Agent based modeling of *Eciton burchelli* swarm patterns

Jose Almora[*], Albert Izarraraz[†], Qiao Liang[‡], Crystal Nesmit[§]
David Murillo[¶]

Abstract

Eciton burchelli are a unique species of army ants that demonstrate distinct swarm behavior. An important common characteristic of these ants is their extremely poor eyesight causing them to rely heavily on the distribution and tracking of pheromone to survive. We create a simulated ant world to observe their reactions to different initial food distributions and determine the stability their swarm patterns to perturbations and external factors. The goal is to recover the swarm behavior observed in nature using the simplest possible behavioral rules. The focus of this study is to investigate the impact of intensity and direction of the pheromone trails on the emergent behavior of ants through simulations. We chose the pheromone trail because it is a key element in the survival of the *E. burchelli* colony. We study several cases including a single food source, multiple food sources, and perturbations to the multiple food sources implemented by randomly placing barriers over established trails. In addition to the simulations, we propose a modified version of Fisher's equation to demonstrate ant swarm behavior from a mathematical approach. The basic Fisher's model has a traveling wave solution that integrates the logistic growth and the diffusion of the ants population density. We add a taxis and a removal rate into the equation to include pheromone trail drift and loss of population during a swarm.

[*]University of North Carolina Chapel Hill
[†]University of California Irvine
[‡]University of New Mexico Albuquerque
[§]St. Maru's College of Maryland
[¶]Arizona State University

1 Introduction

Swarm patterns of raiding army ants are a spectacular demonstration of emergent social behavior in nature. Different species of army ants display different predatory raiding patterns: column raiding and swarm raiding. The largest of these predatory patterns are found in the neotropical army ants, *Eciton burchelli*. Their swarm raiding formation can contain up to $200,000$ workers per raid. Raids of this size can extend as wide as 15 meters or more, and can proceed at a speed of 0.2 m/min. Within a single day, they are capable of sweeping over an area of more than 1500 m^2 and can catch around $3,000$ invertebrates per hour.

The swarm raid system in Figure 1 is composed of three parts: swarm front, principal trail and fan.

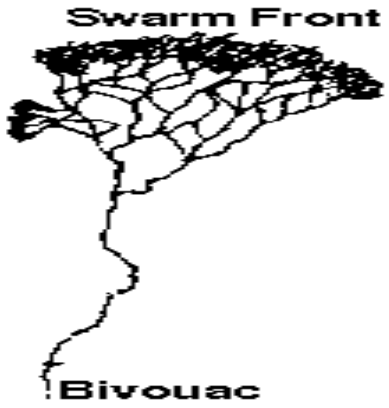

Figure 1: *E. Burchelli* swarm pattern

Eventually, most ants will concentrate at the front of the swarm, which is described as a 'dense phalanx of foragers carpeting the ground, and is the leading edge of the swarm'.[4] Connected to the back of the swarm front, a fan-shaped trail creates various links in the swarm. In the direction of the bivouac, which is a living nest constructed of the army ants, the trail sizes get larger but form fewer links. The last link of the fan connects to the principal trail, which is the permanent highway for the incoming and outgoing ants. All three transitions are repeated each day starting with the principle trail,

the fan, and swarm front. This goes on for ten to twelve hours of raiding with ants continuously leaving the bivouac to join the front, and returning any prey caught in their path. It is important to note that this is a three lane highway with the outside lanes going to the swarm front and the inside lane returning to the bivouac. There are several possible explanations for this phenomenon. The formation not only protects the incoming food, but also allows the inside lane to flow faster since they are bounded at the sides and have less possibility of straying from the trail to the bivouac.

The formation is a result of the ants making slow and steady progress at the swarm front as they pass through the terrain. The ants march a few centimeters forward, away from the swarm and then return to the swarm to avoid getting lost. This phenomenon is called the "rebound effect". Since the ants are nearly blind, as the ants swarm, they secrete a pheromone in order to communicate with each other and prevent getting lost. Initially, outbound ants deposit small amounts of pheromone to find their way back to the bivouac. When prey is found, they deposit greater concentrations of pheromone to recruit ants in the surrounding area to reinforce the trail back to the nest, and recruit additional ants at the bivouac.

Computer models are created in order to explain how raid structures of the *E. burchelli* are generated. One model shows that the characteristic patterns of army ants could be self-organized and generated from large numbers of interactions between identical foragers. This is done by analyzing pheromone trail-laying and trail-following behaviors [5]. The concentration of the pheromone evaporates over time, not only as a function of the behavior of the ant depositing the pheromone, but also as a function of the evaporation rate of the pheromone which is described in the following section. The evaporation rate is not only determined by the pheromone's chemical structure but also by external physical conditions, such as airflow.

The question we will answer is whether external factors, such as obstacles, affect pheromone trails. In order to demonstrate the resulting outcomes of obstructions on the ant swarms, we use an agent-based model. Agent-based modeling allows us to specify rules of individual ants and the interactions between them and their environment, and then use computer simulations to discover emergent properties of swarm patterns. This is detailed in section 3, after a brief overview of previous studies in section 2. Section 4 includes the results of our simulations which are then analyzed in section 5. Finally, we analyze the biological implications of our results in Section 6.

2 Previous Work

In previous studies, formulas of pheromone concentration and the change of their concentration over time have been determined. Couzin and Franks concluded that the pheromone concentration is a function of radius, r, and diffusion time, τ.[5] The equation is given as

$$C(r,\tau) = \frac{Q}{2D\pi\tau} exp\left(\frac{-r^2}{4D\tau}\right) \tag{1}$$

where Q is the amount of pheromone deposited (g/cm) and D is the diffusion coefficent which is given as $0.01\text{cm}^2\ \text{s}^{-1}$.[5] The graphs of the pheromone concentration with respect to the radius and diffusion time are shown in Figure 2. As seen in the graph, the pheromone concentration decreases as the radius of pheromone increases, and it also exponentially decreases as the diffusion time increases. We assume that ants lay the same concentration of pheromone per unit time until they reach the prey. Thus, they leave a trail of pheromone that decreases in the direction of the nest. Once the ants encounter prey, they secrete stronger concentrations of pheromone and return to their nest. As a result, the change of pheromone increases in the direction of the nest. When the number of total ants reinforcing the trail increases, there is less change of pheromone concentration over time. When an obstacle is factored in, the life span of the pheromone trail, $1/f$ is shortened, therefore f increases. As a result, the obstacle can cause the decrease of pheromone concentration.

The probability of ants moving left, P_L, or right, P_R, is related to the intensity of the pheromone concentration on left and right branches of the swarm.[?] The relationships are:

$$P_L = \frac{(k+C_L)^n}{[(k+C_L)^n + (k+C_R)^n]} \qquad P_R = \frac{(k+C_R)^n}{[(k+C_R)^n + (k+C_L)^n]} \tag{2}$$

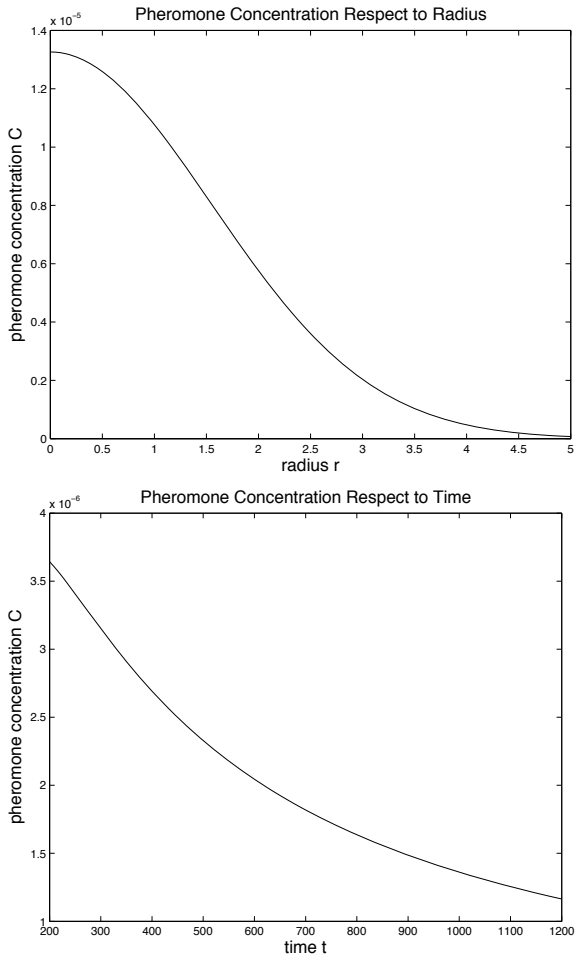

Figure 2: (a) Pheromone concentration with respect to r (b) Pheromone concentration with respect to τ

The parameter n, is the degree of nonlinearity in the choice. When n is large, the probability of ants passing through one branch with slightly higher pheromone concentration is higher. The parameter k, is the attractiveness of an unmarked branch. When k gets larger, the marking difference gets larger, and that decreases the randomness of choosing branches by pure chance. In this case the modeler chose $n = 2$ and $k = 5$.[4]

3 Agent-Based Modeling of Swarm Behavior

The *E. Burchelli* species of army ants are distinguishable from other ants because of the swarm patterns they produce when capturing prey. Due to their lack of vision, the ants predominately rely on pheromone trails. The secretion of pheromone from the ants is capable of recruiting and directing large concentrations of ants. There are different kinds of pheromone that are deposited in various situations. For instance, if an ant is being attacked it will leave a large amount of pheromone in order to attract as many nearby ants as possible to repel the attacker. Using an agent-based model, basic behavior rules of individual ants are constructed to see whether similar characteristics of the swarm patterns in nature will appear in the simulation.

Rules were created below in order to establish an accurate portrayal of the formation of swarms. These rules are necessary in order to test our agent-based model against various factors such as perturbations in the environment, as well as different reactions when the environment contains multiple food sources. The following rules produced a scaled-down model of swarm behavior.

- Movement is assigned by a command that allows the ant to move one unit forward within a range of 0 to 360 degrees. The range of movement is a control parameter.
- When searching for food, the ants exhibit a behavior called the rebound effect. The procedure entails the ants leaving the nest for a set amount of steps, while dispensing pheromone with each step. If food is not found, they will return to the nest following the pheromone that was dispensed earlier. The pheromone trail will eventually evaporate so the ants have a limited amount of time before they are no longer able to find their way back.

- If a food source is discovered a new pheromone is introduced into the system and is only deposited when ants find a food source.
- When returning with food, the ants lay another type of pheromone which communicates to other ants that there is a food source ahead. This creates a trail that is constantly being reinforced by outgoing and incoming ants.

A random heading is given to the group that assigns their initial angle of their direction because we are not trying to program the ants but set conditions where choices are arbitrary. Including the random value 0 to 360 allows the *E. Burchelli* ants to search for food together in a clockwise direction since it is the same type of behavior they exhibit in nature.

The concentration of the initial food pheromone is higher than other pheromone so it will attract more ants quickly. This is critical for the ants when confronting large prey since some prey are capable of either defending themselves or escaping.

The ants have a natural tendency to avoid overcrowding when a particular area around a food source is densely populated. They will branch out around the food source, consuming it more efficiently. The ants will maintain a linear path when pheromone are deposited consistently along the path leading to and from the food source. The pheromone at the food source act as an attractor causing the ants to diffuse once they reach it.

We initially modeled the ants' behavior when there is only one food source. Later, simulations were executed when the environment contains multiple food sources and when obstacles disrupt the pheromone trails. The obstacles were placed randomly after a fixed amount of time to observe whether the ants were able to reconstruct the altered trail.

