Foraging Behavior of a Black Soldier Fly Larva:

An Agent-Based Modeling Simulation Study

Ethan Brady and Eagle Yuan

Oak Ridge High School

1450 Oak Ridge Turnpike

Oak Ridge, TN 37830

Center for Nanophase Materials Sciences

Oak Ridge National Laboratory

Table of Contents

1. Introduction

Black Soldier Flies, *Hermetia illucens*, are non-pest insects native to tropical and subtropical areas. Interest in the Black Soldier Fly larvae (BSFL), shown in Figure 1, originates from the possible application for recycling organic waste in an environmentally friendly manner. BSFL are about 20 mm long and 5 mm wide, but despite their small size, they can convert up to 80% of the wet weight of organic waste into high protein biomass (1, 2). This biomass can be used to feed animals such as chickens, fishes, and pets, resulting in several businesses focused on the farming of BSFL (2).

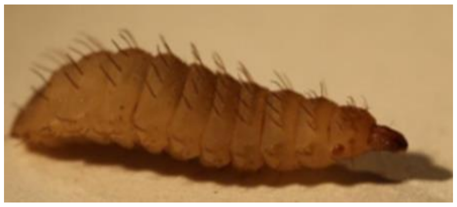
Figure 1: Black Soldier Fly larva are about 20 mm in length and 5 mm in width (3).

To increase the efficiency of BSFL farms, it is important to understand the feeding behavior of BSFL. While BSFL eat cooperatively, understanding how a single larva eats is necessary to create models involving multiple larvae. Agent-based modeling is a technique that can accurately simulate emergent behavior. In agent-based modeling, individuals interact amongst each other, and through these local interactions, population-level phenomena emerge. The goal of this project is to model the eating behavior of a single Black Soldier Fly larva using agent-based modeling in order to understand the mechanics of their collective motion.

2. Background

2.1 Black Soldier Fly Larvae

Black Soldier Flies do not eat; they rely solely on the food eaten as larvae. Thus, BSFL eat up to twice their body weight in food each day. Their primary mode of sensing is smell, but the extent of this sensing is unclear. Experiments have shown that blowing various scents over BSFL in a wind tunnel or placing larvae in a Y-maze does not result in the larvae consistently selecting the direction of a food scent (3). Instead, larvae wander randomly and occasionally happen to come across food.

Recently, the Hu group has researched the eating behavior of BSFL to develop more efficient BSFL farms (3). Modeled in Figure 2, video and particle tracer analysis have shown that BSFL aggregations restricted to a container with a food source form arrangements around the food source that resemble vortices and tori, while sloshing about as if a liquid. These two positions seem to arise from displacement of BSFL around the food source by pushing and shoving interactions amongst the BSFL (3). This movement is reminiscent of the vortices that exist in natural phenomena such as schools of fish, flocks of birds, and bacterial populations (4-6).

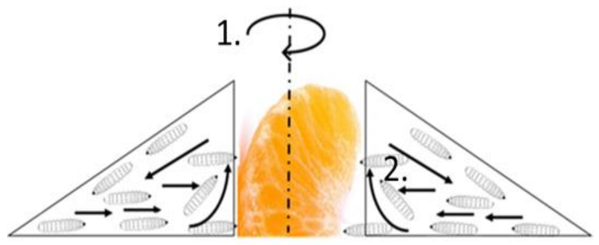

Figure 2: The movement of BSFL around a food source involves two motions as larvae try to eat. In position 1, active mixing of the larvae creates a vortex, represented by the rotational arrow. In position 2, larvae are displaced upward then fall away, creating a torus (3).

The individual eating behavior of BSFL was also quantified by Hu's group (3). In this study, a set of 10 petri dishes, each with a single BSFL and food source of chicken feed, was videoed for one hour. A larva spends on average 26% of the time eating, 62.9% of the time on the border of the dish, and 11.1% in the middle of the dish. However, individual larva showed a large variation in eating and boundary behavior, with some larvae not eating at all.

2.2 Agent-Based Modeling

Agent-based modeling is a simulation technique that is appropriate for understanding emergent behaviors, where the behavior of the system is dependent on the relationships among the individuals (7) such as flocking birds and the economy. In an ABM, simple rules are encoded, and complex results emerge: the individuals interact locally with each other, and out of these local interactions, population-level phenomena emerge. The advantage of an ABM is that computational simulations in science and engineering rely on equation-based, mathematical models but are difficult to implement in systems without a well-defined mathematical approach. However, ABMs allow simulation of previously believed unpredictable systems.

The four main components of an ABM are the agents, rules, environment, and parameters. An agent is defined as the individual that makes decisions in the model based on a set of rules that explicitly tell the agents what to do. The environment is where the agents reside and carry out these rules. In an ABM, the parameters are incorporated into the agents, environment, and rules. It is crucial that an ABM contains several parameters, because varying the parameters can drastically influence outcomes, revealing the relationship between a parameter and outcome. A simple example is cars driving on a highway. The agents in this system are the cars, the environment is the highway, the rules are speed limits and speeding up when there are no cars in front and slowing down when there are cars in front, and the parameters are the number of cars and the rate of acceleration and deceleration.

2.2.1 Termites

NetLogo is a popular agent-based modeling software program that is both resourceful and user-friendly (8). The Termites ABM in NetLogo's model library analyzes the movement of termites (9). Termites commonly gather wood chips and move these chips into piles, but there is no defined leader that controls the termites. Instead, each termite focuses solely on its own task regardless of the other termites. However, the termites indirectly affect each other because the placement of the wood chips forces the movement of the other termites to adjust to the environment. An example of the progression of the Termites model is shown in Figure 3, where the agents are white and wood chips are yellow. As shown in Figure 3a, the initial environment setup is a grid with random areas occupied by a wood chips. Figure 3b reveals the piles that have formed from the termites moving the wood chips. Interestingly, the final result is one, circular pile of wood chips shown in Figure 3c.

