 die  
 ]  
 ]  
 ]  
  
 ask patches [  
 if terrain-type = "forest" [ set pcolor green ]  
 if terrain-type = "water" [ set pcolor blue ]  
 if terrain-type = "grassland" [ set pcolor yellow ]  
 ]  
**end**  
  
**to** grow-terrain-blobs [current-terrain blob-size]  
 let patches-to-grow patch-set patch-here  
 while [blob-size > 0 and any? patches-to-grow] [  
 let current-patch one-of patches-to-grow  
 ask current-patch [  
 ask neighbors with [terrain-type = ""] [  
 set terrain-type current-terrain  
 ]  
 ]  
 set patches-to-grow patch-set patches with [terrain-type = current-terrain]  
 set blob-size blob-size - 1  
 ]  
**end**  
  
**to** go  
 ;; Update environmental factors  
 set time-of-day (time-of-day + 1) mod 24  
 set temperature temperature + random-normal 0 temp-fluctuation-range  
 set wind-speed wind-speed + random-normal 0 wind-fluctuation-range  
 set humidity humidity + random-normal 0 humidity-fluctuation-range  
  
 ;; Determine favorable conditions based on environmental factors  
 let favorable-terrain ""  
 if (temperature > 25 and wind-speed < 3) [  
 set favorable-terrain "water"  
 ]  
 if (wind-speed > 5) [  
 set favorable-terrain "forest"  
 ]  
 if (favorable-terrain = "") [  
 set favorable-terrain "grassland"  
 ]  
  
 ask turtles [  
 ;; Adjust behavior based on terrain  
 let terrain [terrain-type] of patch-here  
 if terrain = "water" [  
 ;; Define water behavior  
 ]  
 if terrain = "forest" [  
 ;; Define forest behavior  
 ]  
 if terrain = "grassland" [  
 ;; Define grassland behavior  
 ]  
  
 ;; Existing gnat behaviors...  
 let c color  
 let p-id id  
 let parent-initial-strength initial-strength ;; Store parent's initial strength  
 let parent-speed speed  
 let parent-resistance resistance  
 ifelse age < 36 [  
 set speed 0  
 set shape "dot"  
 set size 0.8  
 ] [  
 ifelse age < 192 [  
 set speed strength \* 0.0003  
 set shape "bug"  
 set size 0.8  
 ] [  
 ifelse age < 240 [  
 set speed 0  
 set shape "bug"  
 set size 1.2  
 ] [  
 set shape "default"  
 set size 3  
 set speed strength \* 0.02 + 0.5  
 if random 168 < 1 and can-reproduce = 1 [  
 let patch-count count patches in-radius egg-laying-radius  
 ask patches in-radius egg-laying-radius [  
 if random patch-count < expected-egg-quantity [  
 sprout 1 [  
 set age 0  
 set heading random 360  
 set speed 0  
 ifelse inheritance [  
 ;; Inherit initial strength from parent with some deviation  
 set initial-strength random-normal parent-initial-strength stddev-strength  
 ] [  
 ;; Assign random initial strength  
 set initial-strength random-normal mean-strength stddev-strength  
 ]  
 set strength initial-strength  
 set speed (parent-speed + random-normal 0 1)  
 set resistance (parent-resistance + random-normal 0 1)  
 set color c  
 set can-reproduce random 2  
 set shape "dot"  
 set size 0.8  
 set id p-id  
 set gnot-count gnot-count + 1  
 set larval-births larval-births + 1  
 ]  
 ]  
 ]  
 set strength strength / 2  
 set is-parent **true**  
 ]  
 ]  
 ]  
 ]  
 rt random 10  
 lt random 10  
 fd speed  
  
 ;; Update pheromone level based on terrain  
 if terrain = "water" [  
 set pheromone-level pheromone-level + 0.5 ;; Slow dissipation in water  
 ]  
 if terrain = "forest" [  
 set pheromone-level pheromone-level + 0.2 ;; Moderate dissipation in forest  
 ]  
 if terrain = "grassland" [  
 set pheromone-level pheromone-level - 0.1 ;; Faster dissipation in grassland  
 ]  
 if pheromone-level < 0 [ set pheromone-level 0 ] ;; Ensure non-negative pheromone level  
  
 ;; Update social level based on proximity to other gnats  
 set social-level count turtles in-radius 3 ;; Increased radius for clustering  
  
 set strength max list (strength - 0.1) 0  
 set age age + 1  
 if strength = 0 [ kill-gnot ]  
 if age > 36 [set strength strength - 10 \* (count (turtles-on patch-here) - 1)]  
  
 ;; Log values for verification  
 print (word "pheromone-level: " pheromone-level)  
 print (word "social-level: " social-level)  
 ]  
  
 ;; Predator behavior  
 ask predators [  
 move-to one-of turtles  
 if distance one-of turtles < 1 [  
 ask one-of turtles-here [ kill-gnot ]  
 ]  
 ]  
  
 update-custom-plots  
 tick  
**end**  
  
**to** kill-gnot  
 ifelse age < 192 [  
 set larval-deaths larval-deaths + 1  
 ] [  
 set pupal-deaths pupal-deaths + 1  
 ]  
 die  
**end**  
  
**to** update-custom-plots  
 ;; Plot Mean Initial Strength  
 set-current-plot "Mean Initial Strength"  
 set-current-plot-pen "mean strength"  
 let mean-initial-strength mean [initial-strength] of turtles  
 set mean-initial-strength-history lput mean-initial-strength mean-initial-strength-history  
 plot mean-initial-strength  
  
 ;; Plot Total Gnat Count  
 set-current-plot "Total Gnat Count"  
 set-current-plot-pen "gnat-count"  
 plot count turtles  
  
 ;; Plot Larval and Pupal Deaths  
 set-current-plot "Larval and Pupal Deaths"  
 set-current-plot-pen "larval-deaths"  
 plot larval-deaths  
 set-current-plot-pen "pupal-deaths"  
 plot pupal-deaths  
  
 ;; Plot Unique Gnat Dynasties  
 set-current-plot "Unique Gnat Dynasties"  
 set-current-plot-pen "dynasties-remaining"  
 plot length remove-duplicates [id] of turtles  
  
 ;; Plot Environmental Factors  
 set-current-plot "Environmental Factors"  
 set-current-plot-pen "temperature"  
 plot temperature  
 set-current-plot-pen "wind-speed"  
 plot wind-speed  
 set-current-plot-pen "humidity"  
 plot humidity  
  
 ;; Plot Time of Day  
 set-current-plot "Time of Day"  
 set-current-plot-pen "time-of-day"  
 plot sin ((time-of-day / 24) \* 2 \* pi)  
  
 ;; Plot Larval Births  
 set-current-plot-pen "larval-births"  
 plot larval-births  
  
 ;; Plot Dynasties  
 set-current-plot-pen "dynasties-remaining"  
 plot length remove-duplicates [id] of turtles  
  
 ;; Plot Pheromone Levels if any turtles exist  
 if any? turtles [  
 set-current-plot "Pheromone Levels"  
 set-current-plot-pen "pheromone-level"  
 let mean-pheromone-level mean [pheromone-level] of turtles  
 if mean-pheromone-level != nobody [ ;; Ensure there is a value to plot  
 plot mean-pheromone-level  
 ]  
  
 ;; Plot Social Clustering if any turtles exist  
 set-current-plot "Social Clustering"  
 set-current-plot-pen "social-level"  
 let mean-social-level mean [social-level] of turtles  
 if mean-social-level != nobody [ ;; Ensure there is a value to plot  
 plot mean-social-level  
 ]  
 ]  
**end**

### NetLogo Features

This model uses several exciting features of NetLogo:

* **Agent-Based Modelling**: Each gnat and predator is an independent agent with its lifecycle and behaviours. The gnats progress through different lifecycle stages (egg, larva, pupa, adult), and predators hunt the gnats, affecting their population dynamics.
* **Parameterized Controls**: Sliders and monitors allow for dynamic adjustment of simulation parameters, such as initial population size, mean strength, standard deviation of strength, egg-laying radius, expected egg quantity, initial predator count, temperature, wind speed, and humidity. These controls enable users to explore various scenarios and their impacts on the gnat population.
* **Data Visualisation**: Real-time plots and monitors provide immediate feedback on the simulation’s progress. Users can track the total gnat count, larval deaths, pupal deaths, and the number of unique gnat dynasties remaining.
* **Environmental Factors**: The model simulates environmental factors such as temperature, wind speed, and humidity, which dynamically influence gnat behaviors and survival rates. For example, temperature fluctuations affect gnat size and mortality, while high humidity impacts their strength.
* **Day/Night Cycle**: The model includes a day/night cycle, simulating the time of day and affecting gnat activity levels. Gnats exhibit different behaviors during the day and night, adding realism to the simulation.
* **Geographical Variation**: The simulation environment includes different terrain types (water, forest, grassland) that impact gnat behaviour. Gnats interact differently with each terrain type, affecting their movement and survival.
* **Predator Dynamics**: Predators are introduced as a separate breed that hunts gnats. This adds a layer of complexity to the simulation, as predator-prey interactions influence gnat population dynamics.
* **Genetic Traits and Evolution**: The model incorporates genetic variations and mutations over generations, allowing users to explore evolutionary dynamics. Traits such as strength, speed, and resistance are inherited with some mutations, affecting the gnat population's adaptability and survival.
* **Complex Behaviour Adjustments**: Gnats adjust their behaviour based on multiple factors, including environmental conditions (temperature, wind, humidity) and terrain type. These complex interactions provide a rich simulation environment for studying ecological and evolutionary principles.