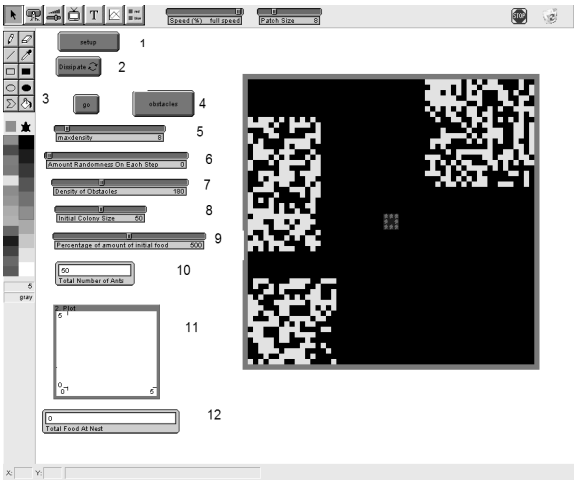

Figure 3: Starlogo Interface

The Starlogo interface contains the following buttons:

1. Setup Button - creates the environment and also the amounts of ants in the system
2. Dissipate button - when activated, it will constantly reduce the amount pheromone on the trial according to (1)
3. Go Button - runs the ants simulation
4. Obstacles Button - creates obstacles that are the color brown
5. Max Density Slider - selects value when ants move away from the group to reduce the density
6. Amount Randomness of Each Step Slider - increases the range of random motion per step
7. Density of Obstacles Slider - sets number amount of random obstacles
8. Initial Colony Size Slider - sets the initial amount of ants
9. Percentage of Amount of Initial Food Slider - sets the density of food produced in the system

10. Total Number of Ants Monitor - keeps count of the current amount of ants in the simulation

11. Plot - graphically depicts the number of ants and the amount of food gathered with respect to time

12. Total Food at Nest Monitor - tracks the amount of food stored at the nest

One purpose of the simulations is to test how ants respond different configurations of food sources, and how fast they deplete a small food source versus a large one. It also shows that the swarm patterns depend on the distribution of food sources.

Initially, there were issues with the creation of the pheromone. Patch colors were assigned to various pheromone types. Red was for the outgoing trail in search of food and green was for the incoming trail with food and were in search of the nest. The problem was that the ants stamped their color every time they passed a patch. As a result, they would color over any previous patch with their own. This led to many "stranded" ants in the environment. After the initial encounter with the food source, the ants will go back to their nest and leave a green trail. This was the trail used to return while the red remained the trail to find food. As the ants were returning with food, their random direction caused them to cross the already laid, red trails. They painted them over with green which caused a broken link in the trail. Thus, any new ant that came from the nest in search for food came to an abrupt stop once their red trail was interrupted with the green patch.

To fix this disruption we eliminated any dependence of color when an ant follows a pheromone trail. Instead, we assigned values to the different pheromone. From previous research, the *E. burchelli* will leave a weaker concentration of pheromone when they are searching for food.[6] They drop less pheromone over an interval of time because they only leave enough pheromone to retrace their steps back to the swarm. Once they find food, a larger concentration of pheromone is secreted and the rate of secretion is increased. Due to this information, we assign the outgoing search for food trail and incoming returning with food trail numerical values. We enforce the incoming trail with a higher value than the outgoing trail. We put a counter on the trails where the pheromone value will decrease 1 unit for every 2 steps the ant takes after walking over a patch. At this stage we came across another problem. Every time another ant steps on the patch that a

previous ant already crossed, it resets the counter to the initial value. We set the outgoing pheromone value at 30 and the incoming trail at 50.

4 Modeling of Ants-Dynamics

In addition to the agent-based modeling, Fisher's Equation is used to describe local swarm behavior on a line when the ants initially encounter food. Fisher's Equation is the logistic population growth at (x, t) with constant diffusion model of the population,

$$\frac{\partial u}{\partial t} = ru(1 - \frac{u}{k}) + D\frac{\partial^2 u}{\partial x^2} \tag{3}$$

where u is the population density, t is time, D is the diffusion coefficient, k is the carrying capacity, and r is the intrinsic growth rate.

We choose Fisher's Equation because its propagating wave solution resembles a local swarm pattern. The logistic growth part of the model describes the growth of the ants population density as the swarm reaches the food source, and the diffusion part of the model demonstrates the branching out of the swarm once the food is discovered.

4.1 Temporal Dynamics of Model System

First, we obtain a numerical solution of the above model via Matlab. Then we analyze the change of population density with respect to time for a given distance in Figure 4(a), where each curve represents a distribution of the population density at a particular time. The figure shows an initial small ant population that progress to the food source. In a spatial homogenous case, we assume the carrying capacity $k = 1$. The population is mostly concentrated at one point, the center. When the population reaches the food, it starts to diffuse left and right. Figure 4(b) is fundamentally the same with Figure 4(a), except Figure 4(b) gives a three dimensional visual effect that can help us understand the relationship between population density, time and space. The initial small population grows very sharply until it reaches the carrying capacity. Once the population reaches the caryying capacity, it stops growing, but starts to diffuse.

4.2 Perturbation Analysis of the Model for Small Diffusion Coefficient

Besides the numerical solution of equation (4), we also derive the analytic solution from using first order perturbation analysis. It is legitimate to use only the first order perturbation because we assume the diffusion coefficient D is sufficiently small. Figure (5) is the plot of the analytic solution, which is very similar to the graph of the numerical solution. The perturbation analysis is done in Appendix 2.

4.3 Behavior of the Model in Heterogenous Space

So far, the Fisher's Equation is analyzed under the assumption that the carrying capacity for the density of ant population is a constant. Next, we make the carrying capacity a function of distance. From Figure 6(a) and 6(b), we see that the population density grows and diffuses at the same time once food is discovered. Since it is unlikely that a food source is equally distributed, it is realistic to have the carrying capacity a function of distance. In this case, we choose $k = k(x) = 5 \times |(x - .5)| + 5$. The density gets higher when the value of $k(x)$ becomes larger.

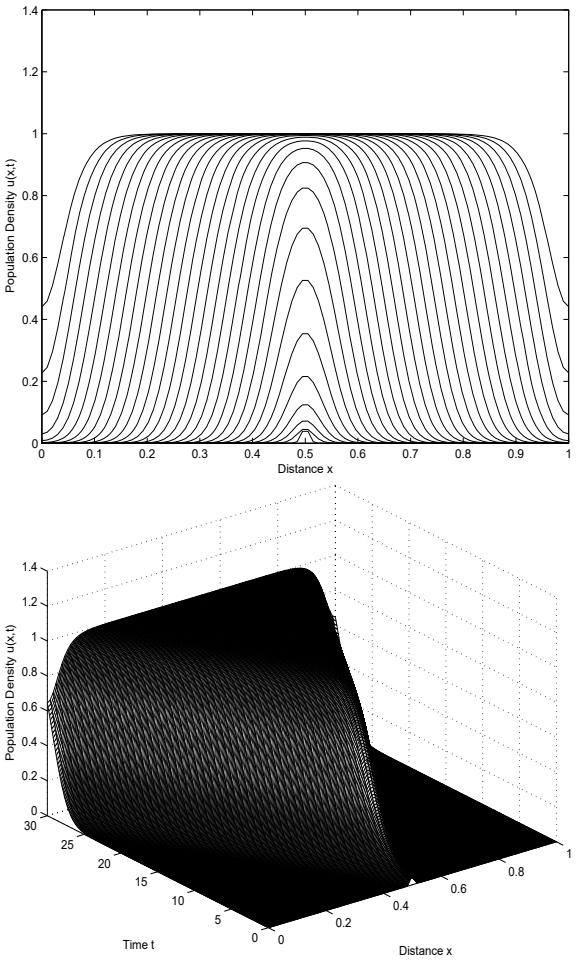

Figure 4:
(a) Evolution of the density as a function of space and time at each step of time when k=1 (b) Evolution of the density as a function of space and time when k=1

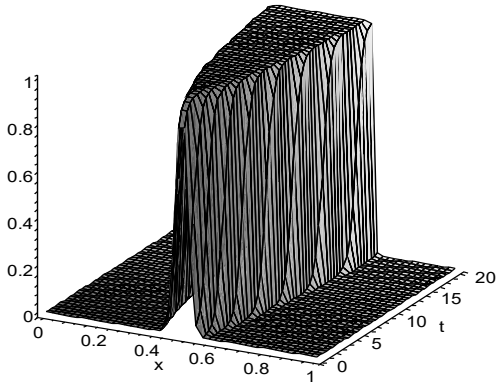

Figure 5: Analytical solution of perturbated Fisher's Equation

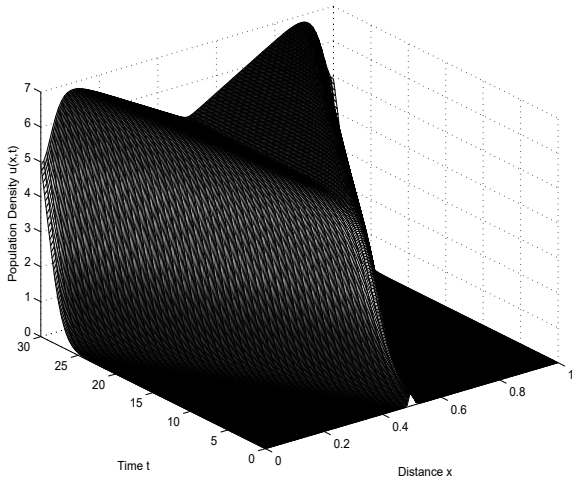

Figure 6: (a) Evolution of the density as a function of space and time at each time step when k=k(x) (b) Evolution of the density as a function of space and time when k=k(x)

5 Wave solutions of the model

In order to further analyze the spatial homogeneous Fisher model and its propagating wave solutions, a phase plane analysis is done to obtain the propagating wave speed. Base on the results from Murray's book – Mathematical Biology, we start the analysis by rescaling the equation in order to simplify calculations.[10]
Let

$$\begin{aligned} s &= At \\ z &= Bx. \end{aligned} \tag{4}$$

$$Then \quad \begin{aligned} \frac{\partial u}{\partial t} &= A\frac{\partial u}{\partial s} \\ \frac{\partial u}{\partial x} &= B\frac{\partial u}{\partial z} \\ \frac{\partial^2 u}{\partial x^2} &= B^2\frac{\partial^2 u}{\partial z^2}. \end{aligned} \tag{5}$$

Equation (3) becomes:

$$\frac{\partial u}{\partial s} = \frac{r}{A}u(1-u) + \frac{DB^2}{A}\frac{\partial^2 u}{\partial z^2}. \tag{6}$$

Let

$$\begin{aligned} A &= r \\ B &= \sqrt{\frac{r}{D}}, \end{aligned} \tag{7}$$

and write

$$t^* = s, \quad and \quad x^* = z. \tag{8}$$

then

$$\begin{aligned} t^* &= rt \\ x^* &= x\sqrt{\frac{r}{D}}. \end{aligned} \tag{9}$$

For convenience, we ignore the notation of asterisks. Now, equation (3) is reduced to

$$\frac{\partial u}{\partial t} = u(1-u) + \frac{\partial^2 u}{\partial x^2}. \tag{10}$$

To further simplify Fisher's Equation, we convert the partial differential equation into an ordinary differential equation. Assume a traveling wave solution exists, it can be written in the form:

$$\begin{aligned} u(x,t) &= U(z), with \\ z &= x - ct, \end{aligned} \tag{11}$$

where c is the speed of wave propagation. In the present case, c is the velocity of the ants branching out when food is found. Since equation (10) is invariant if $x \to -x$, mathematically, c can be positive or negative. We assume c is positive, so

$$\begin{aligned} \frac{\partial u}{\partial t} &= -c\frac{dU}{dz} \\ \frac{\partial u}{\partial x} &= \frac{dU}{dz} \\ \frac{\partial^2 u}{\partial x^2} &= \frac{d^2 U}{dz^2}. \end{aligned} \tag{12}$$

Using the same notation, we obtain a second order ordinary differential equation

$$U'' + cU' + U(1-U) = 0, \tag{13}$$

where primes denote differentiation with respect to z. Equation (13) can also be written as an equivalent system of

$$\begin{aligned} U' &= V \\ V' = U'' &= -cV - U(1-U). \end{aligned} \tag{14}$$

System (14) has heteroclinics connecting steady states $(0,0)$ and $(1,0)$ that corresponds to the bounded wavefront of the PDE equation (10).[2] Equation (10) must satisfy:

$$\begin{aligned} \lim_{x\to\infty} U(z) &= 0 \\ \lim_{x\to-\infty} U(z) &= 1. \end{aligned} \tag{15}$$

The traveling wavefront solutions of equation (10) is analyzed in the range of $0 \le U \le 1$, since $U > 1$ can be scaled down to $U \le 1$, and $U < 0$ does not have any physical meaning. Figure 7 shows a traveling wave solution goes to the right, which indicates a positive wave speed.