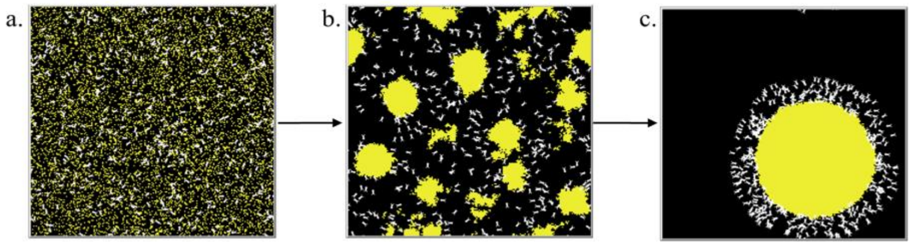

Figure 3: In the Termites environment, white represents the termites, and yellow represents the wood chips. (a) In the initial setup, the termites and wood chips are randomly scattered throughout the environment. (b) Circular piles of wood chips have been formed from the termites moving and placing their wood chips. (c) The final result shows an unexpected single, circular pile of wood chips.

In the Termites model, the termites are the agents, the initial environment setup is a grid with random areas occupied by a wood chip, and the two parameters are the number of agents and the probability that a wood chip occupies an area when the environment is setup. In the example shown in Figure 3, the default number of agents is set at 400, and the probability that a wood chip occupies an area is 20%. The rule for the agent is that if the agent encounters a wood chip, it holds the chip until it bumps into another chip. The agent then drops the chip they were holding. When not picking up or putting down a wood chip, the agents move forward one unit and rotate, according to the probability distribution shown in Figure 4.

2.2.2 Probability Distributions

A probability distribution (Figure 4) plots the probability of an event occurring versus a near continuous variable. A 0.02 chance of 0 in Figure 4 seems low, but the shape of the distribution being higher near 0 shows that values near 0 are most common. The most well-known distribution is the Gaussian distribution function, or normal distribution, which is widespread in nature and inherent in statistical sampling. It is bell-shaped, and like the distribution in Figure 4, values near the center are most common.

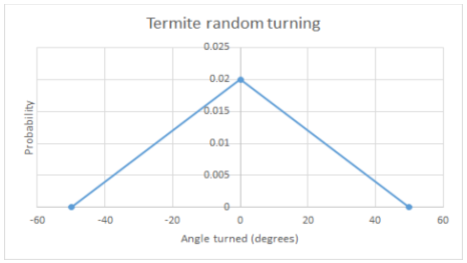

Figure 4: The graph above shows the probability distribution for a termite changing its heading. The probability that the termite continues to move in the same direction is the greatest.

2.2.3 Flocking

Another well-known example of an ABM in the field of natural sciences involves the emergence of flocking in birds. Flocking has been a subject of interest because these birds self-organize into formations without the assistance of a leading bird. This collective motion was first simulated by Craig Reynolds in his computer program Boids in 1986 (10). A more recent approach to flocking is in the model library of NetLogo called Flocking (11). Figure 5 is a visual progression of the model over 800 time steps. In the initial setup (Figure 5a), the birds are randomly distributed throughout the environment with random headings. After 800 time steps, the birds generally have the same heading and have formed groups with each other (Figure 5b).

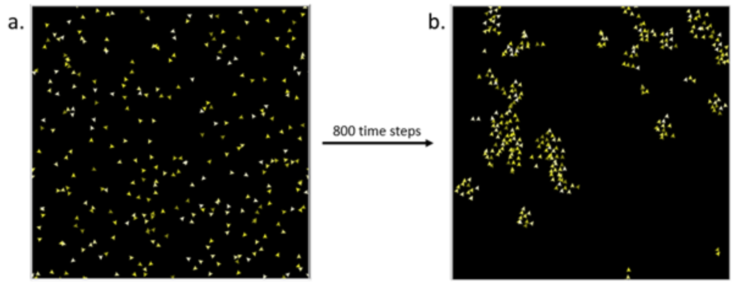

Figure 5: (a) The initial random orientation of the birds in Flocking. The birds are randomly distributed throughout the environment with random headings. (b) After 800 time steps, the birds are aligned in the same direction and moving parallel to each other, grouping themselves in clumps.

In the flocking model, the birds are the agents, and the environment is the *xy*-plane. The environment in flocking is wrapped meaning that when one agent goes through the upper edge of the square, it reappears at the bottom edge of the square, and the same applies horizontally. Wrapping the environment allows for an infinitely large environment, so the agents are free to continue moving. The agents follow four main rules: alignment, separation, cohesion, and move as shown in Figure 6.

Alignment is the tendency of agents to turn in the same direction as other agents, separation is the turn away from an agent when the agents are too close to each other, and cohesion is the long-range attraction between agents. In move, the agent simply advances one unit. Alignment, separation, and cohesion each require parameters max-align-turn, max-separate-turn, and max-cohere-turn, respectively. The remaining parameters include population, vision, and minimum separation of agents. Population is the number of agents, vision is how many units ahead the agents can detect, and minimum separation is the value compared to the distance between two agents.

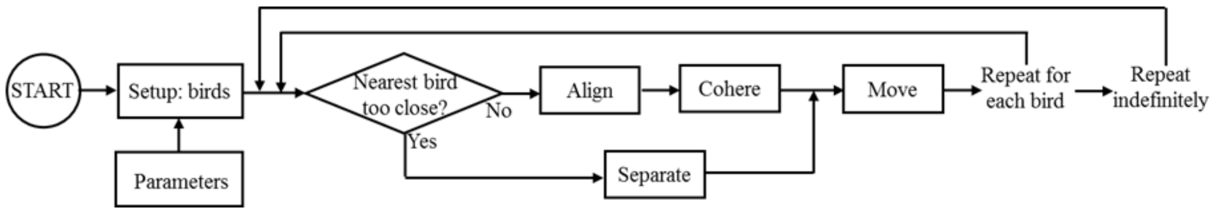

Figure 6: A flowchart describing the steps of the Flocking ABM in NetLogo.