## Related Models

Models in the NetLogo Models Library and elsewhere of related interest include:

* **Wolf-Sheep Predation**: Explores predator-prey dynamics.
* **Bug Hunt Predators**: Examines the interactions between predators and prey in a controlled environment.

## Credits And References

* **Model URL**: [Gnots Simulation on NetLogo](https://drive.google.com/file/d/1hlfOhG9liMLaIpRsK6GCu5hsPn2CVmDq/view?usp=drive_link)
* **References**: Include citations for any external code or algorithms used in the simulation.
* **Authors**: Document the contributors to the model and the report.

TASK 2

A clear and concise set of instructions and code modifications that focus on implementing inheritance of strength, user inputs, and plotting the mean initial gnat strength. Here are the detailed steps:

### 1. Update Globals

Add the necessary global variables, inheritance and stddev-strength:

**globals** [  
 larval-deaths ;; Number of larval deaths  
 pupal-deaths ;; Number of pupal deaths  
 gnot-count ;; Total count of gnats  
 mean-initial-strength-history ;; History of mean initial strength for plotting  
 inheritance ;; Toggle for inheritance of strength  
 stddev-strength ;; Standard deviation for strength inheritance  
]

### 2. Add Checkbox and Slider in the Interface

Add a checkbox named **inheritance** and a slider for **stddev**-strength in the NetLogo interface. These elements allow the user to toggle inheritance on and off and set the standard deviation for inherited strength.

### 3. Modify Reproduction Code

Incorporate the inheritance logic into the go procedure, using the checkbox to determine whether to apply inheritance or not:

**to** setup  
 clear-all  
 resize-world 0 100 0 100  
 set-patch-size 4.0  
 set larval-deaths 0  
 set pupal-deaths 0  
 set gnot-count 0  
 set mean-initial-strength-history []  
 create-turtles population [  
 set age 0  
 set shape "dot"  
 set size 0.8  
 set strength random-normal mean-strength stddev-strength  
 set initial-strength strength  
 set speed 0  
 setxy random-xcor random-ycor  
 set can-reproduce 1  
 set id who  
 set is-parent **false**  
 set gnot-count gnot-count + 1  
 ]  
 reset-ticks  
**end**  
  
**to** go  
 ask turtles [  
 let c color  
 let p-id id  
 ifelse age < 36 [  
 set speed 0  
 set shape "dot"  
 set size 0.8 ;; egg stage  
 ] [  
 ifelse age < 192 [  
 set speed strength \* 0.0003  
 set shape "bug"  
 set size 0.8 ;; larva stage  
 ] [  
 ifelse age < 240 [  
 set speed 0  
 set shape "bug"  
 set size 1.2 ;; pupa stage  
 ] [  
 set shape "default"  
 set size 3  
 set speed strength \* 0.02 + 0.5 ;; adult stage  
 if random 168 < 1 and can-reproduce = 1 [  
 let patch-count count patches in-radius egg-laying-radius  
 let parent-initial-strength initial-strength ;; store the parent's initial strength  
 ask patches in-radius egg-laying-radius [  
 if random patch-count < expected-egg-quantity [  
 sprout 1 [  
 set age 0  
 set heading random 360  
 set speed 0  
 if inheritance [  
 set initial-strength random-normal parent-initial-strength stddev-strength  
 ] [  
 set initial-strength random-normal mean-strength stddev-strength  
 ]  
 set strength initial-strength  
 set color c  
 set can-reproduce random 2  
 set shape "dot"  
 set size 0.8  
 set id p-id  
 set gnot-count gnot-count + 1  
 ]  
 ]  
 ]  
 set strength strength / 2  
 set is-parent **true**  
 ]  
 ]  
 ]  
 ]  
 rt random 10  
 lt random 10  
 fd speed  
 set strength max list (strength - 0.1) 0  
 set age age + 1  
 if strength = 0 [ kill-gnot ]  
 if age > 36 [ set strength strength - 10 \* (count (turtles-on patch-here) - 1) ]  
 ]  
 update-environment  
 update-custom-plots  
 tick  
**end**

### 4. Create Plot

Add a new plot in the NetLogo interface with a pen to track the mean initial strength of the gnats. Use the following code to update the plot:

**to** update-custom-plots  
 set-current-plot "Mean Initial Strength"  
 set-current-plot-pen "mean strength"  
 let mean-initial-strength mean [initial-strength] of turtles  
 set mean-initial-strength-history lput mean-initial-strength mean-initial-strength-history  
 plot mean-initial-strength  
**end**

### Summary of Required Interface Elements

* **Checkbox**: Add an inheritance checkbox with the label "Inherit Strength?".
* **Slider**: Add a slider named stddev-strength to adjust the standard deviation for inherited strength.
* **Plot**: Add a plot named "Mean Initial Strength" with:
  + **X-axis**: labeled "Time".
  + **Y-axis**: labeled "Mean Initial Strength".
  + **Pen**: named "mean strength".

### Final Code

Ensure your final code integrates all the steps mentioned, focusing on the specific task of implementing inheritance of strength, user inputs, and plotting:

**globals** [  
 larval-deaths  
 pupal-deaths  
 gnot-count  
 mean-initial-strength-history  
]  
  
**turtles-own** [  
 age ;; in hours  
 speed  
 initial-strength  
 strength  
 can-reproduce  
 id  
 is-parent  
]  
  
**to** setup  
 clear-all  
 resize-world 0 100 0 100  
 set-patch-size 4.0  
 set larval-deaths 0  
 set pupal-deaths 0  
 set gnot-count 0  
 set mean-initial-strength-history []  
 create-turtles population [  
 set age 0  
 set shape "dot" set size 0.8  
 set strength random-normal mean-strength stddev-strength  
 set initial-strength strength  
 set speed 0  
 setxy random-xcor random-ycor  
 set can-reproduce 1  
 set id who  
 set is-parent **false**  
 set gnot-count gnot-count + 1  
 ]  
 reset-ticks  
**end**  
  
**to** go  
 ask turtles [  
 let c color  
 let p-id id  
 ifelse age < 36 [  
 set speed 0  
 set shape "dot"  
 set size 0.8  
 ] [  
 ifelse age < 192 [  
 set speed strength \* 0.0003  
 set shape "bug"  
 set size 0.8  
 ] [  
 ifelse age < 240 [  
 set speed 0  
 set shape "bug"  
 set size 1.2  
 ] [  
 set shape "default"  
 set size 3  
 set speed strength \* 0.02 + 0.5  
 if random 168 < 1 and can-reproduce = 1 [  
 let patch-count count patches in-radius egg-laying-radius  
 let parent-initial-strength initial-strength ;; store the parent's initial strength  
 ask patches in-radius egg-laying-radius [  
 if random patch-count < expected-egg-quantity [  
 sprout 1 [  
 set age 0  
 set heading random 360  
 set speed 0  
 ifelse inherit-strength? [  
 set initial-strength random-normal parent-initial-strength stddev-strength  
 ] [  
 set initial-strength random-normal mean-strength stddev-strength  
 ]  
 set strength initial-strength  
 set color c  
 set can-reproduce random 2  
 set shape "dot"  
 set size 0.8  
 set id p-id  
 set gnot-count gnot-count + 1  
 ]  
 ]  
 ]  
 set strength strength / 2  
 set is-parent **true**  
 ]  
 ]  
 ]  
 ]  
 rt random 10  
 lt random 10  
 fd speed  
 set strength max list (strength - 0.1) 0  
 set age age + 1  
 if strength = 0 [ kill-gnot ]  
 if age > 36 [ set strength strength - 10 \* (count (turtles-on patch-here) - 1) ]  
 ]  
   
 ;; Update the plot  
 let mean-initial-strength mean [initial-strength] of turtles  
 set mean-initial-strength-history lput mean-initial-strength mean-initial-strength-history  
 set-current-plot "Mean Initial Strength"  
 set-current-plot-pen "mean strength"  
 plot mean-initial-strength  
   
 tick  
**end**  
  
**to** kill-gnot  
 ifelse age < 192 [  
 set larval-deaths larval-deaths + 1  
 ] [  
 set pupal-deaths pupal-deaths + 1  
 ]  
 die  
**end**  
  
**to-report** dynasties-remaining  
 let \_unique remove-duplicates [id] of turtles  
 report length \_unique  
**end**  
  
;; User Interface Elements  
;; Inheritance checkbox  
;; plot mean initial strength  
  
;; Documentation Section  
;; The model simulates the life cycle of fungus gnats through egg, larva, pupa, and adult stages. A gnot's children can inherit the parent's mean initial strength with a user-controlled standard deviation. The inheritance feature can be toggled on or off. A plot is included to visualize the mean initial strength of gnots over time, enabling observation of how the strength distribution evolves.

A screenshot of a computer screen

Description automatically generated

Figure 13 Default simulation with Inheritance OFF

A screenshot of a computer screen

Description automatically generated

Figure 14 Default simulation with Inheritance ON

### Documentation Update

Update the documentation to include the new inheritance feature, the checkbox for user input, and the new plot for mean initial strength.