In correspondence between the bounded traveling wave solution of Fisher's Equation (10), phase plane trajectories are obtained from equation (13) as shown in Figure 8.[2] In addition, due to the continuity arguments, the heteroclinics exist for any c. The wavefronts correspond to the heteroclinic phase curves which is a separatrix from saddle to node.[2] Wave system (14) allows us to study the phase pane trajectories, which are the solutions of

$$\frac{dV}{dU} = \frac{-cV - U(1-U)}{V}. \tag{16}$$

First we find the Jacobian of (14),

$$J = \begin{pmatrix} -c & 2U-1 \\ 1 & 0 \end{pmatrix}. \tag{17}$$

Then we calculate the eigenvalues λ:

$$det \begin{pmatrix} -c-\lambda & 2U-1 \\ 1 & -\lambda \end{pmatrix} = 0 \Rightarrow \lambda = \frac{-c \pm \sqrt{c^2 - 4(-2U+1)}}{2}. \tag{18}$$

We can calculate the eigenvalues λ at the steady state (0,0),

$$\lambda_{1,2} = \frac{-c \pm \sqrt{c^2 - 4}}{2}. \tag{19}$$

If $c^2 > 4$, the steady state is a stable node, since the corresponding eigenvalues are real and negative. If $c^2 < 4$, the steady state is a stable spiral, because both eigenvalues are complex with negative real parts. The eigenvalues at steady state (1,0) are

$$\lambda_{1,2} = \frac{-c \pm \sqrt{c^2 + 4}}{2}. \tag{20}$$

In this case, one eigenvalue is positive and one eigenvalue is negative, so the point (1,0) is a saddle point.

In order to have a real and positive population density, the wave speed must be greater than or equal to 2. It is biologically unrealistic if $c < 2$, since this indicates oscillations around $U = 0$, which means that the population density becomes negative. The same analysis is applied to the original Fisher's Equation (3) to find its wave speed,

$$c \geq c_{min} = 2\sqrt{rD}. \tag{21}$$

The dependence of the wave speed c on the initial conditions at infinity can also be determined. Since u is very small, u^2 can be neglected. Then equation (10) can be linearized

$$\frac{\partial u}{\partial t} = u + \frac{\partial^2 u}{\partial x^2}, \tag{22}$$

and its initial condition is defined as

$$U(x, 0) = Ae^{-ax}, \quad x \to \infty, \tag{23}$$

where $a > 0$ and $A > 0$ are arbitrary. The traveling wave solutions are now in the form of

$$u(x,t) = Ae^{-a(x-ct)}. \tag{24}$$

This solution is the leading edge form of the wavefront solution. Substitution of the last expression into equation (22) gives the dispersion relation between c and a,

$$c = \frac{1}{a} + a. \tag{25}$$

It is apparent that $c_{min} = 2$ implies $a = 1$, but the value of the wave speed when $c > 2$ must also be determined. Consider $min[e^{-ax}, e^{-x}]$ when $a < 1$. If

$$a < 1, \quad e^{-x} < e^{-ax}. \tag{26}$$

In this case, the speed of propagation with asymptotic initial condition behavior depends on the leading edge of the wave. Therefore, the wave speed c is given by (25). If

$$a > 1, \quad e^{-x} > e^{-ax}. \tag{27}$$

The speed of the propagation in this case is bounded by $c_{min} = 2$. In conclusion, the asymptotic wave speed of the traveling wave solution of equation (22) can be written as:

$$\begin{aligned} c = a + \frac{1}{a}, \quad & 0 < a \leq 1, \\ c = 2, \quad & a \geq 1. \end{aligned} \tag{28}$$

where $c = 2$ gives stable wavefront.

6 Model with taxis

Due to the attraction food source has on ants, a drift of the population density can occur. Consider a taxis term in the spatial homogeneous Fisher's Equation to describe the drift,

$$\frac{\partial u}{\partial t} = ru(1-u) + (\alpha + \beta u)\frac{\partial u}{\partial x} + D\frac{\partial^2 u}{\partial x^2}, \tag{29}$$

where α is the drifting velocity, β is the density dependent taxis.[2] As shown in Figure 9, the initial small population does not grow as sharply as before. Instead, it grows and drifts at the same time.

Same as in the previous section, we find the wave system of (29),

$$\begin{aligned} U' &= V \\ V' &= \frac{-(c+\alpha+\beta U)V - rU(1-U)}{D}, \end{aligned} \tag{30}$$

and search for heteroclinics of equilibria $(0,0)$ and $(1,0)$. Then, by determining the eigenvalues, we find (0,0) to be stable and (1,0) to be saddle. Figure 10 graphs the phase plane trajectories around the equilibrium points. Because (0.0) is a stable spiral sink, trajectories go toward it; on the other hand, $(1,0)$ is a saddle point, trajectories go away from it, except along the stable directions. Notice that the heterochinics connects the two equilibriums.

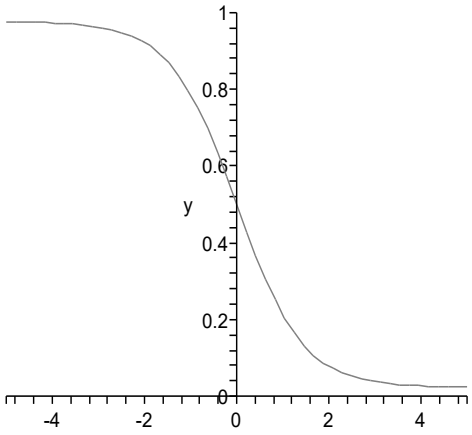

Figure 7: Traveling wave solution – wavefront

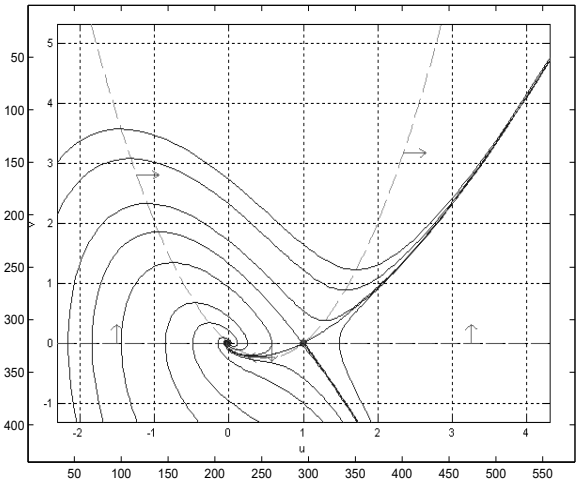

Figure 8: Heteroclinics of the wave system from system 14, c=1

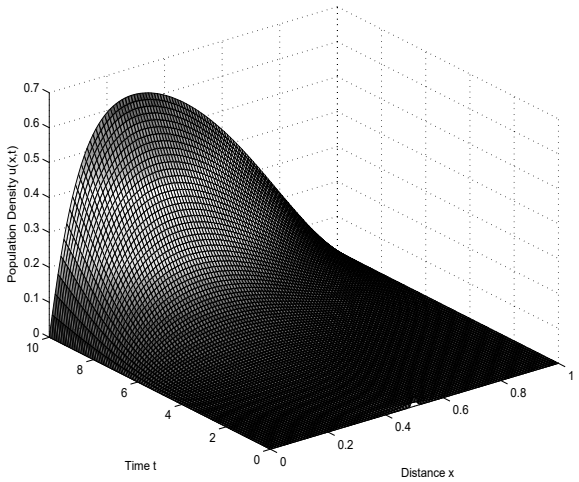

Figure 9: Evolution of the density with taxis as a function of time and space

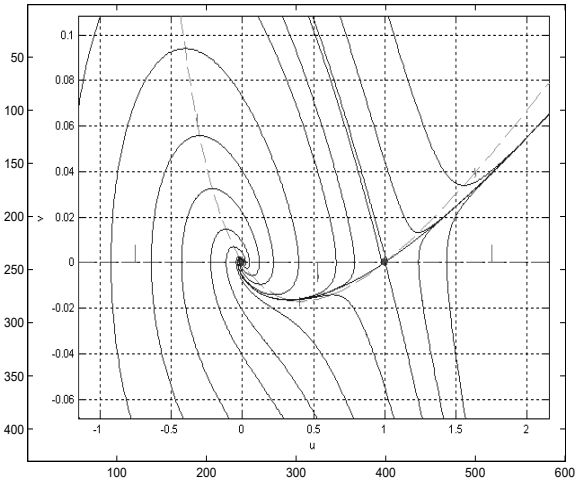

Figure 10: Heteroclinics of the wave system (30), $c = 0.001$, $\alpha = 0.1$, $\beta = 0.1$, $r = 0.01$, $D = 1$

In order to have non-negative population density, the wave speed c must be greater than or equal to $2\sqrt{rD} - \alpha$. Since u is small, between 0 and 1, u^2 is even smaller that can be neglected. Meanwhile, the slope of the wavefront solution is very small near $z = 0$, $\beta u \frac{\partial u}{\partial x}$ is also very small that can be neglected. Therefore, equation (29) can be linearized

$$\frac{du}{dt} = ru + \alpha \frac{\partial u}{\partial x} + D \frac{\partial^2 u}{\partial x^2}. \tag{31}$$

We propose the same initial condition from the basic Fisher's Equation,

$$u(x, 0) = Ae^{-ax}, \quad as \quad x \to \infty. \tag{32}$$

The solution once again can be written in the form

$$u(x, t) = Ae^{-a(x-ct)}. \tag{33}$$

By substituting the above solution into (31), we find the wave speed

$$c = Da - \alpha + \frac{r}{a}, \tag{34}$$

and when

$$\begin{aligned} c \geq c_{min} &= 2\sqrt{rD} - \alpha \\ a &= \sqrt{\frac{r}{D}}. \end{aligned} \tag{35}$$

In conclusion,

$$\begin{aligned} a \geq \sqrt{\frac{r}{D}}, \quad &\Rightarrow \quad c = 2\sqrt{rD} - \alpha \\ a < \sqrt{\frac{r}{D}}, \quad &\Rightarrow \quad c = Da - \alpha + \frac{r}{a}. \end{aligned} \tag{36}$$

A limit cycle occurs when $c = -\alpha$ and the eigenvalues at (0,0) are purely. The eigenvalues at $(0, 0)$ are:

$$\lambda_{1,2} = \frac{\frac{-c-\alpha}{D} \pm \sqrt{(\frac{c+\alpha}{D})^2 - \frac{4r}{D}}}{2}. \tag{37}$$

However, having a limit cycle when the population density is 0 does not have any biological meaning.