In the alignment method (Figure 7), one agent first identifies nearby birds by finding the summation of the *x*-component and *y*-component headings of all nearby birds given by

$$c = \left(\sum_{i=0}^{P} \sin(h_i), \sum_{i=0}^{P} \cos(h_i)\right) \tag{1}$$

where P is the population size of nearby birds and h_i is the heading of the *i*th nearby bird. Netlogo defines North as 0 degrees, East as 90 degrees, South as 180 degrees, and West as 270 degrees. If there are no nearby birds, then there is no change in heading. Also if $c = (0,0)$, there is no change in heading. However, if one of the components of *c* is non-zero, the model calculates the difference in the heading by

$$|H_\mu - H_c| > t_a \tag{2}$$

where H_c is the heading of the agent executing align, t_a is the parameter input max-align-turn, and H_μ is determined by the equation

$$H_\mu = tan^{-1} \frac{\sum_{i=0}^{P} sin\,(h_i)}{\sum_{i=0}^{P} cos(h_i)}. \tag{3}$$

If Equation 2 is true, the bird turns right by $H_\mu - H_c$ degrees. Otherwise, the simulation computes $H_\mu - H_c$. If $H_\mu - H_c > 0$, the agent turns right by t_a. If $H_\mu - H_c < 0$, the bird turns left by t_a.

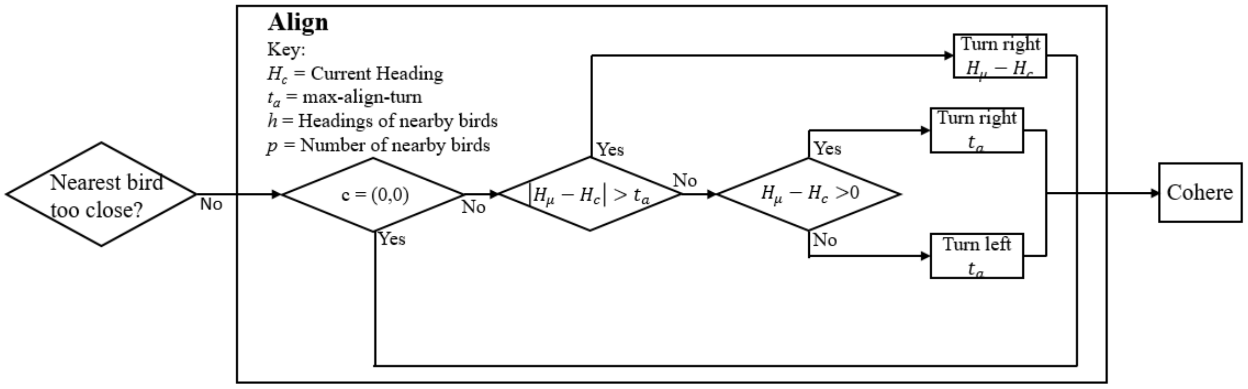

Figure 7: This flow diagram shows the decision-making process of the align rule.

2.3 Genetic Algorithms

A genetic algorithm is an optimization technique inspired by the process of natural selection. In a way, Darwinian evolution is how nature optimizes species for survival, so mimicking this process is an intuitive optimization technique. Genetic algorithms are effective for complex, nonlinear problems and for "quickly finding a sufficiently good solution" (12). A genetic algorithm attempts to optimize a fitness function in a population of parameter sets through the operators of selection, crossover, and mutation.

2.3.1 Fitness

Optimization is a common difficulty in science, where one might change variables to maximize or minimize some property. Generally, the property is quantified by an objective function $f(x_1, \mathrm{x}_2, \ldots, x_n)$ for given variables $x_1, x_2, \ldots, x_n$. In a genetic algorithm, the objective function is called a fitness function and the variables are called parameters. The fitness function defines the goal in the context of the problem.

For example, birds align to nearby birds in the flocking ABM, and one might use a genetic algorithm to find under what conditions, or parameters, all the birds align in the same direction (11, 13). The parameters of the genetic algorithm are the parameters of the flocking ABM: vision, max-align-turn, max-cohere-turn, max-separate-turn, and minimum-separation. The fitness function is defined for a state of birds after 75 time steps of the ABM; there are N birds where h_i is the heading of bird i. The fitness function is the sum of the standard deviation of the x- and y-components of a bird's heading given by

$$fitness = \sqrt{\frac{\sum_{i=1}^{N} \sin{(h_i)}^2}{N-1}} + \sqrt{\frac{\sum_{i=1}^{N} \cos{(h_i)}^2}{N-1}}. \tag{4}$$

The fitness function quantifies alignment, and the goal is minimize fitness since $fitness = 0$ is perfect alignment. Because the random initial position of birds creates different alignment for each simulation, the true fitness for a given parameter set is the average fitness of 5 simulation runs.

2.3.2 Population

A genetic algorithm creates a population of parameter sets to optimize the fitness function. In the context of the population, a parameter set is called an individual. In Figure 8, the individual is essentially the same as the parameter set and has the same color, and the population has individuals of different colors which correspond to different parameter sets. Individuals in the population are simultaneously evolved over generations. In Figure 8, each of the 3 generations have populations with individuals of different color. Typically, a population has 20 to 50 individuals, and 50 to 500 generations are processed (12). A genetic algorithm starts with a population of individuals with randomly determined parameters and finishes after any individual surpasses a desired fitness level or after a set computational time. The best individual of the final generation is the optimized solution.

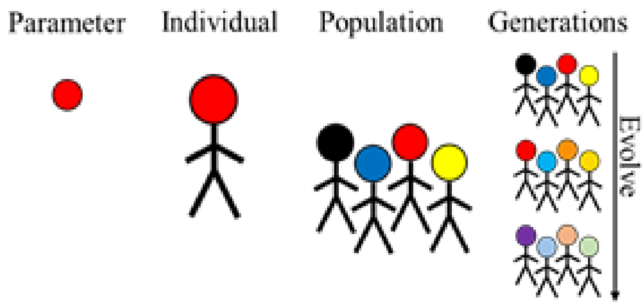

Figure 8: This hierarchy shows the entities of a genetic algorithm. The parameter set is represented by a color (14). The genetic algorithm acts on the parameter sets as individuals. Many individuals form the population, and populations are successively evolved over generations to reach a desired fitness.