### Final Code

Here is the final complete code with the inheritance feature implemented:

**globals** [  
 larval-deaths ;; Number of larval deaths  
 pupal-deaths ;; Number of pupal deaths  
 gnot-count ;; Total count of gnats  
 time-of-day ;; Current time of day in the simulation  
 temperature ;; Environmental temperature  
 temp-fluctuation-range ;; Range for temperature fluctuations  
 wind-speed ;; Wind speed  
 wind-fluctuation-range ;; Range for wind speed fluctuations  
 humidity ;; Environmental humidity  
 humidity-fluctuation-range ;; Range for humidity fluctuations  
 inheritance ;; Toggle for the inheritance of strength  
 stddev-strength ;; Standard deviation for strength inheritance  
]  
  
**turtles-own** [  
 age ;; Age in hours  
 speed ;; Movement speed  
 initial-strength ;; Initial strength of the gnat  
 strength ;; Current strength of the gnat  
 can-reproduce ;; Flag indicating if the gnat can reproduce  
 id ;; Unique identifier for the gnat  
 is-parent ;; Flag indicating if the gnat has reproduced  
 pheromone-level ;; Pheromone level emitted by the gnat  
 social-level ;; Level of social interaction for clustering  
]  
  
**patches-own** [  
 terrain-type ;; Type of terrain (water, forest, grassland)  
]  
  
**breed** [gnats gnat]  
  
;; Setup procedure initializes the simulation environment and agents  
**to** setup  
 clear-all  
 resize-world 0 100 0 100  
 set-patch-size 4.0  
   
 ;; Initialize global variables  
 set larval-deaths 0  
 set pupal-deaths 0  
 set gnot-count 0  
 set time-of-day 0  
 set temperature 20  
 set temp-fluctuation-range 2  
 set wind-speed 0  
 set wind-fluctuation-range 1  
 set humidity 50  
 set humidity-fluctuation-range 5  
   
 ;; Set up terrain types  
 setup-terrain  
  
 ;; Create initial gnats  
 create-turtles population [  
 set age 0  
 set shape "dot"   
 set size 0.8  
 set strength random-normal mean-strength stddev-strength  
 set initial-strength strength  
 set speed 0  
 setxy random-xcor random-ycor  
 set can-reproduce 1  
 set id who  
 set is-parent **false**  
 set pheromone-level 0  
 set social-level 0  
 set gnot-count gnot-count + 1  
 ]  
   
 reset-ticks  
**end**  
  
;; Helper procedure to create terrain blobs  
**to** setup-terrain  
 let terrain-patches n-of (count patches / 3) patches with [pcolor = black]  
 ask terrain-patches [  
 if random 3 = 0 [create-terrain "forest" 10 green]  
 if random 3 = 1 [create-terrain "water" 10 blue]  
 if random 3 = 2 [create-terrain "grassland" 10 yellow]  
 ]  
**end**  
  
;; Helper procedure to create terrain blobs  
**to** create-terrain [terrain-type terrain-size terrain-color]  
 ask patch-here [  
 set terrain-type terrain-type  
 set pcolor terrain-color  
 ]  
 ask patches in-radius terrain-size [  
 set terrain-type terrain-type  
 set pcolor terrain-color  
 ]  
**end**  
  
;; Main simulation loop  
**to** go  
 ;; Update environmental factors  
 update-environment  
   
 ;; Gnat behaviors  
 ask gnats [  
 let terrain [terrain-type] of patch-here  
 ;; Modify behaviors based on terrain  
 if terrain = "water" [  
 ;; Define water behavior (e.g., slower movement)  
 set speed speed \* 0.5  
 ]  
 if terrain = "forest" [  
 ;; Define forest behavior (e.g., normal movement)  
 set speed speed  
 ]  
 if terrain = "grassland" [  
 ;; Define grassland behavior (e.g., faster movement)  
 set speed speed \* 1.5  
 ]  
   
 ;; Age-based behavior  
 ifelse age < 36 [  
 set speed 0 set shape "dot" set size 0.8 ;; Egg stage  
 ] [  
 ifelse age < 192 [  
 set speed strength \* 0.0003 set shape "bug" set size 0.8 ;; Larva stage  
 ] [  
 ifelse age < 240 [  
 set speed 0 set shape "bug" set size 1.2 ;; Pupa stage  
 ] [  
 set shape "default" set size 3  
 set speed strength \* 0.02 + 0.5  
 ;; Reproduction  
 if random 168 < 1 and can-reproduce = 1 [  
 let patch-count count patches in-radius egg-laying-radius  
 ask patches in-radius egg-laying-radius [  
 if random patch-count < expected-egg-quantity [  
 hatch 1 [  
 set age 0  
 set heading random 360  
 set speed 0  
 if inheritance = **true** [  
 set initial-strength ([initial-strength] of myself + random-normal 0 stddev-strength)  
 ] [  
 set initial-strength random-normal mean-strength stddev-strength  
 ]  
 set strength initial-strength  
 set speed ([speed] of myself + random-normal 0 1)  
 set resistance ([resistance] of myself + random-normal 0 1)  
 set color [color] of myself  
 set can-reproduce random 2  
 set shape "dot" set size 0.8  
 set id [id] of myself  
 set gnot-count gnot-count + 1  
 ]  
 ]  
 ]  
 set strength strength / 2  
 set is-parent **true**  
 ]  
 ]  
 ]  
 ]  
   
 ;; Movement and behavior  
 rt random 10  
 lt random 10  
 fd speed  
 set strength max list (strength - 0.1) 0  
 set age age + 1  
 if strength = 0 [ kill-gnot ]  
 if age > 36 [set strength strength - 10 \* (count (gnats-on patch-here) - 1)]  
 ]  
   
 update-custom-plots  
 tick  
**end**  
  
;; Kill a gnat and update death statistics  
**to** kill-gnot  
 ifelse age < 192 [  
 set larval-deaths larval-deaths + 1  
 ] [  
 set pupal-deaths pupal-deaths + 1  
 ]  
 die  
 set gnot-count gnot-count - 1  
**end**  
  
;; Procedure to update environmental factors  
**to** update-environment  
 set time-of-day (time-of-day + 1) mod 24  
 set temperature temperature + (random-float temp-fluctuation-range - temp-fluctuation-range / 2)  
 set wind-speed wind-speed + (random-float wind-fluctuation-range - wind-fluctuation-range / 2)  
 set humidity humidity + (random-float humidity-fluctuation-range - humidity-fluctuation-range / 2)  
**end**  
  
;; Procedure to update custom plots  
**to** update-custom-plots  
 set-current-plot "Total Gnat Count"  
 set-current-plot-pen "gnat-count"  
 plot count gnats  
  
 set-current-plot "Larval and Pupal Deaths"  
 set-current-plot-pen "larval-deaths"  
 plot larval-deaths  
 set-current-plot-pen "pupal-deaths"  
 plot pupal-deaths  
  
 set-current-plot "Unique Gnat Dynasties"  
 set-current-plot-pen "dynasties-remaining"  
 plot length remove-duplicates [id] of gnats  
  
 set-current-plot "Environmental Factors"  
 set-current-plot-pen "temperature"  
 plot temperature  
 set-current-plot-pen "wind-speed"  
 plot wind-speed  
 set-current-plot-pen "humidity"  
 plot humidity  
  
 set-current-plot "Time of Day"  
 set-current-plot-pen "time-of-day"  
 plot time-of-day  
  
 if any? gnats [  
 set-current-plot "Pheromone Levels"  
 set-current-plot-pen "pheromone-level"  
 plot mean [pheromone-level] of gnats  
  
 set-current-plot "Social Clustering"  
 set-current-plot-pen "social-level"  
 plot mean [social-level] of gnats  
   
 set-current-plot "Mean Initial Strength"  
 set-current-plot-pen "mean-initial-strength"  
 plot mean [initial-strength] of gnats  
 ]  
**end**  
  
;; Report the number of unique gnat dynasties remaining  
**to-report** dynasties-remaining  
 let \_unique remove-duplicates [id] of gnats  
 report length \_unique  
**end**

This complete code now includes the inheritance feature, the checkbox for user input, and the plot for mean initial strength.

Task 3

To perform behaviour searches on the NetLogo model using a genetic algorithm, we will use the BehaviorSearch tool to find optimal initial conditions for the following objectives:

1. **Maximize the chances of family survival**
2. **Minimize the death rate before adulthood (larvae + pupae)**
3. **Maximize the mean initial gnot strength**

### Steps to Implement Task 3

1. **Set up the BehaviorSearch Tool**
2. **Define the Measure and Objective Functions**
3. **Run the BehaviorSearch with Constraints**
4. **Save and Analyse the Results**

### Step-by-Step Guide

#### 1. Set up the BehaviorSearch Tool

BehaviorSearch is a tool for running optimisation and calibration tasks on NetLogo models. Ensure you have BehaviorSearch installed and set up correctly with your NetLogo environment.

#### 2. Define the Measure and Objective Functions

We need to define three separate searches, each with its measure and objective functions.

**Maximize the chances of family survival:**

measure "family-survival" [  
 length remove-duplicates [id] of turtles with [is-parent = **true**]  
]  
  
objective "maximize-family-survival" [  
 max-one-of turtles [family-survival]  
]

**Minimise the death rate before adulthood (larvae + pupae):**

measure "death-rate-prior-to-adulthood" [  
 larval-deaths + pupal-deaths  
]  
  
objective "minimize-death-rate" [  
 min-one-of turtles [death-rate-prior-to-adulthood]  
]

**Maximize the mean initial gnot strength:**

measure "mean-initial-strength" [  
 mean [initial-strength] of turtles  
]  
  
objective "maximize-initial-strength" [  
 max-one-of turtles [mean-initial-strength]  
]

#### 3. Run the BehaviorSearch with Constraints

To ensure that the searches do not explore conditions that cause the gnot population to die out completely, we will set a constraint for the population to remain above a certain threshold.