Due to various reasons, such as death, can cause population density loss during a swarm. We add a removal rate into Equation (29), and the model becomes

$$\frac{du}{dt} = ru(1-u) - \nu_1 + (\alpha_1 + \beta_1 u)\frac{\partial u}{\partial x} + D_1\frac{\partial^2 u}{\partial x^2}, \tag{38}$$

where ν_1 is the removal rate.[2] After rescaling the above Partial Differential Equation, (37) becomes

$$\frac{du}{dt} = u(1-u) - \nu + (\alpha + u)\frac{\partial u}{\partial x} + D\frac{\partial^2 u}{\partial x^2}, \tag{39}$$

where ν is the new removal rate. The wave system of (39) is:

$$\begin{aligned} U' &= V \\ V' &= (\frac{1}{D})(-U(1-U) + \nu + (c - \alpha - U)V). \end{aligned} \tag{40}$$

The steady states for this autonomous system are

$$\begin{aligned} U_- &= \frac{1-\sqrt{1-4\nu}}{2} > 0 \\ U_+ &= \frac{1+\sqrt{1-4\nu}}{2} > 0, \end{aligned} \tag{41}$$

where U_- is the stable equilibrium, and U_+ is the saddle point. Notice that the stable equilibrium is no longer located at $U = 0$, but at $U_- > 0$. The eigenvalue at this equilibrium is

$$\lambda_{1,2} = \frac{c_1 - U_- \pm \sqrt{(c_1 - U_-)^2 - 4D(1 - 2U_-)}}{2D}, \tag{42}$$

where

$$c_1 = c - \alpha. \tag{43}$$

A Hopf-bifurcation occurs when

$$\begin{aligned} c_1 - U_- &= 0, and \\ 4D(1 - 2U_-) &> 0. \end{aligned} \tag{44}$$

The bifurcation causes the appearance of a limit cycle close to the equilibrium at U$_-$. In our model, we let $\nu = D = 0.1$, and a limit cycle occurs when $c1$ is approximately in between 0.120 and 0.185. As shown in Figure 11, all local trajectories inside of the limit cycle goes outward and converge to the limit cycle; all trajectories between the limit cycle and the saddle point go inward and converge to the limit cycle. It has been proven that each limit cycle has its corresponding wave train solution.[2] We illustrate the wave train solution in Figure 12. From the figure, we can see the population density gets high and low periodically.

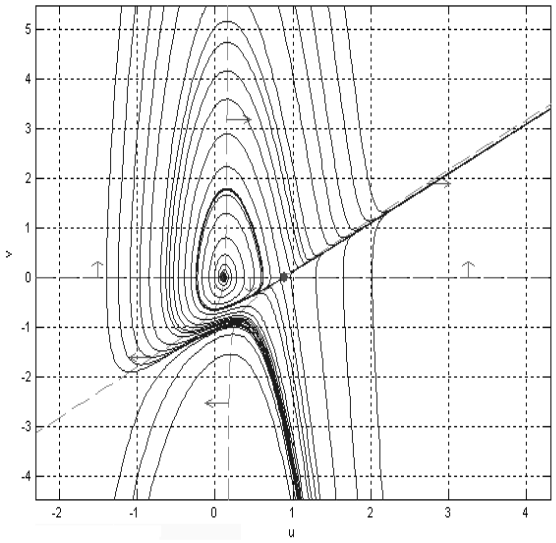

Figure 11: Limit cycle of system (40)

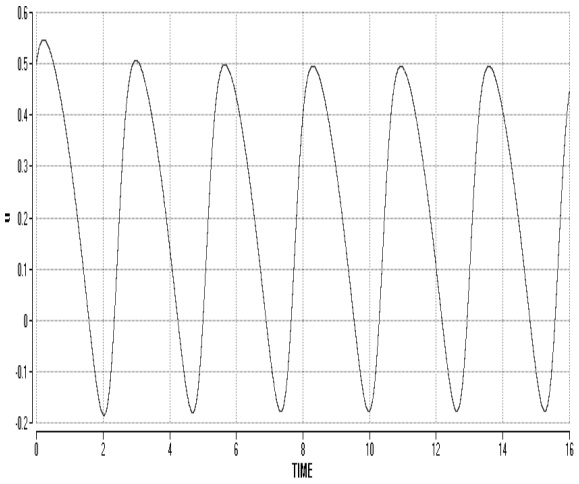

Figure 12: The wavetrain solution of model (39) corresponding to the limit cycle of its wave system (40)

We can also prove that this limit cycle from Hopf bifurcation is stable by using the stability criterion derived by Marsden and McCraken in 1976, which is also seen in Leah's book Mathematical Models in Biology.[7] Since the Jacobian at $(U_-, 0)$ is

$$\begin{pmatrix} f_V & f_U \\ g_V & g_U, \end{pmatrix} \quad \begin{pmatrix} 0 & 1 \\ \frac{-1+2U_-}{D} & \frac{c1-U_-}{D} \end{pmatrix} \tag{45}$$

We can calculate

$$W''' = \frac{3\pi}{4b}(f_{VVV} + f_{VUU} + g_{VVU} + g_{UUU}) + \frac{3\pi}{4b^2}[f_{VU}(f_{VV} + f_{UU}) + g_{VU}(g_{VV} + g_{UU}) + f_{VV}g_{VV} - f_{UU}g_{UU}] \tag{46}$$

If

$$W''' < 0, \tag{47}$$

the limit cycle around $(U_-, 0)$ is stable, which is true in the present case.

Base on Matlab simulation, when

$$\nu = D = 0.1, \tag{48}$$

and when the value of $c1$ grows toward to 0.185, the limit cycle expands and eventually tends to a homoclinic orbit that is done by separatrixes of saddle point $(U_+, 0)$. When a homoclinic orbit occurs, the period of the oscillation tends to ∞. Figure 13(a) shows the limit cycle of the wave system (40) approaches to homoclinics at $c1 = 0.185$. Figure 13(b) shows the wavetrain solution of the equation (39) as the limit cycle approaches the homoclinc orbit. This graph is done with Berkeley Madonna, and we approximate $c1$ up to 10 digits after the decimal place,

$$c1 = 0.1845173656. \tag{49}$$

As the limit cycle gets really close to the homoclinic orbit, the wavetrain solution breaks down and tend to a wave pulse shown in Figure 14.[2]

Besides the models on this report, two dimensional models can be used to describe local swarm behavior as well. For example, equation (1) is the solution of a two dimensional Fisher's Equation

$$\frac{\partial u}{\partial t} = \frac{2D}{r}\frac{\partial u}{\partial r} - \frac{r^2 D}{2Dt - r^2}\frac{\partial^2 u}{\partial r^2}, \tag{50}$$

where r is the radius of the pheromone trail. Developing a two dimensional model is rather difficult that can be continued for future research.

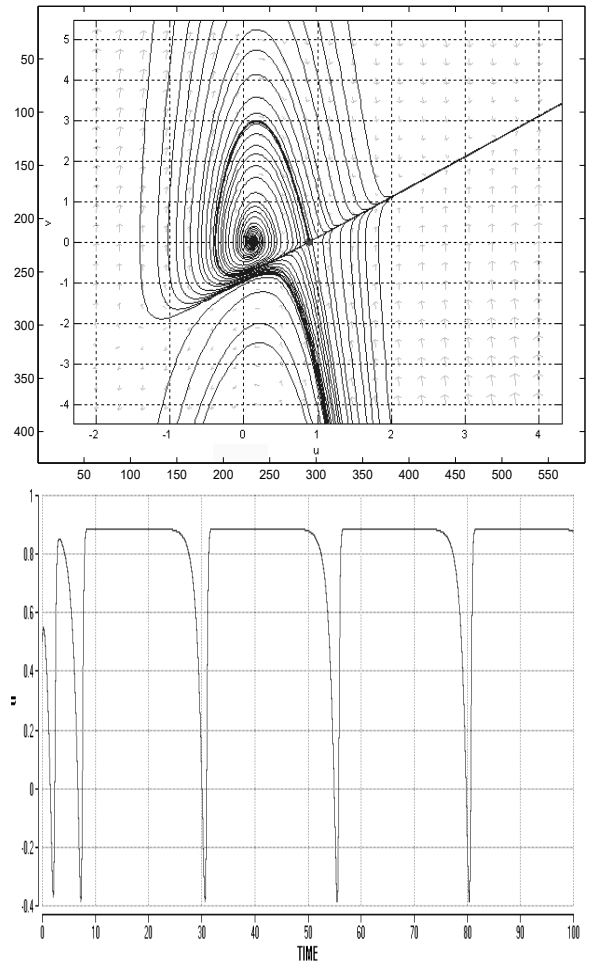

Figure 13: (a) Limit Cycle of system (40) approaches to homoclinics (b) Wavetrain solution of the model (39) approaches to wave impulse

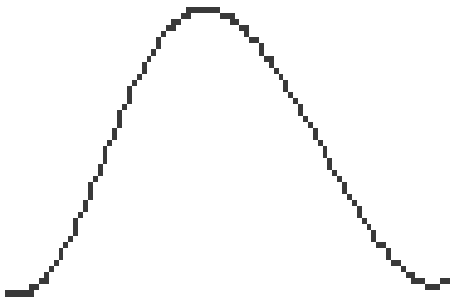

Figure 14: Wave Pulse corresponds to the homoclinic curve

7 Results

The simulations are in three main categories: single food source, multiple food sources, and multiple food sources with random placement of obstacles. The obstacles in our model represent the natural debris from local vegetation and terrain. The duration of each simulation was adjusted for five minutes. The two variables that are adjusted in these simulations are the range in which the ants can move and the direction the ants choose based on the amount of nearby ants. Commands are created to change the concentration of the factors tested. The sliders are the randomness at which the ants choose a direction and how dense different areas of the environment are with ants. The amount of randomness per step is in the order of 0, 25, 45, 90, and 180. Each degree of randomness is tested against the following densities: 1, 5, 20, and 50. Each simulation starts with 50 ants and begin from the nest.

Three different categories of simulations are run using this agent-based model. One set of simulations have an obstacle free environment with a single food source. This means that there are no obstructions in the way of getting to the food from the nest. Another simulation type contains multiple food sources and the third is multiple food sources with random obstacles appearing half way through the simulations. Results from the simulations are analyzed by taking the mean of the amount of food gathered and the surviving ants after five minutes. They are then plotted on a different densities of ants in the swarm versus the allowed random heading of the ants graph.

The testing of the single food simulation is used as our control because it did not contain any perturbations. We predicted that as the slider for randomness increased, the probability of the ants finding and creating a trail to the food source would increase up to a certain point. Unlike the prediction, the ants are able to collect the greatest amount of food when the randomness of their movement is 0 with the densities 1, 5, or 20.15 This makes sense because the ants have a higher tendency to stay on the pheromone trail leading to the food. The collection of food increased exponentially once an ant locates food. In a matter of seconds, ants that rebound will pick up the scent of the ants that find food and head towards the food source. The higher the amount of randomness per step, the higher the probability that each ant will either get confused on the pheromone trail or just not follow it. These populations gather little food and the recruitment rate is very low.