A genetic algorithm uses the population to simultaneously investigate the entire search space. While other search algorithms might investigate the area around a local optimum while ignoring the path to the global optimum, a genetic algorithm is a global search technique that looks past local optima. Thus, genetic algorithms surpass other optimization methods in handling multimodal and non-differentiable problems (15).

2.3.3 Operators

The operators of selection, crossover, and mutation evolve each generation by choosing, combining, then altering individuals, respectively. There is no concrete way of coding a genetic algorithm, and many styles exist for each operator, so a general description follows (16). Figure 9 shows the three operators acting on a population, where Figure 9a, b, and c show net changes in the population as a result of each operator. Selection chooses individuals with relatively high fitness to become parents, thereby pushing the population towards the desired characteristics. In Figure 9, red is selected twice and black is not selected, so the net change (Figure 9a) is black to red. Crossover blends individuals, which can condense favorable characteristics of a population into a single individual. Two children are created from two parents, but this combination is sometimes asexual. Parents are blended at a probability of the crossover rate, typically 0.6 to 0.9; otherwise, the two selected parents are duplicated as two children. In Figure 9b, red and yellow are crossed to produced two orange shades, and the remaining individuals, red and blue, reproduce asexually to produce red and blue. Mutation slightly alters individuals, which allows for the exploration of new possibilities. As shown in Figure 9c, dark blue becomes light blue.

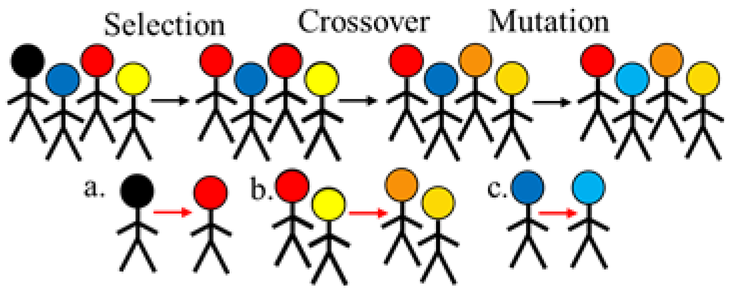

Figure 9: This progression shows the operators of a genetic algorithm, and the subparts are net changes in the population. Individuals have parameter sets that correspond to face color. Selection chooses individuals red, blue, red, then yellow; black is left out. Crossover blends red and yellow into dark orange and light orange. Mutation changes dark blue to light blue.

The representation of individuals determines how the operators actually change individuals. In a genetic algorithm, an individual is represented as a binary string, a series

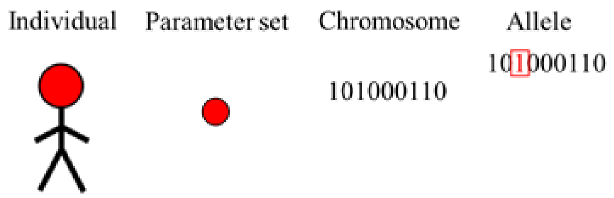

Figure 10: This hierarchy shows the binary string makeup of an individual in a genetic algorithm. The 1's and 0's of the chromosome decode to the parameter set of red. An allele of the chromosome is the "1" given in red.

of 0’s and 1’s (Figure 10). This representation is an individual’s genotype, the binary string is its chromosome, and each 0 or 1 is an allele. Genetic algorithms are named as such because they use this chromosomal representation.

3. Methods

3.1 The Agent-Based Model

An ABM was developed to simulate the eating behavior of a single larva in a circular container. The model was compared to experimental video recordings from Hu’s team of 10 circular petri dishes, each containing one larva and one source of food (3). A comparison of the experimental setup and the simulation world is shown in Figure 11.

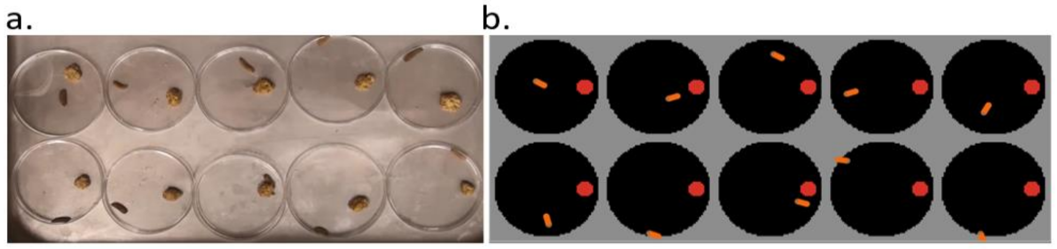

Figure 11: (a) The experimental setup is shown. (b) The simulation world is shown where the larva is colored orange, the food source is red, and the petri dish space is black.

The larva model was developed in Netlogo. In the model, one agent corresponds to one larva with a head that defines its direction of movement, colored brown for the head and orange for the rest of the body in Figure 11b. The environment is a grid-world of square patch units that contains a black, circular petri dish and a red food source. In order to create an environment that resembled the experimental setup, units were declared with a patch side length of 2 mm and a time step of 1 second. Using this framework, the ratio of the diameter of the food source to the diameter of the petri dish is roughly 20 mm to 120 mm, which in the model is equivalent to 10 and 60 patches, respectively. From experimental data, the average speed of a larva is 2 mm/second, or 1 patch/time step in the model. Neither the size nor the location of the food source changes during the experiments and simulation.

Figure 12 shows how progression of the BSFL ABM. First, the environment is set up and a larva is created. At each time step, the larva adjusts its direction, moves forward, then eats if it sees food. The simulation ends after 3600 time steps, which corresponds to 1 hour to match the physical experiment. The rules of the ABM are wiggle, align, move, and eat. The parameters of the ABM are wiggle-amount, wiggle-often, max-align-turn, vision, and food-xcor.