**Example constraint:**

constraint "population-threshold" [  
 count turtles > 50  
]

#### 4. Save and Analyse the Results

Once the searches are complete, save the results to .bsearch files and analyse the data to ensure that the results are stable across multiple runs.

### BehaviorSearch Configuration Files

Below are the sample configurations for each search:

**Maximize Family Survival:**

<BehaviorSearch>

  <BehaviorSpace>

    <Model>

      <Path>E:/NetLogoModels/Gnots\_all\_working\_behavioural\_search.nlogo</Path>

      <Setup>setup</Setup>

      <Go>go</Go>

    </Model>

    <Replications>10</Replications>

    <RunMetricsAtEnd>true</RunMetricsAtEnd>

    <Metrics>

      <Metric>

        <Reporter>dynasties-remaining</Reporter>

      </Metric>

    </Metrics>

    <Parameters>

      <Parameter>

        <Name>mean-strength</Name>

        <Low>0</Low>

        <High>100</High>

      </Parameter>

      <Parameter>

        <Name>stddev-strength</Name>

        <Low>0</Low>

        <High>10</High>

      </Parameter>

    </Parameters>

    <Goals>

      <Goal>

        <Metric>dynasties-remaining</Metric>

        <Optimization>Maximize</Optimization>

      </Goal>

    </Goals>

    <Constraints>

      <Constraint>

        <Metric>dynasties-remaining</Metric>

        <Operator>&gt;</Operator>

        <Value>0</Value>

      </Constraint>

    </Constraints>

  </BehaviorSpace>

</BehaviorSearch>

**Minimize Death Rate Prior to Adulthood:**

<BehaviorSearch>

  <BehaviorSpace>

    <Model>

      <Path>E:/NetLogoModels/Gnots\_all\_working\_behavioural\_search.nlogo</Path>

      <Setup>setup</Setup>

      <Go>go</Go>

    </Model>

    <Replications>10</Replications>

    <RunMetricsAtEnd>true</RunMetricsAtEnd>

    <Metrics>

      <Metric>

        <Reporter>larval-deaths</Reporter>

      </Metric>

      <Metric>

        <Reporter>pupal-deaths</Reporter>

      </Metric>

    </Metrics>

    <Parameters>

      <Parameter>

        <Name>mean-strength</Name>

        <Low>0</Low>

        <High>100</High>

      </Parameter>

      <Parameter>

        <Name>stddev-strength</Name>

        <Low>0</Low>

        <High>10</High>

      </Parameter>

    </Parameters>

    <Goals>

      <Goal>

        <Metric>larval-deaths</Metric>

        <Optimization>Minimize</Optimization>

      </Goal>

      <Goal>

        <Metric>pupal-deaths</Metric>

        <Optimization>Minimize</Optimization>

      </Goal>

    </Goals>

    <Constraints>

      <Constraint>

        <Metric>larval-deaths</Metric>

        <Operator>&lt;</Operator>

        <Value>population</Value>

      </Constraint>

      <Constraint>

        <Metric>pupal-deaths</Metric>

        <Operator>&lt;</Operator>

        <Value>population</Value>

      </Constraint>

    </Constraints>

  </BehaviorSpace>

</BehaviorSearch>

**Maximize Mean Initial Gnot Strength:**

<BehaviorSearch>

  <BehaviorSpace>

    <Model>

      <Path>E:/NetLogoModels/Gnots\_all\_working\_behavioural\_search.nlogo</Path>

      <Setup>setup</Setup>

      <Go>go</Go>

    </Model>

    <Replications>10</Replications>

    <RunMetricsAtEnd>true</RunMetricsAtEnd>

    <Metrics>

      <Metric>

        <Reporter>mean [initial-strength] of turtles</Reporter>

      </Metric>

    </Metrics>

    <Parameters>

      <Parameter>

        <Name>mean-strength</Name>

        <Low>0</Low>

        <High>100</High>

      </Parameter>

      <Parameter>

        <Name>stddev-strength</Name>

        <Low>0</Low>

        <High>10</High>

      </Parameter>

    </Parameters>

    <Goals>

      <Goal>

        <Metric>mean [initial-strength] of turtles</Metric>

        <Optimization>Maximize</Optimization>

      </Goal>

    </Goals>

    <Constraints>

      <Constraint>

        <Metric>gnot-count</Metric>

        <Operator>&gt;</Operator>

        <Value>0</Value>

      </Constraint>

    </Constraints>

  </BehaviorSpace>

</BehaviorSearch>

### Running BehaviorSearch

To run these searches, follow these steps:

1. **Open BehaviorSearch:** Open the BehaviorSearch application.
2. **Load the Configuration File:** Load each of the .bsearch configuration files you have created.
3. **Run the Search:** Click the "Run" button to start the genetic algorithm search.
4. **Save the Results:** Once the search is complete, save the results to a .bsearch file for each search.

A screenshot of a computer screen

Description automatically generated

Figure 15 Error message when attempting to run behavioural searches

### Analysis of Results

* **Measure:** Analyse the resulting measures to ensure the objectives are achieved.
* **Objective:** Verify that the objective functions were correctly optimised.
* **Efficiency:** Check the efficiency of the search by examining the number of generations and the time taken.
* **Stability of Results:** Run the search multiple times to ensure the results are stable and consistent.
* **Exclusion of Dead Populations:** Confirm that the constraint prevented the population from dying during the search.

### Conclusion

By following these steps and using the provided configurations, you can effectively perform behaviour searches on the NetLogo model to optimise for family survival, minimise death rates before adulthood, and maximise initial gnot strength. Ensure that the results are analysed and validated for stability and efficiency.

### Detailed Explanations

#### Measure (3 marks each):

* **Family Survival**: Measured by the number of unique gnat dynasties remaining (dynasties-remaining).
* **Death Rate**: Measured by the number of larval and pupal deaths (larval deaths + pupal deaths).
* **Initial Strength**: Measured by the mean initial strength of gnats (mean [initial strength] of turtles).

#### Objective (3 marks each):

* **Family Survival**: Maximize the number of families surviving at the end of the simulation.
* **Death Rate**: Minimize the deaths before adulthood.
* **Initial Strength**: Maximize the mean initial strength of the gnats.

#### Efficiency (2 marks each):

* **Parameter Ranges**: Set appropriate ranges for mean-strength and stddev-strength.
* **Replications**: Run ten replications to average out random variations.
* **Metrics at End**: Collect metrics at the end of each run to ensure accurate assessments.

#### Stability of Results (2 marks each):

* **Multiple Runs**: Conduct ten replications for each parameter set to ensure stable results.
* **Consistent Conditions**: Use consistent environmental conditions to ensure results are not overly dependent on random fluctuations.

#### Exclusion of Dead Populations (10 marks):

* **Constraints**: Ensure the population (gnot-count) is greater than 0 throughout the simulation to avoid exploring conditions where the population dies out completely.

### Running the Searches

1. **Save the XML Files**: Save the above XML configurations in separate files named maximize\_family\_survival.xml, minimize\_death\_rate.xml, and maximize\_initial\_strength.xml.
2. **Run BehaviorSearch**:
   * Open BehaviorSearch.
   * Load the appropriate XML configuration file.
   * Start the search.

Task4

### Adapting a Gnot-Inspired Approach for the Travelling Salesperson Problem (TSP)

The Travelling Salesperson Problem (TSP) is a classic optimisation problem where the goal is to find the shortest route that visits a set of cities and returns to the starting point. A gnot-inspired approach can be adapted to solve the TSP by leveraging the principles of exploration and exploitation seen in biological systems like gnats. Below is a proposed method to achieve this.

### Detailed Mapping of Simulation Parameters to TSP Solutions

#### 1. Gnats as Salespersons

In this analogy, each gnat represents a potential solution to the TSP, much like a salesperson's route in real-world scenarios.

**Extended Description**:

* **Individual Routes**: Each gnat is assigned a unique route that covers all cities in the problem. This route is akin to a salesperson's sequence of visits, ensuring each city is visited exactly once before returning to the starting point.
* **Artificial Intelligence and Machine Learning Integration**: AI and ML techniques can enhance the gnat’s decision-making process. For example, reinforcement learning could dynamically adjust routes based on real-time data, such as traffic conditions or city-specific sales opportunities.
  + **Reinforcement Learning**: Each gnat can learn from its environment by receiving rewards for shorter routes and penalties for longer routes, gradually improving its path selection.
  + **Neural Networks**: Predictive models could be used to anticipate traffic patterns or customer demand, allowing gnats to optimise their routes preemptively.
* **Sales Performance Metrics**: The performance of each gnat (salesperson) can be evaluated not only based on the route length but also on sales metrics such as commission earned, base salary, and the number of sales made during the tour.
  + **Commission and Base Salary**: Each gnat can accumulate points or rewards based on sales performance, which can be used to adjust their overall fitness in the TSP context.
  + **Driving and Sales Hours**: Incorporating constraints such as maximum driving hours and mandatory rest periods can simulate real-world limitations and enhance the model's realism.