More general trends are seen when the ant environment contains multiple food sources. As randomness increases, ant populations and the amount of

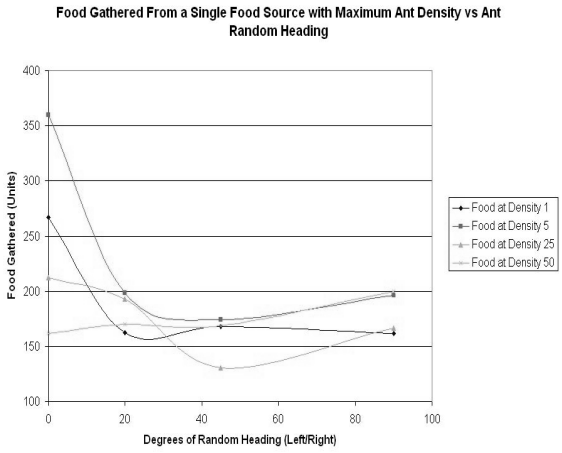

Figure 15: The amount of food gathered with respect to ant randomness

food gathered generally decrease.17 Another observation is that as the ant density increases, the ant population and food gathered remain constant. In this particular setting, two or more swarms occur frequently.18 The main difference between the two simulations are the total amount of food gathered. Ants with multiple food sources have an approximate 20% increase in food gathered over the ants in the single food simulations. This is attributed to the fact that unless the ants in the single food source setting are initially heading towards the food, they spend more time searching for food than gathering. In the situation where there are more food sources, the ants have a higher chance of finding food quickly.

The obstacles added to the multi-food environment, produce similar results. These obstacles are introduced in order to see how the ants cope with a change in their pheromone trail. The resulting graphs are compared between an environment with multiple food sources and the same environment containing obstacles. The comparison demonstrates that as the random heading of the ants increase, there is a significant decrease in the amount of food gathered.19

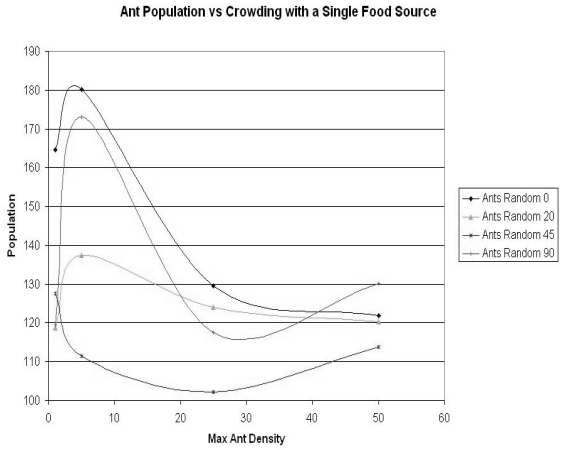

Figure 16: Change in ant population with respect to maximum ant density

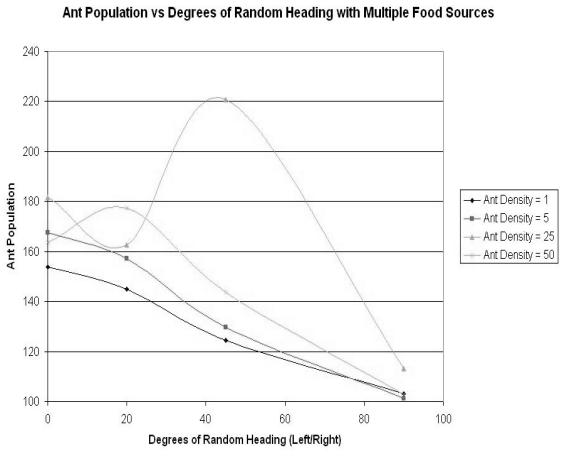

Figure 17: Change in ant population with respect to randomness

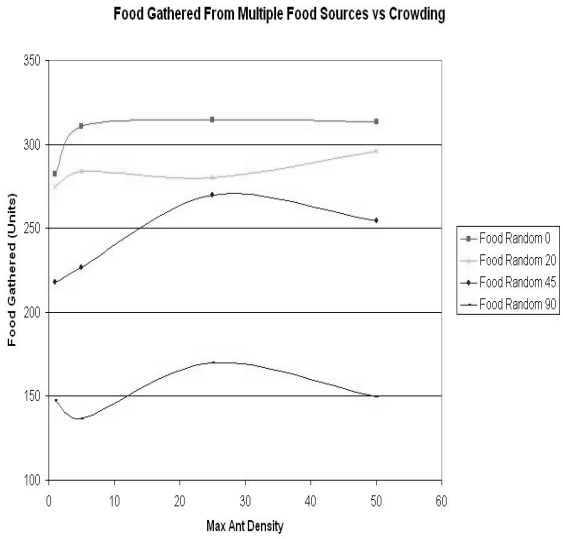

Figure 18: Amount of food gathered with respect to ant density

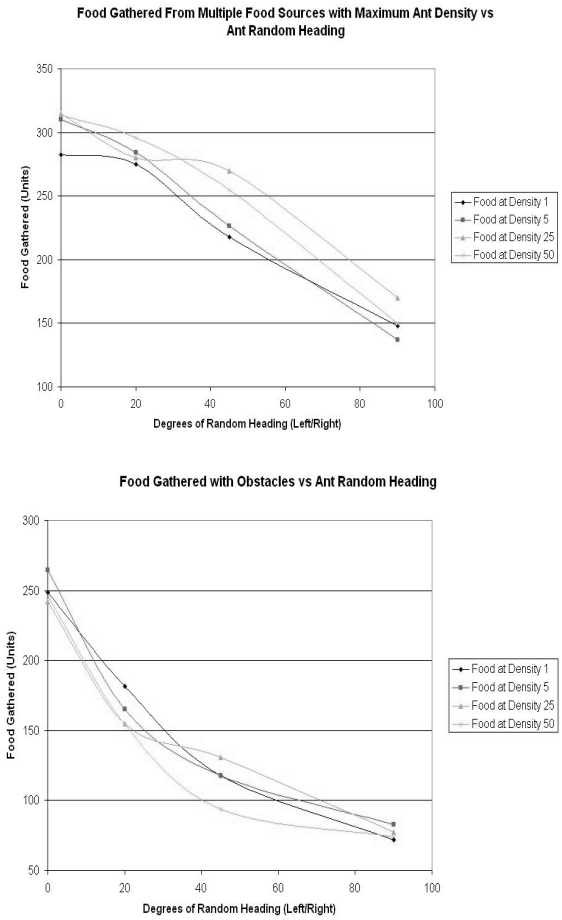

Figure 19: (a) Amount of food gather with respect to ant randomness in a multiple food environment (b) Amount of food gathered with respect to ant randomness in an environment with obstacles

This behavior leads to decreased ant populations and often extinction as random heading increases.20 When comparing the crowding of ants, the same

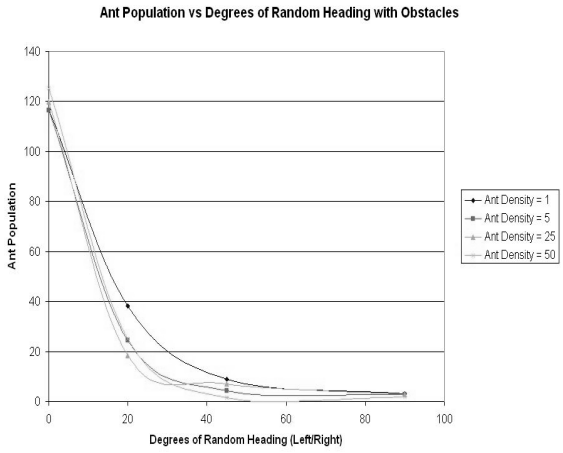

Figure 20: Ant population with respect to randomness in environment with obstacles

trends are seen, but the presence of the obstacles shift the curves down.21 We conclude that the ant population and the amount of food gathered is independent of density, but is highly dependent on randomness.

8 Discussion

The majority of the simulations indicate that the ants produce one main swarm but do not use it as its only source of mobility. The presence of sub-swarms were frequent throughout the simulations with multiple food sources. It is important to note there are degrees of randomness in the path that affect the ants. Greater efficiency is reached if the ants all swarm the food source together even though there is no guarantee that there will be enough food to sustain the colony or if food is found at all. As a result, alternate trails are vital in discovering auxillary food sources to reinforce the survival of the colony.

However, a lack of coordination in keeping the swarm together can be detrimental to the survival of the colony. The simulations demonstrate a

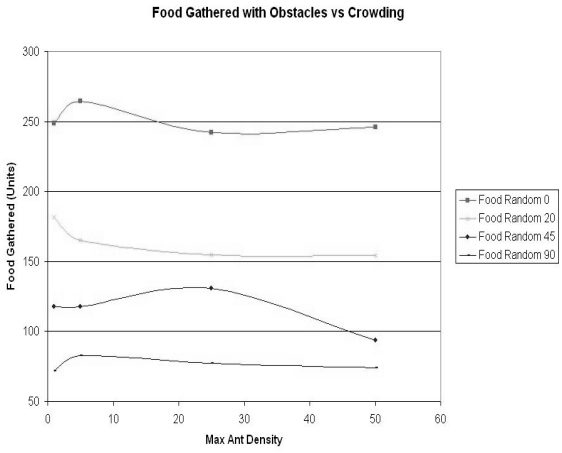

Figure 21: (a) Amount of food gather with respect to ant density in an environment with obstacles

strong correlation between an increase in randomness and decrease in population size and loss of food gathered. Increase in randomness causes the ants to break away from the swarm, which will lead to failed attempts of finding food. The strength of the swarm is in its numbers and in order to successfully acquire and consume food, the majority of the ants need to stay on the swarm trail.

Since it is critical for the ants to hunt together, the *E. burchelli* have developed pheromone chemical exchanges to reduce the randomness of their trails. This limits the proportion of ants that get lost and lose the food when returning to the nest. However, as ants search for food, a degree of random branching off from the main trail is necessary in order to find other food sources. In most cases, once a food source is found it is likely that others are nearby. Some ants will break off from the swarm in order to sweep the area close to the original food source. Essentially, the multitude of swarms and subswarms are vital the survival of the army ant colony.

9 Conclusion

Modeling a natural phenomenon such as ant swarms onto sophisticated mathematical equations and advanced simulations provide a closer look as to how

things work. The analysis can be refined to the point of quantitatively deriving the inner workings of a complex system such as an ant swarm. This project enabled us to study localized behavior of a self-organizing species.

The two approaches, modified Fisher's Equation and agent-based modeling, concentrate on separate parts of the army ant swarm patterns. The two different forms of analysis provide various insights when studying the swarm behaviors.

Agent-based modeling involves the creation of rules and then running simulations using the rules as boundaries. For swarm behavior, it creates a visual that captures the basic formation and constant reinforcement of the pheromone trails to generate swarms. Observations are emphasized on how ants interact with one another and the effectiveness of their food hunts that keep the colony alive. Adjustments are made to the randomness of each step or the maximum ants that can move in an area before they have to change direction in order to find the conditions that produce the most efficient colony. Small environmental perturbations are made in order to test the stability of the trails formed by the ants. Statistical analysis provide a more in depth understanding on how much of an effect the environmental perturbations had on the army ant's ability to gather food or follow a pheromone trail back to the nest. The data is dependent on the amount of simulations conducted and the amount of time allotted for each simulation. These factors are influential in determining the accuracy of the collected data. Agent-based modeling provides a general understanding on how a complex system works.

Fisher's Equation gives more of a quantitative understanding of the swarm front. A model is developed with parameters that correspond to the swarm behavior. Simulations of the model are easily constructed to give visual insight on traveling wave solutions. The overall dynamics of the system can be explained through the graphs. The limits of working with Fischer's Equation are that it only studies local parts of the swarm behavior and does not take in count factors such as: the reinforcement of pheromone, recruitment rate, or any behavior other then the swarm front. The data obtained from the equations is only useful to the extent of its biological interpretations. Assumptions are necessary in order to coincide with the results achieved from the simulations.