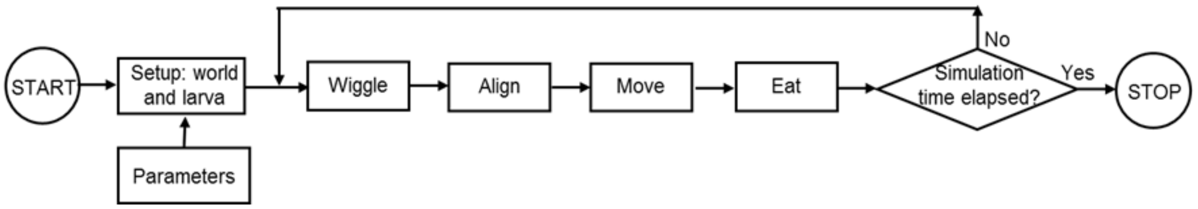

Figure 12: This flowchart shows the progression of the BSFL ABM code. Boxes denote steps, and diamonds denote branches in the flow. Setup is the first action and is based on parameters.

Wiggle: At each time step, a larva's direction is reset to a Gaussian distribution function centered at its current direction with a standard deviation of wiggle-amount (Figure 13) (9, 17). This adjustment takes place with a probability given by wiggle-often.

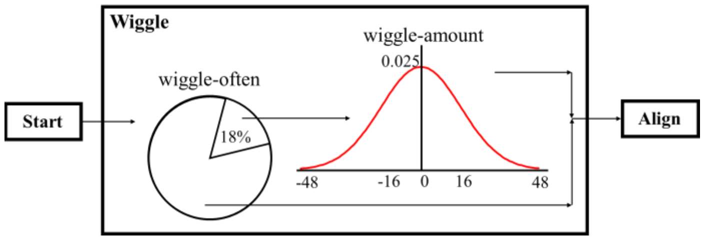

Figure 13: This flowchart depicts the wiggle procedure. Here, wiggle-often is 18% and wiggle-amount is 16 degrees.

Align: Experimental evidence suggests that a larva tends to follow the boundary of an object, no matter if this object is a petri dish wall or food source (3). To code for this behavior, larva turns parallel to the nearest object it sees by max-align-turn degrees (11). Whether a larva can see the boundary is determined by the vision parameter. Although the sensing extent of BSFL is yet to be definitively determined, a simulated larva can sense the food source and wall of the container up to a radius of vision but not the 100 degrees directly behind it. Figure 14 shows a flow diagram of the align method, which is similar to the align method of Flocking in Figure 7.

However, instead of Equation 3 for H_μ in Flocking, this align rule uses H_p, which is determined by evaluating this inequality

$$|(H_o + 90) - H_c| < |(H_o - 90) - H_c| \tag{5}$$

where H_o is the heading to the closest wall or food area in range of vision and H_c is the heading of the agent executing the rule. If this inequality holds, $H_p = (H_o + 90)$. Otherwise, $H_p = (H_o - 90)$.

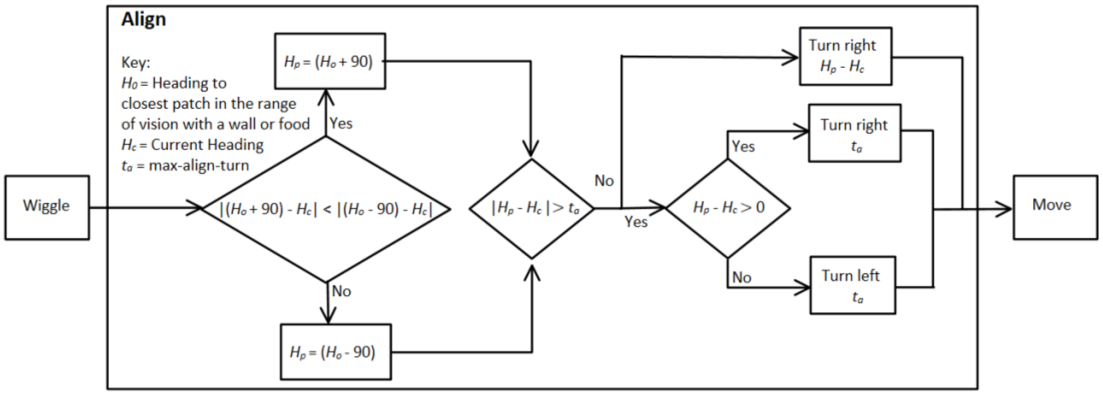

Figure 14: Flow diagram of the mathematical reasoning in the align method.

Move: At each simulation time step, the larva moves unless it is already eating (Figure 15). Its speed is given by a Gaussian distribution function centered at 1 patch per time step with a standard deviation of 0.15. The larva cannot escape the petri dish nor enter the food source, and it avoids these obstacles by the minimum angle necessary. Figure 15 elaborates on the Move rule.

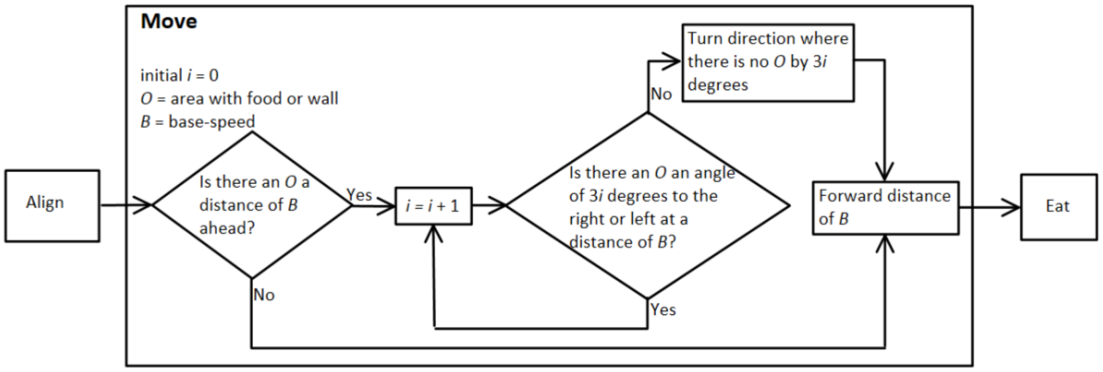
Move
initial $i = 0$
O = area with food or wall
B = base-speed
Align
Is there an O a distance of B ahead?
Yes
No
$i = i + 1$
Is there an O an angle of $3i$ degrees to the right or left at a distance of B?
No
Yes
Turn direction where there is no O by $3i$ degrees
Forward distance of B
Eat

Figure 15: Flow diagram for move. In the rare case that an agent can turn both left and right, the agent randomly selects one direction to turn.