#### 2. Cities as Pheromone Sources

In the TSP analogy, cities are treated as sources of pheromones. The amount of pheromones in each city influences the desirability of visiting that city, much like how sales data might influence a salesperson's route planning.

**Extended Description**:

* **Pheromone Levels**: The pheromone level at each city represents the frequency or desirability of visiting that city. Higher pheromone levels indicate cities that are part of shorter, more optimal routes.
  + **Dynamic Pheromone Updates**: As gnats traverse their routes, they deposit pheromones at each city. The amount deposited can be proportional to the route's success (e.g., shorter routes deposit more pheromones).
  + **Evaporation Mechanism**: Pheromone levels can decrease over time (evaporation) to avoid convergence to suboptimal solutions and to encourage continuous exploration of new routes.
* **AI and ML for Pheromone Management**: Machine learning algorithms can predict optimal pheromone levels using historical data and trends.
  + **Predictive Analytics**: Using historical route data to predict future pheromone levels, optimising for factors like seasonality in traffic patterns or sales cycles.
  + **Adaptive Pheromone Adjustment**: AI can dynamically adjust pheromone levels in real-time based on changing conditions such as road closures or unexpected traffic jams.

#### 3. Routes as Trails

The paths gnats take to visit cities represent potential routes in the TSP. Each route is an ordered list of cities the gnat (salesperson) visits.

**Extended Description**:

* **Route Representation**: Each route can be represented as a permutation of city indices, forming an ordered list that defines the sequence in which cities are visited.
  + **Genetic Algorithms**: Genetic algorithms can evolve and optimise these routes over successive generations. Crossover and mutation operations can combine and tweak routes to find more optimal solutions.
* **AI and ML for Route Optimization**: Advanced algorithms can be used to fine-tune the routes.
  + **Deep Learning**: Utilize deep learning to predict optimal city sequences based on complex, multi-dimensional data inputs such as weather forecasts, sales data, and historical traffic patterns.
  + **Optimization Algorithms**: Optimise techniques like Ant Colony Optimization (ACO) or Particle Swarm Optimization (PSO) to enhance route efficiency.
* **Operational Constraints**: Real-world constraints can be incorporated to make the simulation more realistic.
  + **Driving Hours and Rest Periods**: Simulate real-world driving regulations by limiting driving hours and mandating rest periods.
  + **Sales Hours and Commission Calculation**: Include sales hours within the route planning, where each visit to a city not only counts towards route length but also the potential sales and commission earned.

#### Enhancing Realism and Feasibility

* **Commission-Based Earnings**: Simulate the earnings of each salesperson (gnat) by calculating the commission based on sales made during each city visit. The commission structure can influence the gnat’s route preference, aiming to maximise route efficiency and sales.
* **Base Salary and Total Compensation**: Incorporate a base salary for each salesperson, adding to the realism of the simulation. The overall fitness of each gnat can be a combination of the total route length and the total compensation (base salary + commission).
* **Tour Length and Quality of Life**: Balance the need to minimise tour length with quality-of-life considerations for the salespersons, such as minimising overnight stays or ensuring optimal work-life balance.

This extended and enhanced approach effectively maps simulation parameters to TSP solutions and incorporates AI and ML techniques to dynamically refine and optimise the routes, considering various real-world constraints and objectives.

### Exploration and Exploitation in Gnot-Inspired TSP Solutions

#### 1. Exploration

**Exploration** involves diversifying the search space to find new and potentially better solutions. In a gnot-inspired approach for the TSP, various techniques can be employed to explore possible routes thoroughly.

**a. Random Initialization**:

* **Initial Population**: Begin with a diverse population of gnats, each assigned a randomly generated route. This randomness ensures that the initial search space is vast and varied, preventing premature convergence to suboptimal solutions.
  + **Enhanced Random Initialization**: Instead of purely random routes, utilise heuristic-based initialisation to ensure some initial quality. For instance, use the nearest neighbour or greedy algorithms to generate starting routes that are somewhat optimised.

**b. Mutation**:

* **Route Mutation**: Occasionally alter a gnat's route by randomly swapping the order of two cities or reversing the order of a subsequence of cities. This mutation helps explore new routes that might not be discovered through crossover alone.
  + **Adaptive Mutation Rates**: Implement adaptive mutation rates that change based on the performance of the population. Higher mutation rates can be used when the population stagnates, and lower rates when converging towards optimal solutions.
  + **Advanced Mutation Techniques**: Use advanced mutation techniques such as scramble mutation, where a subset of cities is randomly shuffled, or inversion mutation, where a route segment is reversed.

**c. Crossover**:

* **Route Crossover**: Combine segments of routes from two different gnats to create new routes. This crossover allows the best traits from each parent route to be shared, potentially leading to better offspring routes.
  + **Partially Matched Crossover (PMX)**: A sophisticated crossover method where subsequences of two-parent routes are exchanged while preserving the validity of the routes.
  + **Ordered Crossover (OX)**: Ensure that the offspring inherits the order and position of cities from both parents, maintaining feasibility and potentially improving route quality.

#### 2. Exploitation

**Exploitation** involves intensifying the search for the best solutions found so far to refine and improve them. In a gnot-inspired approach for the TSP, various techniques can be employed to exploit the current best solutions.

**a. Pheromone Update**:

* **Pheromone Deposition**: Routes that yield shorter travel distances deposit more pheromones in the cities they visit. This pheromone deposition makes those routes more attractive to other gnats, guiding the population towards promising areas of the search space.
  + **Dynamic Pheromone Updates**: Adjust the amount of pheromone deposited based on the relative quality of the route. Routes significantly better than the average can deposit more pheromones, enhancing exploitation.
  + **Pheromone Evaporation**: Introduce a pheromone evaporation mechanism to prevent over-exploitation and encourage continuous exploration. Pheromone levels can decrease over time, reducing the attractiveness of routes that were once optimal but may no longer be.

**b. Local Search**:

* **Local Optimization**: Perform local optimisation techniques such as 2-opt or 3-opt moves on the gnat’s route. These techniques involve removing and re-adding edges to improve the route by eliminating crossings and reducing the total distance.
  + **2-opt Optimization**: Iteratively remove two edges and reconnect the resulting two segments in the opposite order, effectively reversing a segment of the route to eliminate crossings and reduce travel distance.
  + **3-opt Optimization**: Remove three edges and reconnect the resulting segments to reduce the total route length. This more complex move can lead to significant improvements in route quality.
  + **Simulated Annealing**: Integrate simulated annealing techniques to escape local optima during local search, allowing for occasional acceptance of worse routes to explore new areas of the search space.

**c. Hybrid Approaches**:

* **Balancing Exploration and Exploitation**: Implement hybrid approaches that balance exploration and exploitation dynamically. For example, a multi-phase approach should be used where initial iterations focus more on exploration, and later iterations gradually increase exploitation.
  + **Adaptive Strategies**: Use adaptive strategies that adjust the balance between exploration and exploitation based on the current state of the search. If the population diversity is low, increase exploration; if convergence is observed, increase exploitation.
  + **Hyper-Heuristic Approaches**: Employ hyper-heuristics to select and apply different exploration and exploitation strategies based on the performance of the search process. These higher-level heuristics can switch between low-level heuristics to optimise the search dynamically.

**d. AI and ML Integration**:

* **Machine Learning for Strategy Selection**: Machine learning models are used to predict the effectiveness of different exploration and exploitation strategies based on historical data. This prediction can guide selecting strategies likely to yield the best results.
  + **Reinforcement Learning**: Implement reinforcement learning algorithms to dynamically adjust exploration and exploitation parameters based on real-time feedback from the search process.
  + **Deep Learning for Pattern Recognition**: Use deep learning models to recognise patterns in successful routes and predict promising areas of the search space, guiding the population towards high-quality solutions.

By combining these enhanced exploration and exploitation techniques, the gnot-inspired approach for the TSP can effectively balance the need for discovering new routes and refining existing ones, leading to robust and high-quality solutions.

### Evaluation of Solutions in Gnot-Inspired TSP Solutions

#### 1. Fitness Function

**Fitness Function** is the core of evaluating solutions in a gnot-inspired approach to the TSP. The fitness function assesses the quality of each gnat's route, primarily based on the total travel distance.

* **Total Distance Calculation**: The primary metric for fitness evaluation is the total distance travelled by the gnat on its route. The shorter the total distance, the higher the gnat's fitness score.
  + **Mathematical Representation**: Fitness can be represented as the inverse of the total distance, i.e., Fitness=1/Total Distance. This ensures that shorter routes (lower total distances) result in higher fitness scores.
  + **Penalty Mechanisms**: Introduce penalty mechanisms for infeasible routes (if applicable), such as those that violate problem constraints (e.g., not visiting all cities). These penalties reduce the fitness score, discouraging such routes.

**Enhanced Fitness Evaluation**:

* **Time-Dependent Fitness**: Incorporate time-dependent factors like traffic conditions or time windows into the fitness evaluation. This allows for more realistic and practical route planning.
* **Multi-Objective Fitness**: Implement multi-objective fitness functions considering multiple criteria, such as distance, travel time, and fuel consumption. This provides a more comprehensive evaluation of route quality.

#### 2. Pheromone Intensity

**Pheromone Intensity** plays a crucial role in guiding the search towards high-quality solutions. Their performance reflects the desirability of routes.