The two methods applied produce a substantial amount of information on the complexity of self-organizing systems. The approaches used can be applied to other types of self-organizing systems such as bees, termites, and fish. These studies lead to greater understanding of the systems, which pro-

vides a window of reflection based on our own behavior.

10 Future Work

Further work can be done on the agent-based model by creating competing ant colonies. *Eciton Burchelli* are capable of retreating if they sense that they are losing a fight with another colony or species. Even though rival colonies between the ants are present, it is possible that they can co-exist. By finding co-existence in a competitive model, it is possible to find equilibrium points to analyze.

External factors, such as wind and temperature, can also be added to the current model in order to intensify weather conditions on the ants. Studies show that these factors can affect the pheromone trails but little is known about the extent of which it affects the trails. These factors can alter the pheromone path in a new direction or dissipate the trail at a quicker rate. Like other animals, significant changes in temperature and precipitation can change the way of life for the species. Pertaining to ants, it is possible that extreme hot or cold conditions can alter the speed at which the ants move. As a result, this will affect the ability to successfully search and gather food. Integrating these observations with agent-based models will provide concrete data and can be analyzed mathematically.

The limits of Fisher's Equation arrive at the fact that it is one dimensional. By modifying or using a system of two dimensions, leads to the possibility of increasing the amount of analysis evaluated about the swarm front. The combinations of these techniques are useful for future work in studying other local areas of the *Eciton Burchelli*.

11 Acknowledgements

Essentially we would like to thank Carlos Castillo-Chavez for giving us the opportunity to conduct this research with the Mathematical and Theoretical Biology Institute and Los Alamos National Laboratory. This work could not have been possible without the assistance of the following people whom we would like to thank: Anthony Tongen, Linda Gao, Faina Berezovskaya, John Urrea-Roque, David Murillo, and Hugh Greenburg for all their help and support. This research is supported by grants from the Theoretical Division

at Los Alamos National Laboratory, National Science Foundation, National Security Agency, Provost office at Arizona State University, and the Sloan Foundation.

References

[1] Allen, L.J.S. 2003 *An Introduction to Stochastic Processes with Applications to Biology*, chapter 8, Pearson Education Inc.

[2] Berezovskaya,F.S., Karev, G,P. 1999 *Bifurcation of travelling waves in population taxis models*, Uspekhi Fizicheskikh Nauk, Russian Academy of Sciences.

[3] Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraula, G., Bonabeau, E., 2003, *Self-Organization in Biological Systems (Princeton Studies in Complexity)*, chapter 4, Princeton University Press

[4] Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraula, G., Bonabeau, E., 2003, *Self-Organization in Biological Systems (Princeton Studies in Complexity)*, chapter 14, Princeton University Press

[5] Couzin, I.D., and Franks, N.R. 2002, *Self-organized lane fomations and optimized trafic flow in army ants* , The Royal Society.

[6] Deneubourg, J.L., Gross, S., Franks, N.R., and Pasteels, J.M. 1989, *The blind leading the blind in army ants raid patterns: Modeling chemically mediated army ant raid patterns*, J. Insect 9

[7] Edelstein-Keshet, L. 1988, *Mathematical Models in Biology* McGraw-Hill

[8] Epstein, J.M., and Axtell, R. 1996, *Growing Artificial Societies: Social Science from the Bottom Up* The Bookings Institution

[9] Kennedy, J., and Eberhart, R.C., 2001, *Swarm Intelligence* Academic Press

[10] Murray, J.D. 1989 *Mathematical Biology*, Biomathematics, 19, $277-286$

[11] Okubo,A. and Levin, S.A. 2001 *Diffusion and Ecological Problems: Modern Perspectives*, 2^{nd} ed., Springer-Verlag New York, Inc.

[12] Resnnick, M. 1997, *Turtles, Termites, and Trafic Jams: Explorations in Massively Parallel Microworlds* MIT Press