First, the agent locates areas with an obstacle at a distance of base-speed, *B*. If there is no obstacle, the agent moves forward. Otherwise, the agent checks left and right for areas with no obstacle. The agent keeps checking for an available area to turn until it finds a non-obstacle occupied area. Then, it turns in that direction and moves forward base-speed, *B*.

Eat: During the total simulation time, the larva "eats" at multiple instances. As shown by the red line in Figure 16, the duration of each one of these instances is given by a Gaussian distribution function centered around 180 time steps with a standard deviation of 240 steps. However, eating time cannot be negative, so negative values are set to 1. In Figure 16, the negative part of the distribution becomes dotted, and the cumulative area is a red dot. The mean and standard deviation were experimentally determined (3).

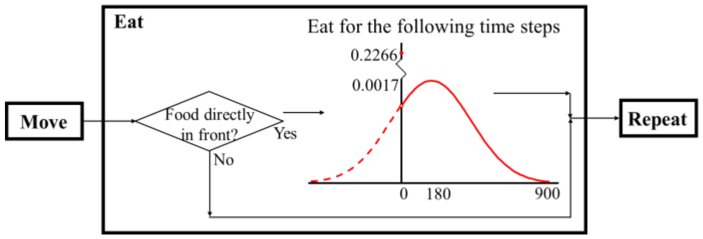

Figure 16: The probability distribution of eating time where $\mu = 180$ and $\sigma = 240$. If the patch directly in front of the larvae at a distance of vision is food, the larvae eats for the time steps given by the distribution

Lastly, the food-xcor parameter defines the position, on the *x*-axis, of the center of the food source. The *y*-coordinate is always set to zero. For the physical experiments examined, the food source was near the edge of the petri dish.

3.2 The Genetic Algorithm Calibration

The five parameters of the ABM code were optimized to match physical experiments of BSFL. Since different parameter sets produce different larva behavior, the goal was to find the ABM parameters that accurately represented the experimental BSFL behavior. Genetic algorithms were appropriate for this problem because it appeared to be multimodal and a global investigation of the parameter possibilities was desired.

Fitness of a parameter set was quantified by examining the total experimental time a larva spent eating (26%), on the wall (62.9%), and in the middle (11.1%) (3). The fitness function combines these criteria as *f*, *w*, and *m* respectively, which are percent times during a simulation run, as

$$fitness = \frac{\left(\frac{f-0.26}{0.26}\right)^2 + \left(\frac{w-0.629}{0.629}\right)^2 + \left(\frac{m-0.111}{0.111}\right)^2}{3} \quad (5)$$

inspired by (18). This fitness function quantifies error of the simulation in replicating the physical experiments, and the goal is to minimize this function. Because of randomness in the wiggle and eat rules, the fitness of a parameter set is determined by averaging the fitness function over 10 simulation runs, each 3600 time steps.

Table 1 shows the range of each parameter for the optimization. These ranges mostly represent all feasible values, but some bounds are further constrained. Vision has no maximum value except in the context of the larva, which are likely incapable of sensing far, so the maximum value of vision is low. To force larva to align to boundaries and to move somewhat randomly, the minimum values of max-align-turn and wiggle-amount are increased. Lastly, food-xcor is constrained from both sides to set the food source near the edge of the petri dish, as in the physical experiment. With the five parameters of discrete values, there are 30 million combinations of parameter sets.

Table 1: Range of parameters for genetic algorithm

vision	max-align-turn	wiggle-amount	wiggle-often	food-xcor
1.5 – 5	10 -100	10 -100	10 – 100	18-22

To carry out the search, the toolkit BehaviorSearch was used, since it is integrated with NetLogo (19). The settings of the genetic algorithm follow: population size 50, tournament style selection, tournament size 3, single point crossover, crossover rate 0.7, mutation rate 0.03, Gray code representation, and fitness caching used. These settings are default for BehaviorSearch but found accurate results, so adjusting the genetic algorithm was unnecessary.

4. Results and Discussion

An ABM was created to simulate BSFL eating behavior as compared to a physical experiment. The results of the initial ABM, run for 3600 time steps, are shown in Figure 17. The agent is randomly placed in the environment with a random heading. After 3600 time steps, the pathway that the agent traveled is shown in white (Figure 17c). In order to match the simulations to the experimental data, the parameters of the ABM must be optimized.

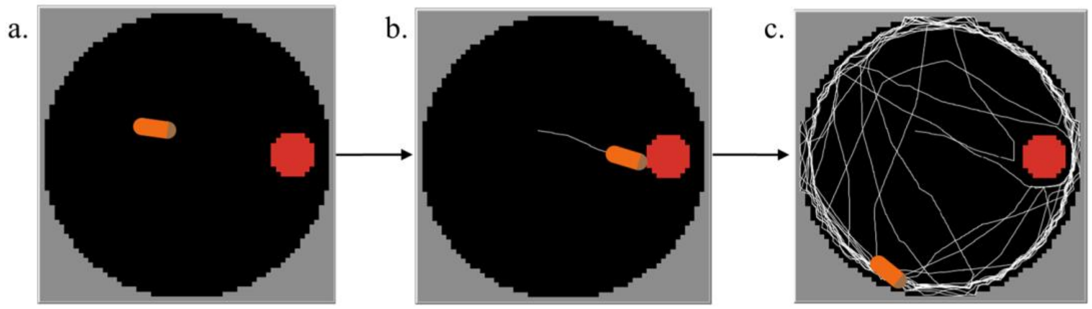

Figure 17: (a) The original setup of the environment where the agent is randomly placed in the environment with a random heading. (b) The agent is running the eating procedure where the white line represents a trace of the movement of the agent. (c) After 3600 time steps, the model ends. The agent ended while in the align procedure, as the agent is parallel to the border. Also, the lines along border are the thickest, revealing the general tendency to follow walls.