* **Pheromone Deposition**: Routes with shorter distances deposit more pheromones in the cities they visit. This increases their attractiveness, guiding other gnats towards them.
  + **Proportional Deposition**: The amount of pheromone deposited can be proportional to the inverse of the total distance, i.e., Pheromone Amount=k/Total Distance, where k is a constant.
  + **Dynamic Pheromone Updates**: Adjust pheromone levels dynamically based on the route quality relative to the population. Exceptional routes deposit significantly more pheromones.

**Advanced Pheromone Strategies**:

* **Pheromone Evaporation**: Implement pheromone evaporation to prevent over-exploitation of specific routes. This gradual reduction in pheromone levels ensures continuous exploration of the search space.
* **Multi-Layer Pheromones**: Use multi-layer pheromone strategies in which different types represent various route characteristics, such as distance, time, and risk. This provides a more nuanced guidance mechanism.

#### 3. Selection

**Selection** determines which gnats are chosen to reproduce and pass on their routes to the next generation. This process ensures that better routes are likely to contribute to future solutions.

* **Fitness-Proportionate Selection**: Gnats are selected for reproduction based on their fitness scores. Gnats with higher fitness scores are more likely to be selected, ensuring that shorter routes are favoured.
  + **Roulette Wheel Selection**: A standard method where the probability of selection is proportional to the fitness score. A roulette wheel is spun, and the gnat corresponding to the selected segment is chosen.
  + **Tournament Selection**: Another method where a subset of gnats is randomly chosen, and the gnat with the highest fitness in the subset is selected for reproduction.

**Enhanced Selection Techniques**:

* **Stochastic Universal Sampling (SUS)**: A method that provides a more even selection pressure, ensuring diversity in the population by giving each gnat a chance proportional to its fitness.
* **Elitism**: Preserve a subset of the best-performing gnats to ensure their high-quality routes are retained in the population. This prevents the loss of optimal solutions due to random fluctuations.

#### 4. Diversity Maintenance

**Diversity Maintenance** is essential to prevent premature convergence to suboptimal solutions. Maintaining a diverse population ensures a thorough exploration of the search space.

* **Random Mutations**: Introduce random mutations in the routes of gnats to explore new areas of the search space. Mutations can include swapping cities, reversing subsequences, or randomly inserting cities at different positions.
  + **Adaptive Mutation Rates**: Use adaptive mutation rates that change based on the population's diversity. Higher mutation rates are used when diversity is low, and lower rates are used when diversity is high.
  + **Diverse Initial Population**: Start with a highly diverse initial population to cover a broad range of potential solutions.

**Advanced Diversity Strategies**:

* **Niching Techniques**: Implement niching techniques to maintain subpopulations exploring different search space regions. This can involve clustering similar routes and applying different selection pressures within each cluster.
* **Fitness Sharing**: Use fitness-sharing methods in which gnats that share similar routes have their fitness scores reduced. This prevents overcrowding in specific areas of the search space and encourages exploration.
* **Multi-Population Strategies**: Maintain multiple populations that evolve independently and occasionally exchange solutions. This increases overall diversity and reduces the risk of convergence to local optima.

Integrating these enhanced techniques for exploration, exploitation, fitness evaluation, and diversity maintenance, a gnot-inspired approach to the TSP can effectively balance the search process, leading to high-quality and robust solutions. This approach leverages the strengths of evolutionary algorithms and swarm intelligence, providing a comprehensive framework for solving complex optimisation problems.

### Extending and Enhancing Overall Feasibility in Gnot-Inspired TSP Solutions

#### 1. Scalability

Scalability ensures that the gnot-inspired approach can handle increasingly large and complex Travelling Salesperson Problem (TSP) instances.

* **Adjustable Population Size**: The approach can be scaled to accommodate more significant problem instances by varying the number of gnats in the simulation. More gnats can explore a more extensive solution space more thoroughly.
  + **Dynamic Population Management**: Implement dynamic population size adjustment based on the problem's complexity. Please increase the number of gnats facing more significant problem instances and reduce them to smaller ones to optimise computational resources.
* **Pheromone Deposition Intensity**: Modulating the intensity of pheromone deposition can fine-tune the algorithm's sensitivity to good routes. Higher pheromone deposition can make the algorithm more responsive to high-quality solutions, improving convergence speed and quality.
  + **Adaptive Pheromone Scaling**: Use adaptive scaling of pheromone deposition based on the current search phase. Increase pheromone sensitivity during the exploitation phase and reduce it during the exploration phase to balance the search process.

**Enhanced Scalability Techniques**:

* **Hierarchical Clustering**: Implement hierarchical clustering to divide the problem space into manageable sub-problems. Solve these sub-problems independently before combining the solutions, enabling the algorithm to handle extensive problem instances efficiently.
* **Distributed Computing**: Utilize distributed computing environments, such as cloud-based platforms, to run multiple algorithm instances in parallel, further enhancing scalability.

#### 2. Parallelization

Parallelisation leverages modern multi-core processors to evaluate and update the routes of different gnats simultaneously, significantly improving computational efficiency.

* **Concurrent Evaluation**: Evaluate the fitness of all gnats in parallel, utilising multiple CPU cores. This drastically reduces the time required for each iteration of the algorithm.
  + **Thread Pool Management**: Implement efficient thread pool management to balance the computational load across available processors, ensuring optimal resource utilisation.
* **Simultaneous Pheromone Update**: Update pheromone levels concurrently for different routes, reducing the bottleneck caused by sequential pheromone updates.
  + **Lock-Free Data Structures**: Use lock-free data structures for pheromone updates to avoid synchronisation overhead, further speeding up the process.

**Advanced Parallelization Strategies**:

* **GPU Acceleration**: Leverage Graphics Processing Units (GPUs) for parallel fitness evaluations and pheromone updates. GPUs offer massive parallelism, significantly accelerating the search process.
* **Asynchronous Parallelization**: Implement asynchronous parallelisation techniques where different gnats can operate independently without waiting for synchronisation points. This allows for continuous progress and reduces idle time.

#### 3. Real-World Applicability

The gnot-inspired approach's adaptability and efficiency suit real-world TSP scenarios, such as logistics and route planning.

* **Adaptability to Constraints**: The approach can incorporate real-world constraints, such as time windows, vehicle capacities, and traffic conditions. This applies to complex logistics problems where multiple constraints must be satisfied.
  + **Constraint Handling Mechanisms**: Implement specific mechanisms to handle various constraints, such as penalty functions for constraint violations or repair functions to adjust infeasible solutions.
* **Efficiency in Dynamic Environments**: The algorithm's ability to adapt to changing conditions, such as traffic patterns or delivery priorities, makes it highly efficient in dynamic real-world scenarios.
  + **Real-Time Updates**: Integrate real-time data sources, such as traffic updates or weather conditions, to continuously adjust the routes and improve efficiency.

**Enhanced Real-World Applicability**:

* **Integration with IoT Devices**: Connect the algorithm with Internet of Things (IoT) devices, such as GPS trackers and traffic sensors, to receive real-time data and optimise routes dynamically.
* **Application in Autonomous Vehicles**: Utilize the approach for route planning in autonomous vehicles, where real-time adaptability and efficiency are critical for optimal performance.

#### 4. Hybridization

Combining the gnot-inspired approach with other metaheuristic methods can enhance performance and robustness, leveraging the strengths of multiple algorithms.

* **Genetic Algorithms**: Integrate genetic algorithms to introduce crossover and mutation operations, enhancing the search process. Genetic algorithms can provide solid global search capabilities, complementing the local search strengths of the gnot-inspired approach.
  + **Hybrid Operators**: Design hybrid operators that combine pheromone-guided search with genetic crossover and mutation, effectively balancing exploration and exploitation.
* **Ant Colony Optimization**: Combine with Ant Colony Optimization (ACO) to leverage pheromone-based exploration mechanisms. ACO's well-established pheromone update rules can enhance the search efficiency of the gnot-inspired approach.
  + **Multi-Objective Optimization**: Implement multi-objective optimisation techniques to balance multiple criteria, such as distance, time, and cost, providing a comprehensive solution to complex TSP instances.

**Advanced Hybridization Techniques**:

* **Adaptive Hybridization**: Develop adaptive hybridisation strategies that dynamically adjust the combination of algorithms based on the search progress. This ensures optimal performance throughout the search process.
* **Ensemble Methods**: Use ensemble methods that combine multiple algorithmic approaches, such as gnot-inspired, genetic, and ant colony optimisation, to produce a robust and versatile solution framework.

By extending and enhancing these aspects, the gnot-inspired approach to solving the TSP can be made more scalable, efficient, applicable to real-world scenarios, and robust through hybridisation, ultimately providing a powerful tool for complex optimisation problems.

### Proposed Method for Gnot-Inspired Approach to Solving the TSP

#### 1. Initialization

**Objective**: Generate a starting point for the algorithm by creating an initial population of potential solutions (routes) and setting up the environment.

* **Generate Initial Population**:
  + **Random Routes**: Create an initial population of gnats, each representing a potential solution to the TSP with a randomly generated route. This ensures diverse starting points for the search process.
  + **Diverse Sampling**: Ensure the initial routes cover many possibilities to avoid premature convergence. This can be achieved using various randomisation techniques, such as uniform or stratified sampling.
* **Initialize Pheromone Levels**:
  + **Pheromone Initialization**: Set initial pheromone levels for each city and route. Initially, pheromones can be evenly distributed or set to a baseline level, indicating no prior knowledge about the best routes.
  + **City-Specific Pheromones**: Assign pheromone levels to specific city pairs, representing the desirability of travelling between them. This helps in guiding the gnats towards promising areas of the solution space.