Appendix 1

```
 ;MTBI 2004
;Jose Almora ;Alberto Izarraraz ;Qiao Liang ;Crystal Nesmith


;Reproduces the swarm pattern of army ants ;Permaters may vary to
produce different results


;Turtle Procedures

turtles-own [sum step deathcount food newh tp recruit trail x1 x2 y1
y2]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;sum -
counts the total amount of the steps the turtle  ; ;step - is useful
for the rebound effect        ; ;deathcount - lifespan of the ant
; ;food- food variable for the ant            ; ;newh - new heading
after pheromone detection       ; ;tp - temporary variable that
helps levels of pheromone ; ;recruit - allows an ant to only recruit
one more ant   ; ;trail - regulates the gradient of the pheromone
; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


to wiggle ;rand-turn is a slider ;allows user to direct how random
each ant is per step rt random rand-turn lt random rand-turn ifelse
chemical > 20    ;does not allow each ant to step on each other
    [ifelse color = magenta [leap 1]
        [fd 1]]
    [fd 1]
setxy round xcor round ycor ;rounds off to the nearest coordinate
end
```

```
to rebound ;resets steps so it can be looped again set step 0 repeat
(15 + random 10)         ;returns a random rebound back [if pc not=
yellow
    [fallon         ;die if on brown
    iwannagohome        ;follows thier own pheromone back
    ultrapath       ;if green (pher) is seen follows that instead
    white-food-path ;follows chemical pheromone if seen
    borderpatrol        ;makes sure that it will not hit brown
    wiggle          ;takes a step
    set sum (sum + 1)   ;keeps track of sum of steps
    if sum > deathcount ;kills ant if sum is greater then their deathcount
        [die]
    if food2 <= 0 and pc = yellow
;if there is not food on the patch it turns black
        [ask-patch-at 0 0 [setpc black]]]]
set where (where + 10)  ;changes current heading by 10 degrees if
color = magenta
    [seth where]
end


to looking-for-food ifelse pc = yellow      ;called if they found
food
                ;sets a different pheromone
                ;changes the color of the ant
            ;deducts food from patch and gives it to the ant
    [setchemical 30 setc turquoise counting if food2 = 9 [makecircle]
        ifelse path = 0     ;keeps in line of the trail gradient
            [setpath trail pheromone set trail (trail + 1)]
            [settrail path]
    stop]


    [if pc not= blue        ;sets a green (pher) pheromone
        [ifelse p = 0
            [setp 50 pheromone]
            [if p > .01
```

```
                    [setp (p + 10)]]
;reinforces pher pheromone

        if path = 0     ;makes the path sum of pheromone
            [setpath sum pheromone]]

    set step (step + 1)         ;adds a step, sum, and a trail per step
    set sum (sum + 1)
    set trail (trail + 1)
    fallon              ;makes sure ant dies if on brown spot
    ultrapath           ;follows the green pheromone
    youstillhere            ;changes direction if too many turtles are near
    white-food-path             ;follows chemical pheromone
    borderpatrol            ;makes sure it does not hit brown
    unstuck             ;if ant is in a loop it will delete that square
    wiggle              ;moves a square
]

if sum >= deathcount            ;kills turtle if sum surpasses their
life span
    [die]


if step >= (15 + random 10) and pc not= yellow and pc not= green
;makes sure it rebounds if it has not hit food yet
    [rebound]
looking-for-food            ;runs again if it has not hit food end


to return-to-nest wiggle                    ;moves a step set sum (sum
+ 1)           ;adds a sum per step if sum >= deathcount
;kills turtle if sum surpasses their life span
    [die]
if pc not= blue and pc not= yellow ;sets green pheromone and lays
trail gradient
    [ifelse pher = 0
        [setpher 100 pheromone]
        [if pher > .01
```

```
                [setpher (pher + 10)]
        ifelse path = 0
            [setpath trail pheromone set trail (trail + 1)]
            [settrail path]]]


if pc = yellow          ;turns ant around if it hits food
    [seth heading + 180]
if pc = blue ;hits nest, changes color back to magenta, recruits one
ant, and goes to ;procedure doitagain
    [setc magenta store
setfinaltrail trail     ;stores amount of trail of the ant birth
doitagain] fallon               ;kills turtle when on brown headback
;looks for either red or green pheromone to return to nest
borderpatrol            ;attempts to avoid brown unstuck
;deletes pheromone if stuck in a loop return-to-nest      ;runs same
subroutine again unless it hits nest end


to doitagain if pc = blue        ;ant heads back opposite way
    [seth heading + 180]

if sum >= deathcount        ;kills turtle if sum surpasses their
life span
    [die]

ifelse pc = yellow ;when it hits food, secretes chemical ;follows
subroutine of return-to-nest
    [setc turquoise counting setchemical 30 if food2 = 9 [makecircle]
            ifelse path = 0
            [setpath trail pheromone set trail (trail + 1)]
            [settrail path]
    stop]

    [if pc not= blue
;creates a red pheromone when looking for food
        [ifelse p = 0
            [setp 50 pheromone]
```

```
            [if pc = red
                [setp (p + 10)]]]
    pheromone
    set step (step + 1)     ;adds a step, sum, and a trail per step
    set sum (sum + 1)
    set trail (trail + 1)

    ;if green (pher) is seen follows that instead
    ;follows chemical pheromone if seen
    ;makes sure that it will not hit brown
    ;deletes pheromone if ant is stuck in loop
    ;moves foward a space
    ultrapath
    white-food-path
    youstillhere
    borderpatrol
    unstuck
    wiggle
]

doitagain end

;reduces the food amount of the food and gives it to the ant

to counting if food2 > 0
    [setfood2 (food2 - 1)
    setfood (food + 1)]
if food2 <= 0 and pc = yellow
    [ask-patch-at  0 0 [setpc black]]
end

;adds the amount of food stored by the ant

to store if food >= 0 and pc = blue
    [setfood2 (food2 + food)
    setfood (0)]
end
```

```
;attempts to avoid the brown squares by changing directions

to borderpatrol

if border?-at dx dy
    [rt random 45
    lt random 45 borderpatrol]
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;gives them a limit of how over crowded they can be;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


to youstillhere

if (count-turtles-at dx dy) > maxdensity [rt random 150 lt random
150]

end

;recruits one ant per ant that brings back food and ;surpasses the
min. of food

to birth if recruit = 0 [if pc = blue and food2 >= 5
    [ask-patch-at 0 0
        [sprout [setc magenta fd 1
        set deathcount (200 + random 100)
        setshape termite settrail finaltrail go]]]
setrecruit 1] end


;colors p to red and pher to green to pheromone if p >= 30 and pc
not= blue and pc not= brown [ask-patch-at 0 0 [setpc red]]
    if pher >= 50 and pc not= blue and pc not= brown
[ask-patch-at 0 0 [setpc green]] end
```

```
;creates a toggle switch that will let ;the observer window create a
white pheromone to makecircle setx xcor sety ycor settoggle 1 end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;this is so there
wouldn't be an error when obstacle occurs ;just kills turtle if they
touch it ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to fallon if pc = brown
    [die]
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;this will
help prevent them to stay stuck in a loop ;kills turtles that touch
brown ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to unstuck if 0 = sum mod 2 [ if (x1 = x2 and y2 = y1) or (x2 = xcor
and y2 = ycor)
    [if pc not= blue
        [setpher 0 set p 0 set chemical 0 setpath 0 stamp black]]]


setx2 x1 sety2 y1 setx1 xcor sety1 ycor end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;this is going from
lowest to highest ; and must have green pheromone ;a representation
of pheromone after food is found ;follows path
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


to ultrapath set newh heading settp (0)

;compares tp so it can find the highest number of trail ;makes sure
it is the green pheromone it follows ;andgels are used by the
positoin of the grid
```

```
if tp < path-at 0 1 [if 0 < pher-at 0 1 [ set newh 0 set tp (path-at
0 1)]]

if tp < path-at 1 1 [if 0 < pher-at 1 1 [ set newh 45 set tp
(path-at 1 1)]]

if tp < path-at 1 0 [if 0 < pher-at 1 0 [ set newh 90 set tp
(path-at 1 0)]]

if tp < path-at 1 -1 [if 0 < pher-at 1 -1 [ set newh 135 set tp
(path-at 1 -1)]]

if tp < path-at 0 -1 [if 0 < pher-at 0 -1 [ set newh 180 set tp
(path-at 0 -1)]]

if tp < path-at -1 -1 [if 0 < pher-at -1 -1 [ set newh 225 set tp
(path-at -1 -1)]]

if tp < path-at -1 0 [if 0 < pher-at -1 0 [ set newh 270 set tp
(path-at -1 0)]]

if tp < path-at -1 1 [if 0 < pher-at -1 1 [ set newh 315 set tp
(path-at -1 1)]]

if tp < path-at 0 2 [if 0 < pher-at 0 2  [ set newh 0 set tp
(path-at 0 2)]]

if tp < path-at 1 2 [if 0 < pher-at 1 2 [ set newh 30 set tp
(path-at 1 2)]]

if tp < path-at 2 2 [if 0 < pher-at 2 2  [ set newh 45 set tp
(path-at 2 2)]]

if tp < path-at 2 1 [if 0 < pher-at 2 1 [ set newh 60 set tp
(path-at 2 1)]]

if tp < path-at 2 0 [if 0 < pher-at 2 0 [ set newh 90 set tp
(path-at 2 0)]]
```

```
if tp < path-at 2 -1 [if 0 < pher-at 2 -1 [ set newh 120 set tp
(path-at 2 -1)]]

if tp < path-at 2 -2 [if 0 < pher-at 2 -2 [ set newh 135 set tp
(path-at 2 -2)]]

if tp < path-at 1 -2 [if 0 < pher-at 1 -2 [ set newh 150 set tp
(path-at 1 -2)]]

if tp < path-at 0 -2 [if 0 < pher-at 0 -2 [ set newh 180 set tp
(path-at 0 -2)]]

if tp < path-at -1 -2 [if 0 < pher-at -1 -2 [ set newh 210 set tp
(path-at -1 -2)]]

if tp < path-at -2 -2 [if 0 < pher-at -2 -2 [ set newh 225 set tp
(path-at -2 -2)]]

if tp < path-at -2 -1 [if 0 < pher-at -2 -1 [ set newh 240 set tp
(path-at -2 -1)]]

if tp < path-at -2 0 [if 0 < pher-at -2 0 [ set newh 270 set tp
(path-at -2 0)]]

if tp < path-at -2 1 [if 0 < pher-at -2 1 [ set newh 300 set tp
(path-at -2 1)] ]

if tp < path-at -2 2 [if 0 < pher-at -2 2 [ set newh 315 set tp
(path-at -2 2)]]

if tp < path-at -1 2 [if 0 < pher-at -1 2 [ set newh 330 set tp
(path-at -1 2)]]

seth newh lt random 15    ;gives a little bit of randomness in the
direction rt random 15 end
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;follows path from highest to
lowest; ;same comparison ;makes sure that there is some amount of
pher or p on the patch ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


to headback set newh heading settp (600)

if tp > path-at 0 1
 [if 0 < path-at 0 1
[if (0 < pher-at 0 1) or (0 < p-at 0 1) [set newh 0 set tp (path-at
0 1)]]]

if tp > path-at 1 1 [if 0 < path-at 1 1 [if (0 < pher-at 1 1) or (0
< p-at 1 1) [set newh 45 set tp (path-at 1 1)]]]

if tp > path-at 1 0 [if 0 < path-at 1 0 [if (0 < pher-at 1 0) or (0
< p-at 1 0) [set newh 90 set tp (path-at 1 0)]]]

if tp > path-at 1 -1 [if 0 < path-at 1 -1 [if (0 < pher-at 1 -1) or
(0 < p-at 1 -1) [set newh 135 set tp (path-at 1 -1)]]]

if tp > path-at 0 -1 [if 0 < path-at 0 -1 [if (0 < pher-at 0 -1) or
(0 < p-at 0 -1) [set newh 180 set tp (path-at 0 -1)]]]

if tp > path-at -1 -1 [if 0 < path-at -1 -1 [if (0 < pher-at -1 -1)
or (0 < p-at -1 -1) [set newh 225 set tp (path-at -1 -1)]]]

if tp > path-at -1 0 [if 0 < path-at -1 0 [if (0 < pher-at -1 0) or
(0 < p-at -1 0) [set newh 270 set tp (path-at -1 0)]]]

if tp > path-at -1 1 [if 0 < path-at -1 1 [if (0 < pher-at -1 1) or
(0 < p-at -1 1) [set newh 315 set tp (path-at -1 1)]]]

if tp > path-at 0 2
 [if 0 < path-at 0 2
[if (0 < pher-at 0 2) or (0 < p-at 0 2) [set newh 0 set tp (path-at
```

```
0 2)]]]

if tp > path-at 1 2
 [if 0 < path-at 1 2
[if (0 < pher-at 1 2) or (0 < p-at 1 2) [set newh 30 set tp (path-at
1 2)]]]

if tp > path-at 2 2
 [if 0 < path-at 2 2
[if (0 < pher-at 2 2) or (0 < p-at 2 2) [set newh 45 set tp (path-at
2 2)]]]

if tp > path-at 2 1
 [if 0 < path-at 2 1
[if (0 < pher-at 2 1) or (0 < p-at 2 1) [set newh 60 set tp (path-at
2 1)]]]

if tp > path-at 2 0
 [if 0 < path-at 2 0
[if (0 < pher-at 2 0) or (0 < p-at 2 0) [set newh 90 set tp (path-at
2 0)]]]

if tp > path-at 2 -1
 [if 0 < path-at 2 -1
[if (0 < pher-at 2 -1) or (0 < p-at 2 -1) [set newh 120 set tp
(path-at 2 -1)]]]

if tp > path-at 2 -2
 [if 0 < path-at 2 -2
[if (0 < pher-at 2 -2) or (0 < p-at 2 -2) [set newh 135 set tp
(path-at 2 -2)]]]

if tp > path-at 1 -2
 [if 0 < path-at 1 -2
[if (0 < pher-at 1 -2) or (0 < p-at 1 -2) [set newh 150 set tp
(path-at 1 -2)]]]

if tp > path-at 0 -2
```

```
 [if 0 < path-at 0 -2
[if (0 < pher-at 0 -2) or (0 < p-at 0 -2) [set newh 180 set tp
(path-at 0 -2)]]]

if tp > path-at -1 -2
 [if 0 < path-at -1 -2
[if (0 < pher-at -1 -2) or (0 < p-at -1 -2) [set newh 210 set tp
(path-at -1 -2)]]]

if tp > path-at -2 -2
 [if 0 < path-at -2 -2
[if (0 < pher-at -2 -2) or (0 < p-at -2 -2) [set newh 225 set tp
(path-at -2 -2)]]]

if tp > path-at -2 -1
 [if 0 < path-at -2 -1
[if (0 < pher-at -2 -1) or (0 < p-at -2 -2) [set newh 240 set tp
(path-at -2 -1)]]]

if tp > path-at -2 0
 [if 0 < path-at -2 0
[if (0 < pher-at -2 0) or (0 < p-at -2 0) [set newh 270 set tp
(path-at -2 0)]]]

if tp > path-at -2 1
 [if 0 < path-at -2 1
[if (0 < pher-at -2 1) or (0 < p-at -2 1) [set newh 300 set tp
(path-at -2 1)]] ]

if tp > path-at -2 2
 [if 0 < path-at -2 2
[if (0 < pher-at -2 2) or (0 < p-at -2 2) [set newh 315 set tp
(path-at -2 2)]]]

if tp > path-at -1 2
 [if 0 < path-at -1 2
[if (0 < pher-at -1 2) or (0 < p-at -1 2) [set newh 330 set tp
(path-at -1 2)]]]
```

```
seth newh lt random 15 rt random 15 end




;;;;;;;;;;;;;;;;;;;;;;;;;; ;follows chmeical from lowest to
highest;; ;;;;;;;;;;;;;;;;;;;;;;;;;;


to white-food-path set newh heading settp (path-at 0 0)


if tp < chemical-at 0 1 [if 0 < chemical-at 0 1 [set newh 0 set tp
(chemical-at 0 1)]]

if tp < chemical-at 1 1 [if 0 < chemical-at 1 1 [set newh 45 set tp
(chemical-at 1 1)]]

if tp < chemical-at 1 0 [if 0 < chemical-at 1 0 [set newh 90 set tp
(chemical-at 1 0)]]

if tp < chemical-at 1 -1 [if 0 < chemical-at 1 -1 [set newh 135 set
tp (chemical-at 1 -1)]]

if tp < chemical-at 0 -1 [if 0 < chemical-at 0 -1 [set newh 180 set
tp (chemical-at 0 -1)]]

if tp < chemical-at -1 -1 [if 0 < chemical-at -1 -1 [set newh 225
set tp (chemical-at -1 -1)]]

if tp < chemical-at -1 0 [if 0 < chemical-at -1 0 [set newh 270 set
tp (chemical-at -1 0)]]

if tp < chemical-at -1 1 [if 0 < chemical-at -1 1 [set newh 315 set
tp (chemical-at -1 1)]]
```

```
if tp < chemical-at 0 2
 [if 0 < chemical-at 0 2
[set newh 0 set tp (chemical-at 0 2)]]

if tp < chemical-at 1 2
 [if 0 < chemical-at 1 2
[set newh 30 set tp (chemical-at 1 2)]]

if tp < chemical-at 2 2
 [if 0 < chemical-at 2 2
[set newh 45 set tp (chemical-at 2 2)]]

if tp < chemical-at 2 1
 [if 0 < chemical-at 2 1
[set newh 60 set tp (chemical-at 2 1)]]

if tp < chemical-at 2 0
 [if 0 < chemical-at 2 0
[set newh 90 set tp (chemical-at 2 0)]]

if tp < chemical-at 2 -1
 [if 0 < chemical-at 2 -1
[set newh 120 set tp (chemical-at 2 -1)]]

if tp < chemical-at 2 -2
 [if 0 < chemical-at 2 -2
[set newh 135 set tp (chemical-at 2 -2)]]

if tp < chemical-at 1 -2
 [if 0 < chemical-at 1 -2
[set newh 150 set tp (chemical-at 1 -2)]]

if tp < chemical-at 0 -2
 [if 0 < chemical-at 0 -2
[set newh 180 set tp (chemical-at 0 -2)]]

if tp < chemical-at -1 -2
 [if 0 < chemical-at -1 -2
```

```
[set newh 210 set tp (chemical-at -1 -2)]]

if tp < chemical-at -2 -2
 [if 0 < chemical-at -2 -2
[set newh 225 set tp (chemical-at -2 -2)]]

if tp < chemical-at -2 -1
 [if 0 < chemical-at -2 -1
[set newh 240 set tp (chemical-at -2 -1)]]

if tp < chemical-at -2 0
 [if 0 < chemical-at -2 0
[set newh 270 set tp (chemical-at -2 0)]]

if tp < chemical-at -2 1
 [if 0 < chemical-at -2 1
[set newh 300 set tp (chemical-at -2 1)]]

if tp < chemical-at -2 2
 [if 0 < chemical-at -2 2
[set newh 315 set tp (chemical-at -2 2)]]

if tp < chemical-at -1 2
 [if 0 < chemical-at -1 2
[set newh 330 set tp (chemical-at -1 2)]]

seth newh lt random 15 rt random 15 end

;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;follows path from highest to lowest
;only for rebound ;checks only if there is red pheromone
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to iwannagohome set newh heading settp (600)

if tp > path-at 0 1 [if 0 < path-at 0 1 [if (0 < p-at 0 1) [set newh
0 set tp (path-at 0 1)]]]

if tp > path-at 1 1 [if 0 < path-at 1 1 [if (0 < p-at 1 1) [set newh
```

```
45 set tp (path-at 1 1)]]]

if tp > path-at 1 0 [if 0 < path-at 1 0 [if (0 < p-at 1 0) [set newh
90 set tp (path-at 1 0)]]]

if tp > path-at 1 -1 [if 0 < path-at 1 -1 [if (0 < p-at 1 -1) [set
newh 135 set tp (path-at 1 -1)]]]

if tp > path-at 0 -1 [if 0 < path-at 0 -1 [if (0 < p-at 0 -1) [set
newh 180 set tp (path-at 0 -1)]]]

if tp > path-at -1 -1 [if 0 < path-at -1 -1 [if (0 < p-at -1 -1)
[set newh 225 set tp (path-at -1 -1)]]]

if tp > path-at -1 0 [if 0 < path-at -1 0 [if (0 < p-at -1 0) [set
newh 270 set tp (path-at -1 0)]]]

if tp > path-at -1 1 [if 0 < path-at -1 1 [if (0 < p-at -1 1) [set
newh 315 set tp (path-at -1 1)]]]

if tp > path-at 0 2 [if 0 < path-at 0 2 [if (0 < p-at 0 2) [set newh
0 set tp (path-at 0 2)]]]

if tp > path-at 1 2 [if 0 < path-at 1 2 [if (0 < p-at 1 2) [set newh
30 set tp (path-at 1 2)]]]

if tp > path-at 2 2 [if 0 < path-at 2 2 [if (0 < p-at 2 2) [set newh
45 set tp (path-at 2 2)]]]

if tp > path-at 2 1 [if 0 < path-at 2 1 [if (0 < p-at 2 1) [set newh
60 set tp (path-at 2 1)]]]

if tp > path-at 2 0 [if 0 < path-at 2 0 [if (0 < p-at 2 0) [set newh
90 set tp (path-at 2 0)]]]

if tp > path-at 2 -1 [if 0 < path-at 2 -1 [if (0 < p-at 2 -1) [set
newh 120 set tp (path-at 2 -1)]]]
```

```
if tp > path-at 2 -2 [if 0 < path-at 2 -2 [if (0 < p-at 2 -2) [set
newh 135 set tp (path-at 2 -2)]]]

if tp > path-at 1 -2 [if 0 < path-at 1 -2 [if (0 < p-at 1 -2) [set
newh 150 set tp (path-at 1 -2)]]]

if tp > path-at 0 -2 [if 0 < path-at 0 -2 [if (0 < p-at 0 -2) [set
newh 180 set tp (path-at 0 -2)]]]

if tp > path-at -1 -2 [if 0 < path-at -1 -2 [if (0 < p-at -1 -2)
[set newh 210 set tp (path-at -1 -2)]]]

if tp > path-at -2 -2 [if 0 < path-at -2 -2 [if (0 < p-at -2 -2)
[set newh 225 set tp (path-at -2 -2)]]]

if tp > path-at -2 -1 [if 0 < path-at -2 -1 [if (0 < p-at -2 -2)
[set newh 240 set tp (path-at -2 -1)]]]

if tp > path-at -2 0 [if 0 < path-at -2 0 [if (0 < p-at -2 0) [set
newh 270 set tp (path-at -2 0)]]]

if tp > path-at -2 1 [if 0 < path-at -2 1 [if (0 < p-at -2 1) [set
newh 300 set tp (path-at -2 1)]] ]

if tp > path-at -2 2 [if 0 < path-at -2 2 [if (0 < p-at -2 2) [set
newh 315 set tp (path-at -2 2)]]]

if tp > path-at -1 2 [if 0 < path-at -1 2 [if (0 < p-at -1 2) [set
newh 330 set tp (path-at -1 2)]]]

seth newh lt random 15 rt random 15 end


;the order of which the programs are ran'

to go looking-for-food return-to-nest end

;Observer Procedure
```

```
globals [where toggle x y finaltrail] patches-own [food2 p pher
chemical path border?]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;where - used to set heading to a random direction ;toggle- switch
used to create a circle fill up chemical ; x and y - stores the x
and y coordinate of the ant that activated the toggle ; finaltrial -
stores the current trail number of that ant
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;food2 -  equivalent to food but named different for the patches ; p
- pheromone variable secreted when looking for food ;pher- pheromone
variable  secreted when food is found ;chemical- pheromone variable
secreted at the point where the food is ;found ;path - keeps a
gradient that ants follow to go back to the nest ;border? - a
boolean that defines the brown as the border
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


to setup ca          ;clears everthing ask-patches [setpc black]
;makes background black ask-patches [if (xcor = 28 or xcor = -28 or
ycor = 27 or ycor = -27) [setpc brown]] ;colors in borders


;creates food patches and lets there be a density ;deleting any of
these lines will take away a section of food ask-patches [if (pc
not= blue) and (pc not= brown) and (xcor > 6) and (ycor > 6) and
((random 1000) < density) [setpc yellow]] ask-patches [if (pc not=
blue) and (pc not= brown) and (xcor < -10) and (ycor < -10) and
((random 1000) < density) [setpc yellow]] ask-patches [if (pc not=
blue) and (pc not= brown) and (xcor < -13) and (ycor < 20) and (ycor
> -6) and ((random 1000) < density) [setpc yellow]]

;creates nest ask-patches-with [xcor > -2 and xcor < 2 and ycor > -2
and ycor < 2] [setpc blue] ask-patches-with [pc = blue] [setp .01
```

```
setpath .1] ;gives the amount of food per square ask-patches-with
[pc = yellow] [setpher .01 setfood2 10 setchemical 8000]

;num_ant is a slider that allows the control of amount of ants
created intially crt num_ant


;defines border? which is all brown spaces ask-patches [if xcor = 28
or xcor = -28 or ycor = 27 or ycor = -27 [setpc brown]] ask-patches
[setborder? xcor = 28 or xcor = -28 or ycor = 27 or ycor = -27 or pc
= brown]


;inital creation of the ants ask-turtles [setc magenta fd 1
    set deathcount (200 + random 100) ]
ask-turtles [setshape termite] direction end




to dissipate circle          ;creates circle of chemical
ask-patches-with [pc = brown] ;make sure brown does not have any
chemicals on it [setpher 0 setp 0 setchemical 0] ;updates borders
ask-patches-with [pc = brown] [setborder? xcor = 28 or xcor = -28 or
ycor = 27 or ycor = -27 or pc = brown] ;when patch has no kind of
pheromone on it. ;It will make that patch to black and reset it
ask-patches-with [p > 0 or pher > 0 or chemical > 0] [if pher <= 1
and p <= 1 and chemical <= 1 and pc not= yellow and pc not= blue and
pc not= brown
        [setpc black setpath 0]]
if ((average-of-turtles [sum]) mod 2) < 1

    ;it evaporates the pher, p, and chemical pheromone
    [ask-patches
        [ifelse pher < 1
            [if pc not= blue and pc not= yellow
                [set pher 0]]
            [if pc not= blue and pc not= yellow
```

```
            [set pher (pher - 1)]]

        ifelse p < 1 and pc not= blue and pc not= yellow
            [if pc not= blue and pc not= yellow
                [set p 0]]
            [if pc not= blue and pc not= yellow
                [set p (p - 1)]]


        ifelse chemical < 1
            [if pc not= blue and pc not= yellow
                [set chemical 0]]
            [if pc not= blue and pc not= yellow
                [set chemical (chemical - 1)]]]]
end

;sets random direction ot all ants to direction set where (random
360) ask-turtles [if color = magenta [seth where]] end


to circle if toggle = 1 ;creates circle with radius and makes it
stronger in the middle of the circle [ask-patches

    [if (distance x y) < 5
        [ifelse chemical = 0
            [setchemical 30
            if (distance x y) < 3
                [setchemical 40]]
            [setchemical chemical]
        if chemical = 30 and pc not= yellow
             [if pc not= brown and pc not= blue
                [setpc white]]
        if chemical = 40 and pc not= yellow and pc not= blue
            [if pc not= brown
                [setpc pink]]]
    ]
    settoggle 0]        ;resets the toggle
end
```

```
;creates random amount of brown patches ;treefalling is the density
and is also a slider to obstacles
    ask-patches [if ((pc not= blue) and (pc not= yellow) and ((random 1000) < tree
        [setpc brown]]
end
```