A genetic algorithm optimized the ABM parameters to match larvae behavior to the physical experiment. Figure 18 shows the progression of the genetic algorithm optimization, as the fitness of the best individual becomes increasingly better. The stopping condition of 10000 model runs

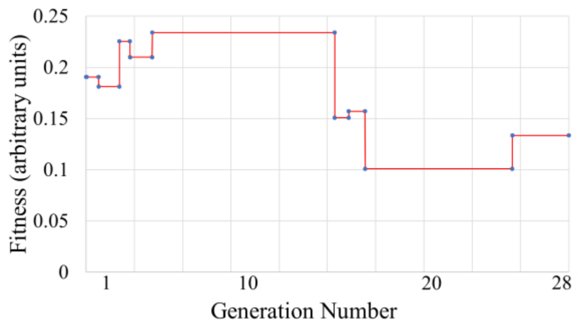

Figure 18: This graph depicts the genetic algorithm optimization. The fitness of the best individual is plotted against the progress in the search, as measured by generation number. Fitness generally becomes better (lower).

equated to 28 generations. The end result of the genetic algorithm optimization is the parameter set given in Table 2.

Table 2: Parameter values from the genetic algorithm optimization

Parameter	**Optimized Value**
wiggle-amount	16 degrees
wiggle-often	18 degrees
max-align-turn	83 degrees
vision	3 patches
food-xcor	22 patches

In order to demonstrate the accuracy of this parameter set, experimental data from the videos was quantitatively compared to simulation data averaged over 100 trials. In Table 3, experimental and simulated times, as a percentage of the total time, that a larva spent eating and on the wall are compared. Clearly, the simulation reproduces the physical experiment well, with an error that is at most 5% the physical data. This is promising but not surprising given that the model was optimized with this data.

Table 3: Comparison of experiment and simulation.

	Physical experiment	Simulation
Time eating	26.0%	24.8%
Time on wall	62.9%	62.0%

To evaluate the robustness of the model, the frequency of eating was examined, which the model was not explicitly optimized for. Frequency of eating is the number of times a larva found the food, while time eating is the total time a larva spent near the food after finding it. The frequency of eating physically and simulated is depicted in Figure 19, and these selected graphs qualitatively show that the large spread seen for the physical data is captured by the simulation. In these graphs, 1 means eating and 0 means not eating, so the number of peaks is the frequency of eating. Quantitatively, a larva ate an average of 6.5 times for the experiment and 4.25 times for the simulation; this 35% error seems acceptable accuracy for modeling the behavior of a living creature.

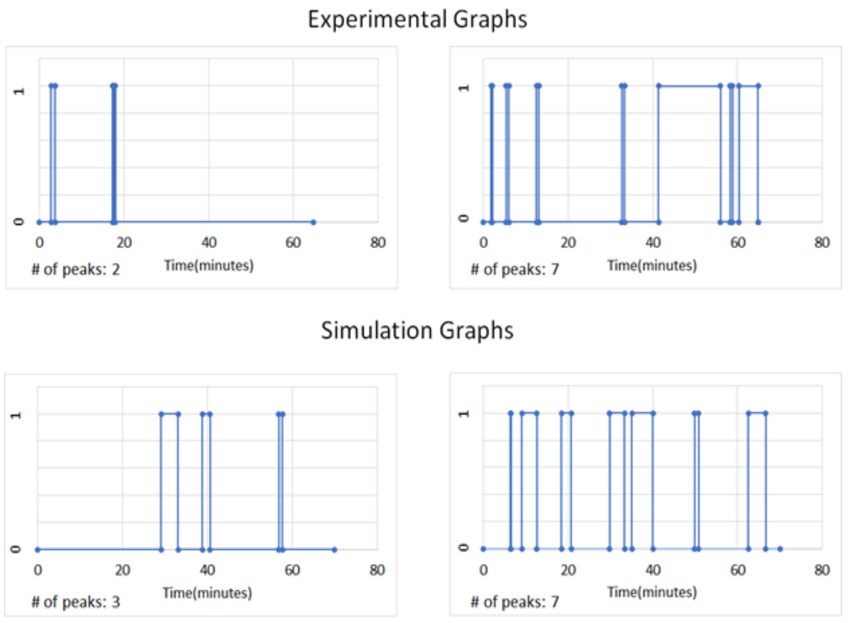

Figure 19: A comparison of selected physical and simulated graphs of eating frequency shows qualitative similarity. For both physical experiment and simulation, two example graphs are shown; each graph corresponds to a different experiment or simulation. Eating is represented by 1, whereas not eating is represented by 0. The observation time was 70 minutes. Eating frequency is the number of peaks and is listed at the bottom left of each graph.

Given that the model reproduces the physical experiment, inferences about BSFL can be made from the parameter values found, but some limitations of the model should be acknowledged. Other solutions of parameter values exist. A genetic algorithm tends to find several good solutions and does not guarantee any given solution is absolutely optimal. Other possible solutions are given in Table 4, and extreme parameter values are highlighted in red. Since the model is stochastic, a 95% confidence interval for the true fitness of a given set was constructed; after 1000 simulations, the given fitness is accurate to about ±0.01 for all the sets. The highlighted parameter set was chosen somewhat arbitrarily with a preference towards strong boundary following behavior. Comparison of these solutions is not rigorous but suggests that that wiggle-often and max-align-turn can vary significantly, whereas wiggle-amount and food-xcor vary much less. An additional limitation of the model is based on the physical experiment. It is frivolous to reduce the error of time eating and on the wall given the larvae variance in the physical experiment; the true time eating and on the wall of larvae are estimates. Furthermore, another video of 10 larvae in a petri dish, but with food in the center not the edge, resulted in the larvae eating 45% of the time. However, this experiment was performed with a different batch of larvae, and it is reasonable to expect that foraging behavior depends on a larva's growth stage, which is only 14 days. Despite the different possibilities for parameter values, speculation follows on their meaning for BSFL behavior.