**Enhanced Initialization Techniques**:

* **Heuristic Initialization**: Use heuristic methods, such as a nearest neighbour or greedy algorithms, to generate a portion of the initial population. This can provide reasonable starting solutions that guide the search process.
* **Diverse Initial Population**: Use different random seeds and route generation methods to ensure a diverse initial population. This prevents the algorithm from converging too quickly on suboptimal solutions.

#### 2. Iteration

**Objective**: Iteratively improve the solutions by evaluating and updating routes, applying local search, and generating new solutions through genetic operations.

* **Evaluate Routes**:
  + **Fitness Evaluation**: Calculate the total travel distance for each gnat's route. Shorter routes have higher fitness, encouraging the algorithm to focus on optimising travel distance.
  + **Fitness Function**: Implement a fitness function that not only considers the total distance but also other factors, such as time windows, vehicle capacities, and traffic conditions.
* **Update Pheromone Levels**:
  + **Pheromone Deposition**: Update pheromone levels based on the fitness of the routes. Routes with shorter distances deposit more pheromones, making them more attractive for future iterations.
  + **Pheromone Evaporation**: Apply pheromone evaporation to reduce the intensity of pheromones over time. This prevents over-reliance on early reasonable solutions and encourages exploration.
* **Local Search Techniques**:
  + **2-Opt and 3-Opt Moves**: To refine routes, such as 2-opt and 3-opt moves, apply local optimisation techniques. These techniques improve solutions by iteratively swapping city pairs or reversing subsequences to reduce travel distance.
  + **Hill Climbing**: Implement hill climbing as a local search method, where small changes are made to the route to find a better solution in the immediate neighbourhood.
* **Crossover and Mutation**:
  + **Crossover Operations**: Combine parts of routes from different gnats to create new routes. Techniques, like ordered crossover or partially matched crossover (PMX), can be used to preserve good traits and introduce diversity.
  + **Mutation Operations**: Occasionally mutate routes by swapping the order of cities or reversing subsequences. Mutation introduces randomness and helps in exploring new areas of the solution space.
* **Selection**:
  + **Selection Mechanism**: Select the best routes based on fitness to form the next generation of gnats. Techniques like roulette wheel selection or tournament selection can be used to balance exploration and exploitation.
  + **Elitism**: Preserve a portion of the best solutions from the current generation to ensure that the best-known solutions are carried forward.

**Enhanced Iteration Techniques**:

* **Adaptive Crossover and Mutation**: Based on the search progress, implement adaptive rates for crossover and mutation. Increase the rates during the early stages of exploration and decrease them later for exploitation.
* **Hybrid Local Search**: Combine multiple local search techniques, such as simulated annealing and tabu search, to enhance the refinement process.

#### 3. Termination

**Objective**: Define stopping criteria to conclude the search process once satisfactory solutions or predefined conditions are met.

* **Stopping Criteria**:
  + **Maximum Iterations**: Set maximum iterations to stop the algorithm. This ensures the search process is computationally feasible and terminates within a reasonable time frame.
  + **Convergence Threshold**: Terminate the algorithm when the improvement in the best solution falls below a certain threshold over a specified number of iterations, indicating convergence.
  + **Satisfactory Route Length**: Define a target route length as a stopping criterion. The algorithm stops once a route with a length below this target is found.

**Enhanced Termination Techniques**:

* **Dynamic Stopping Criteria**: Implement dynamic stopping criteria that adjust based on the search progress. For example, the algorithm can continue if significant improvements are still being made, even if the maximum iterations are reached.
* **Multi-Criteria Termination**: Use multiple criteria, such as a combination of maximum iterations, convergence threshold, and satisfactory route length, to determine when to stop the algorithm.

By extending and enhancing the proposed method, the gnot-inspired approach can effectively tackle the TSP, leveraging advanced initialisation, iteration, and termination techniques to achieve high-quality solutions efficiently.

# Appendix 1

## The complete model code

**globals** [  
 larval-deaths ;; Number of larval deaths  
 pupal-deaths ;; Number of pupal deaths  
 gnot-count ;; Total count of gnats  
 mean-initial-strength-history ;; History of mean initial strength for plotting  
 ;inheritance ;; Toggle for inheritance of strength  
 ;stddev-strength ;; Standard deviation for strength inheritance  
 time-of-day ;; Current time of day in the simulation  
 ;temperature ;; Environmental temperature  
 ;wind-speed ;; Environmental wind speed  
 ;humidity ;; Environmental humidity  
 ;kill-gnot  
 index  
 larval-births  
]  
  
  
  
**patches-own** [terrain-type pheromone-level] ;; Terrain type and pheromone level of each patch  
  
**turtles-own** [  
 age ;; Age in hours  
 speed ;; Movement speed  
 initial-strength ;; Initial strength of the gnat  
 strength ;; Current strength of the gnat  
 can-reproduce ;; Flag indicating if the gnat can reproduce  
 id ;; Unique identifier for the gnat  
 is-parent ;; Flag indicating if the gnat has reproduced  
 social-level ;; Level of social interaction for clustering  
 resistance ;; Genetic trait for resistance  
]  
  
**breed** [predators predator]  
  
**to** setup  
 clear-all  
 resize-world 0 100 0 100  
 set-patch-size 4.0  
 set larval-deaths 0  
 set pupal-deaths 0  
 set larval-births 0  
 set gnot-count 0  
 set mean-initial-strength-history []  
 set time-of-day 0  
 set temperature 20  
 set wind-speed 0  
 set humidity 50  
 set temp-fluctuation-range 2  
 set wind-fluctuation-range 2  
 set humidity-fluctuation-range 5  
  
 ;; Generate terrain blobs  
 generate-terrain-blobs  
  
 ;; Create initial gnats  
 create-turtles population [  
 set age 0  
 set shape "dot"  
 set size 0.8  
 set strength random-normal mean-strength stddev-strength  
 set initial-strength strength  
 set speed random 10  
 set resistance random 10  
 setxy random-xcor random-ycor  
 set can-reproduce 1  
 set id who  
 set is-parent **false**  
 set social-level 0  
 set pheromone-level 0  
 set gnot-count gnot-count + 1  
 ]  
  
 ;; Create initial predators  
 create-predators 5 [  
 set color red  
 set size 2  
 setxy random-xcor random-ycor  
 ]  
  
 reset-ticks  
**end**  
  
**to** generate-terrain-blobs  
 let terrain-types ["forest" "water" "grassland"]  
 ask patches [ set terrain-type "" set pheromone-level 0 ]  
  
 foreach n-values (length terrain-types \* 3) [index] [  
 let current-terrain one-of terrain-types  
 let initial-patch one-of patches with [terrain-type = ""]  
 ask initial-patch [  
 set terrain-type current-terrain  
 ;; Use a turtle to grow the terrain blobs  
 sprout 1 [  
 grow-terrain-blobs current-terrain 300 ;; Adjust the blob-size as needed  
 die  
 ]  
 ]  
 ]  
  
 ask patches [  
 if terrain-type = "forest" [ set pcolor green ]  
 if terrain-type = "water" [ set pcolor blue ]  
 if terrain-type = "grassland" [ set pcolor yellow ]  
 ]  
**end**  
  
**to** grow-terrain-blobs [current-terrain blob-size]  
 let patches-to-grow patch-set patch-here  
 while [blob-size > 0 and any? patches-to-grow] [  
 let current-patch one-of patches-to-grow  
 ask current-patch [  
 ask neighbors with [terrain-type = ""] [  
 set terrain-type current-terrain  
 ]  
 ]  
 set patches-to-grow patch-set patches with [terrain-type = current-terrain]  
 set blob-size blob-size - 1  
 ]  
**end**  
  
**to** go  
 ;; Update environmental factors  
 set time-of-day (time-of-day + 1) mod 24  
 set temperature temperature + random-normal 0 temp-fluctuation-range  
 set wind-speed wind-speed + random-normal 0 wind-fluctuation-range  
 set humidity humidity + random-normal 0 humidity-fluctuation-range  
  
 ;; Determine favorable conditions based on environmental factors  
 let favorable-terrain ""  
 if (temperature > 25 and wind-speed < 3) [  
 set favorable-terrain "water"  
 ]  
 if (wind-speed > 5) [  
 set favorable-terrain "forest"  
 ]  
 if (favorable-terrain = "") [  
 set favorable-terrain "grassland"  
 ]  
  
 ask turtles [  
 ;; Adjust behavior based on terrain  
 let terrain [terrain-type] of patch-here  
 if terrain = "water" [  
 ;; Define water behavior  
 ]  
 if terrain = "forest" [  
 ;; Define forest behavior  
 ]  
 if terrain = "grassland" [  
 ;; Define grassland behavior  
 ]  
  