Appendix 2

First order perturbation analysis done with Maple.

```
> alias(u=u(x,t));
                                 u
> u_expan:=u0(x,t)+epsilon*u1(x,t)+epsilon^2*u2(x,t);
                                                        2
   u_expan := u0(x, t) + epsilon u1(x, t) + epsilon  u2(x, t)
> #eqn:=diff(u(x,t),t)=s*u(x,t)*(1-u(x,t))+epsilon*diff(u(x,t),x,x);
> eqn:=diff(u,t)=s*u*(1-u)+epsilon*diff(u,x,x);
                                                  /  2   \
                d                                 | d    |
         eqn := --- u = s u (1 - u) + epsilon     |---- u|
                dt                                |   2  |
                                                  \ dx   /
> temp:=expand(subs(u=u_expan, eqn));
        / d          \           / d          \          2 / d          \
temp := |--- u0(x, t)| + epsilon |--- u1(x, t)| + epsilon  |---
u2(x, t)| = s
        \ dt         /           \ dt         /            \ dt         /

                       2
  u0(x, t) - s u0(x, t)  - 2 s u0(x, t) epsilon u1(x, t)

                         2
   - 2 s u0(x, t) epsilon  u2(x, t) + s epsilon u1(x, t)

              2         2              3
   - s epsilon  u1(x, t)  - 2 s epsilon  u1(x, t) u2(x, t)

                                                          /  2          \
              2                     4         2           | d           |
   + s epsilon  u2(x, t) - s epsilon  u2(x, t)  + epsilon |---- u0(x, t)|
                                                          |   2         |
                                                          \ dx          /
```

```
             /  2          \            /  2          \
           2 | d           |          3 | d           |
   + epsilon  |---- u1(x, t)| + epsilon  |---- u2(x, t)|
             |   2         |            |   2         |
             \ dx          /            \ dx          /
> eqn0:=diff(u0(x,t),t)=s*u0(x,t)*(1-u0(x,t));
> IC0:=exp(-N*(x-x0)^2);
                              /           2\
                   IC0 := exp\-N (x - x0) /
> sol0:=pdsolve(eqn0, u0(x,t));
                                      1
            sol0 := u0(x, t) = --------------------
                               1 + exp(-s t) _F1(x)
> Sol0:=simplify(subs(_F1(x)=1/IC0-1, sol0));
                   /   /            2\\//   /            2\
Sol0 := u0(x, t) = \exp\-N (-x + x0) // \exp\-N (-x + x0) / + exp(-s
t)

        /          2                  2\\
   - exp\-s t - N x  + 2 N x x0 - N x0 //
> plot3d(subs(N=1000,s=2,x0=.5,rhs(Sol0)), x=0..1,t=0..20, numpoints=2000);

> #when s is decreasing, the spreading/branching is thinner.
```