Table 4: Alternative parameter sets for the ABM

Parameter set	vision	max-align-turn	wiggle-amount	wiggle-often	food-xcor	Fitness, ±0.01
1	3	83	16	18	22	0.15
2	2.75	85	15	10	21	0.13
3	2.75	50	13	30	20	0.15
4	2.75	55	7	90	21	0.15
5	2.5	45	14	20	20	0.16

For parameter set 1, the value of wiggle-amount and wiggle-often are small, 16 degrees and 18% respectively, which indicates that with little randomness, a larva can come off the wall and find the food. Thus, only slight randomness is required to reproduce the experimental behavior observed. The high value of max-align-turn, 83, indicates that once a larva is close to an object or boundary, it aligns and follows the object or boundary tightly. One can speculate that a larva's limiting sensing capability is offset by this alignment behavior where it closely follows whatever object it senses, providing a source of direction. A vision value of 3 means that a larva can detect objects that are at most 3 patches away. Because the size of a larva in the model is 10 patches, each larva can sense around 30% of its body length. It is conceivable that a larva is able to sense food but only sometimes chooses to seek it out. In addition, it is possible that a larva is more inclined to seek out food when other larvae are present; this model does not currently consider this possibility. The value of food-xcor, 22, indicates the close proximity of the food source to the wall. According to the model, a larva is more likely to find the food if it is closer to the wall. This is because the larva follows the wall of the circular container closely and cannot sense the food if it were to be placed in the center of the container. However, other video data suggests the opposite trend; investigating this difference is part of future work.

5. Conclusion

An ABM was developed to model the behavior of a Black Soldier Fly larva in a circular petri dish. The parameters of the model were optimized to match experimental data using a genetic algorithm. It was found that the model is capable of quantitatively reproducing the experimental measurements regarding time spent on the wall of the container, time spent eating, and frequency of eating. The latter measurement was not used during calibration, and that the model is capable of reproducing it indicates robustness. The parameter set found by the genetic algorithm is most likely not the best one, and more experimental data will be needed to further optimize the parameter set. In particular, data regarding BSFL foraging behavior at different stages of growth would be needed.

Nonetheless, the methodology used here is promising, and it paves the way for the development of an ABM that investigates the role that heterogeneity, regarding growth stage, and local physical interactions, play in the emergent patterns that BSFL form around food sources. A natural extension to this work is to apply this model to determine the collective behavior of multiple larvae in a simulation world. Such a model, in conjunction with experiments, could help in optimizing the placement of feeding stations in BSFL farms.

6. References

1. *Conversion of organic material by black soldier fly larvae: establishing optimal feeding rates.* **Diener S, Zurbrugg C, Tockner K.** 2009, Waste Manag Res, pp. 603-10.

2. *Black Soldier Fly Biowaste Processing - A Step-by-Step Guide*. **Dortmans B. M. A., Diener S., Verstappen B. m., Zurbrügg C**. 2017, Eawag: Swiss Federal Institute of Aquatic Science and Technology, Dübendorf, Switzerland.

3. *Black Soldier Fly larvae consume food by actively mixing.* **Shishkov, O., Hu, M., Johnson C., Zhang, B., Hu., D. L.** Tokyo : s.n., 2018. 7th International Symposium on Aero-aqua Bio-Mechanisms.

4. *The principles of collective animal behaviour*. **DJ, Sumpter.** 2006, Philosophical Transactions of the Royal Society of London B: Biological Sciences., pp. 5-22.

5. *Three-dimensional trajectories and network analyses of group behaviour within chimney swift flocks during approaches to the roost.* **Evangelista DJ, Ray DD, Raja SK, Henrick TL.** 2017, Proc R Soc B.

6. *Fluid dynamics of bacterial turbulence.* **Dunkel J, Heidenreich S, Drescher K, Wensink HH, Br M, Goldstein RE.** 2013, Physical review letters.

7. *Agent-based and individual-based modeling: a practical introduction.* **Railsback, Steven F. and Grimm, Volker**. 2012, Princeton University Press.

8. Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

9. Wilensky, U. (1997). NetLogo Termites model. http://ccl.northwestern.edu/netlogo/models/Termites. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL

10. *Flocks, herds and schools: A distributed behavioral model*. **Reynolds, Craig.** 1987, Computer Graphics, pp. 25-34.

11. Wilensky, U. (1998). NetLogo Flocking model. http://ccl.northwestern.edu/netlogo/models/Flocking. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

12. **Mitchell, M.** *An Introduction to Genetic Algorithms.* s.l. : A Bradford Book The MIT Press, 1996.

13. *Finding Forms of Flocking: Evolutionary Search in ABM Parameter-Spaces.* **Forrest Stonedahl, Uri Wilensky.** 2010.

14. *Genetic Algorithm for Variable Selection*. **Pittman, Jennifer**.

15. *Genetic Algorithm Optimization Applied to Electromagnetics: A Review*. **Weile, Daniel S., Michielssen, Eric.** 1997, IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION, VOL. 45, NO. 3.

16. *Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators*. **Larranaga, P., C. M. H. Kuijpers, R. H. Murga, I. Inza, S. Dizdarevic.** 13, s.l. : Artificial Intelligence Review, 1999.

17. Wilensky, U. (1997). NetLogo Ants model. http://ccl.northwestern.edu/netlogo/models/Ants. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

18. *Facilitating Parameter Estimation and Sensitivity Analysis of Agent-Based Models: A Cookbook Using NetLogo and 'R'*. **Thiele, Jan C., Kurth, Winfried and Grimm, Volker**. 2014 Journal of Artificial Societies and Social Simulation 17 (3) 11 . doi: 10.18564/jasss.2503

19. Stonedahl, F., & Wilensky, U. (2010). BehaviorSearch [computer software]. Center for Connected Learning and Computer Based Modeling, Northwestern University, Evanston, IL. Available online: http://www.Behaviorsearch.org