 ;; Existing gnat behaviors...  
 let c color  
 let p-id id  
 let parent-initial-strength initial-strength ;; Store parent's initial strength  
 let parent-speed speed  
 let parent-resistance resistance  
 ifelse age < 36 [  
 set speed 0  
 set shape "dot"  
 set size 0.8  
 ] [  
 ifelse age < 192 [  
 set speed strength \* 0.0003  
 set shape "bug"  
 set size 0.8  
 ] [  
 ifelse age < 240 [  
 set speed 0  
 set shape "bug"  
 set size 1.2  
 ] [  
 set shape "default"  
 set size 3  
 set speed strength \* 0.02 + 0.5  
 if random 168 < 1 and can-reproduce = 1 [  
 let patch-count count patches in-radius egg-laying-radius  
 ask patches in-radius egg-laying-radius [  
 if random patch-count < expected-egg-quantity [  
 sprout 1 [  
 set age 0  
 set heading random 360  
 set speed 0  
 ifelse inheritance [  
 ;; Inherit initial strength from parent with some deviation  
 set initial-strength random-normal parent-initial-strength stddev-strength  
 ] [  
 ;; Assign random initial strength  
 set initial-strength random-normal mean-strength stddev-strength  
 ]  
 set strength initial-strength  
 set speed (parent-speed + random-normal 0 1)  
 set resistance (parent-resistance + random-normal 0 1)  
 set color c  
 set can-reproduce random 2  
 set shape "dot"  
 set size 0.8  
 set id p-id  
 set gnot-count gnot-count + 1  
 set larval-births larval-births + 1  
 ]  
 ]  
 ]  
 set strength strength / 2  
 set is-parent **true**  
 ]  
 ]  
 ]  
 ]  
 rt random 10  
 lt random 10  
 fd speed  
  
 ;; Update pheromone level based on terrain  
 if terrain = "water" [  
 set pheromone-level pheromone-level + 0.5 ;; Slow dissipation in water  
 ]  
 if terrain = "forest" [  
 set pheromone-level pheromone-level + 0.2 ;; Moderate dissipation in forest  
 ]  
 if terrain = "grassland" [  
 set pheromone-level pheromone-level - 0.1 ;; Faster dissipation in grassland  
 ]  
 if pheromone-level < 0 [ set pheromone-level 0 ] ;; Ensure non-negative pheromone level  
  
 ;; Update social level based on proximity to other gnats  
 set social-level count turtles in-radius 3 ;; Increased radius for clustering  
  
 set strength max list (strength - 0.1) 0  
 set age age + 1  
 if strength = 0 [ kill-gnot ]  
 if age > 36 [set strength strength - 10 \* (count (turtles-on patch-here) - 1)]  
  
 ;; Log values for verification  
 print (word "pheromone-level: " pheromone-level)  
 print (word "social-level: " social-level)  
 ]  
  
 ;; Predator behavior  
 ask predators [  
 move-to one-of turtles  
 if distance one-of turtles < 1 [  
 ask one-of turtles-here [ kill-gnot ]  
 ]  
 ]  
  
 update-custom-plots  
 tick  
**end**  
  
**to** kill-gnot  
 ifelse age < 192 [  
 set larval-deaths larval-deaths + 1  
 ] [  
 set pupal-deaths pupal-deaths + 1  
 ]  
 die  
**end**  
  
**to** update-custom-plots  
 ;; Plot Mean Initial Strength  
 set-current-plot "Mean Initial Strength"  
 set-current-plot-pen "mean strength"  
 let mean-initial-strength mean [initial-strength] of turtles  
 set mean-initial-strength-history lput mean-initial-strength mean-initial-strength-history  
 plot mean-initial-strength  
  
 ;; Plot Total Gnat Count  
 set-current-plot "Total Gnat Count"  
 set-current-plot-pen "gnat-count"  
 plot count turtles  
  
 ;; Plot Larval and Pupal Deaths  
 set-current-plot "Larval and Pupal Deaths"  
 set-current-plot-pen "larval-deaths"  
 plot larval-deaths  
 set-current-plot-pen "pupal-deaths"  
 plot pupal-deaths  
  
 ;; Plot Unique Gnat Dynasties  
 set-current-plot "Unique Gnat Dynasties"  
 set-current-plot-pen "dynasties-remaining"  
 plot length remove-duplicates [id] of turtles  
  
 ;; Plot Environmental Factors  
 set-current-plot "Environmental Factors"  
 set-current-plot-pen "temperature"  
 plot temperature  
 set-current-plot-pen "wind-speed"  
 plot wind-speed  
 set-current-plot-pen "humidity"  
 plot humidity  
  
 ;; Plot Time of Day  
 set-current-plot "Time of Day"  
 set-current-plot-pen "time-of-day"  
 plot sin ((time-of-day / 24) \* 2 \* pi)  
  
 ;; Plot Larval Births  
 set-current-plot-pen "larval-births"  
 plot larval-births  
  
 ;; Plot Dynasties  
 set-current-plot-pen "dynasties-remaining"  
 plot length remove-duplicates [id] of turtles  
  
 ;; Plot Pheromone Levels if any turtles exist  
 if any? turtles [  
 set-current-plot "Pheromone Levels"  
 set-current-plot-pen "pheromone-level"  
 let mean-pheromone-level mean [pheromone-level] of turtles  
 if mean-pheromone-level != nobody [ ;; Ensure there is a value to plot  
 plot mean-pheromone-level  
 ]  
  
 ;; Plot Social Clustering if any turtles exist  
 set-current-plot "Social Clustering"  
 set-current-plot-pen "social-level"  
 let mean-social-level mean [social-level] of turtles  
 if mean-social-level != nobody [ ;; Ensure there is a value to plot  
 plot mean-social-level  
 ]  
 ]  
**end**

# Appendix 2

## maximize-family-survival.bsearch

<BehaviorSearch>

  <BehaviorSpace>

    <Model>

      <Path>E:/NetLogoModels/Gnots\_all\_working\_behavioural\_search.nlogo</Path>

      <Setup>setup</Setup>

      <Go>go</Go>

    </Model>

    <Replications>10</Replications>

    <RunMetricsAtEnd>true</RunMetricsAtEnd>

    <Metrics>

      <Metric>

        <Reporter>dynasties-remaining</Reporter>

      </Metric>

    </Metrics>

    <Parameters>

      <Parameter>

        <Name>mean-strength</Name>

        <Low>0</Low>

        <High>100</High>

      </Parameter>

      <Parameter>

        <Name>stddev-strength</Name>

        <Low>0</Low>

        <High>10</High>

      </Parameter>

    </Parameters>

    <Goals>

      <Goal>

        <Metric>dynasties-remaining</Metric>

        <Optimization>Maximize</Optimization>

      </Goal>

    </Goals>

    <Constraints>

      <Constraint>

        <Metric>dynasties-remaining</Metric>

        <Operator>&gt;</Operator>

        <Value>0</Value>

      </Constraint>

    </Constraints>

  </BehaviorSpace>

</BehaviorSearch>

# Appendix 3

## minimize-larval-pupal-deaths.besearch

<BehaviorSearch>

  <BehaviorSpace>

    <Model>

      <Path>E:/NetLogoModels/Gnots\_all\_working\_behavioural\_search.nlogo</Path>

      <Setup>setup</Setup>

      <Go>go</Go>

    </Model>

    <Replications>10</Replications>

    <RunMetricsAtEnd>true</RunMetricsAtEnd>

    <Metrics>

      <Metric>

        <Reporter>larval-deaths</Reporter>

      </Metric>

      <Metric>

        <Reporter>pupal-deaths</Reporter>

      </Metric>

    </Metrics>

    <Parameters>

      <Parameter>

        <Name>mean-strength</Name>

        <Low>0</Low>

        <High>100</High>

      </Parameter>

      <Parameter>

        <Name>stddev-strength</Name>

        <Low>0</Low>

        <High>10</High>

      </Parameter>

    </Parameters>

    <Goals>

      <Goal>

        <Metric>larval-deaths</Metric>

        <Optimization>Minimize</Optimization>

      </Goal>

      <Goal>

        <Metric>pupal-deaths</Metric>

        <Optimization>Minimize</Optimization>

      </Goal>

    </Goals>

    <Constraints>

      <Constraint>

        <Metric>larval-deaths</Metric>

        <Operator>&lt;</Operator>

        <Value>population</Value>

      </Constraint>

      <Constraint>

        <Metric>pupal-deaths</Metric>

        <Operator>&lt;</Operator>

        <Value>population</Value>

      </Constraint>

    </Constraints>

  </BehaviorSpace>

</BehaviorSearch>

# Appendix 4

## maximize\_mean\_initial\_gnot\_strength.bsearch

<BehaviorSearch>

  <BehaviorSpace>

    <Model>

      <Path>E:/NetLogoModels/Gnots\_all\_working\_behavioural\_search.nlogo</Path>

      <Setup>setup</Setup>

      <Go>go</Go>

    </Model>

    <Replications>10</Replications>

    <RunMetricsAtEnd>true</RunMetricsAtEnd>

    <Metrics>

      <Metric>

        <Reporter>mean [initial-strength] of turtles</Reporter>

      </Metric>

    </Metrics>

    <Parameters>

      <Parameter>

        <Name>mean-strength</Name>

        <Low>0</Low>

        <High>100</High>

      </Parameter>

      <Parameter>

        <Name>stddev-strength</Name>

        <Low>0</Low>

        <High>10</High>

      </Parameter>

    </Parameters>

    <Goals>

      <Goal>

        <Metric>mean [initial-strength] of turtles</Metric>

        <Optimization>Maximize</Optimization>

      </Goal>

    </Goals>

    <Constraints>

      <Constraint>

        <Metric>gnot-count</Metric>

        <Operator>&gt;</Operator>

        <Value>0</Value>

      </Constraint>

    </Constraints>

  </BehaviorSpace>

</BehaviorSearch>