Version 1.1 Jun 2017

SWARMATHON 2

ADVANCED BIO-INSPIRED SEARCH

1 FOLLOW THE TRAIL

Have you ever seen a solitary ant find a pile of food, only to be followed by a whole line of ants just a few minutes later? Have you ever wondered how they do that? Stigmergy is the answer.

1.1 WHAT IS STIGMERGY?

Stigmergy is communication that occurs through the environment, rather than from individual to individual. Some ants employ stigmergy by laying a chemical pheromone trail that other ants can follow. Laying pheromone allows an ant to signal to other ants where resources are without direct communication. Ants also reinforce

existing trails if they are still useful. The Swarmies can mimic this behavior too—using spray color and their on-board cameras!

In Swarmathon 2, we will build on some base concepts from Swarmathon 1. In Swarmathon 1, we worked extensively with robots-own variables that represented each robot's memory and state. The robots used their memory to employ the site fidelity strategy. The go procedure controlled the robots' behavior based on their current state.

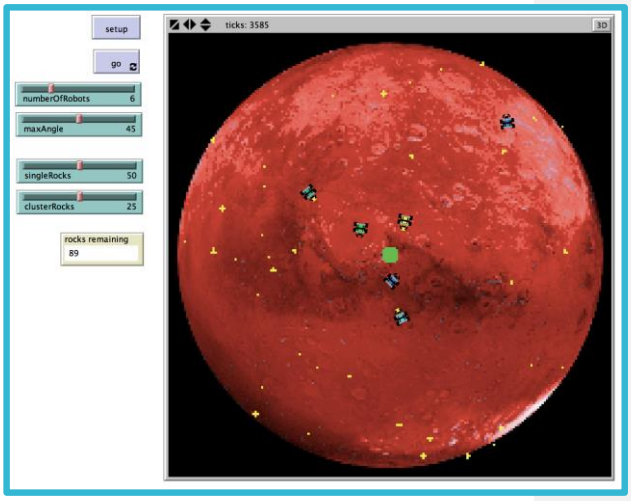
To implement stigmergy, we'll use both robots-own variables and the go procedure again. But we'll also need to work with the "ground" in NetLogo—the patches. To this end, patches-own variables and variables that change their value over time are introduced.

2 GETTING STARTED

2.1 FILE SETUP

As in Swarmathon 1, we will be using NetLogo base code and a background image.

- Create a folder named *yourlastname _Swarmathon2.*
- Place the .nlogo file and the .jpg file in your new folder.
- Open the .nlogo file.



Click the setup and go buttons. The robots should search for rocks and return them to the base.

2.2 REVIEW SWARMATHON 1 CHALLENGES

Click on the Code tab. Note that the **End of Section Challenges** from Swarmathon 1 are also completed. Check your answers!

- On the Interface and in the setup procedure: a slider controls the number of robots. ([Sw1] Section 2)
- In the go procedure: the robots return to the base when no rocks remain. ([Sw1] Section 3)

3 THE TRAIL TO SUCCESS

Shutterstock

3.1 WHAT DO WE NEED TO ADD?

To implement pheromone trail following and laying in the robots, we'll need to code the following behaviors:

MAIN AGENDA

1) Robots need to know if they are using pheromone.
2) Patches need to know if they have pheromone on them and how long it has been there.
3) We should create some larger clusters of rocks to test the pheromone's effectiveness. If our pheromone is working correctly, robots should lay trails from the large cluster and follow trails to the large clusters.
4) Our robot controller statements in the go procedure need to handle the state in which the robots are using pheromone.

Note that an additional state, returning?, has been added for you already.

5) The go procedure should also control how pheromone evaporates on the patches.
6) The look-for-rocks procedure will need to be modified. When a robot finds a rock, it should check if there are other rocks in the immediate area and turn on pheromone if there are. By doing this, it will be able to lay a trail back to the base to signal to other robots that there are rocks at the end of the trail.
7) The return-to-base procedure will need to be modified. While a robot who turned on pheromone is returning to the base, it should lay pheromone. Also, if a robot is at the base after dropping off a rock, it should check to see if there are any trails it can follow.
8) State switching by turning on and off the robots-own variables will be managed throughout the program.

Note that the time limit for a pheromone trail is controlled through a slider on the Interface. This has been implemented for you.

4 MODIFYING GLOBALS & PROPERTIES AND THE SETUP PROCEDURE

Let's begin by tackling agenda items 1 – 3.

AGENDA 1 – 3

1) Robots need to know if they are using pheromone.
2) Patches need to know if they have pheromone on them, and how long it has been there.
3) We should create some larger clusters of rocks to test the pheromone's effectiveness. If our pheromone is working correctly, robots should lay trails from the large clusters and follow trails to the large clusters.

Scroll to the top of the file where the Globals and Properties section is. Add the new pheromone state to the robot and the new pheromone time limit variable to the patches as in the picture here.

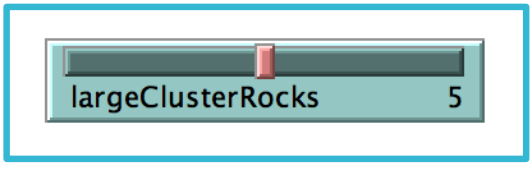
```
;;Each robot knows some information about itself:
 robots-own [
   ;;Is it in the searching? state?
   searching?

   ;;Is it in the returning? state?
   returning?

   ;;1) Is it usingPheromone?
   usingPheromone?
   ]

 ;;Each patch knows some information about itself:
 patches-own [

   ;;What color they start as.
   baseColor

   ;;2) How much time they have left of pheromone before it evaporates.
   pheromoneCounter

   ]
```

Now scroll to the setup procedure. Set the initial values of the variables we just created. Note that you can set robots-own variables at the same time as you are creating robots.

```
;; Create a number of robots based on the slider bar.
;; Set their properties and robots-own variable values.
create-robots numberOfRobots [
  set shape "robot"
  set size 8
  set searching? true
  set returning? false

  ;;1) Robots start off not using pheromone.
  set usingPheromone? false
]

;; Patches remember their starting color
ask patches [
  set baseColor pcolor

  ;;2) There's no pheromone on the patches yet, so
  ;;   set the counter to 0.
  set pheromoneCounter 0
  ]
```

To complete section 4, we'll add an option to create large clusters of rocks. A slider has already been added on the Interface to control the number of these clusters.

Commented [SS1]: What is section 3.2?



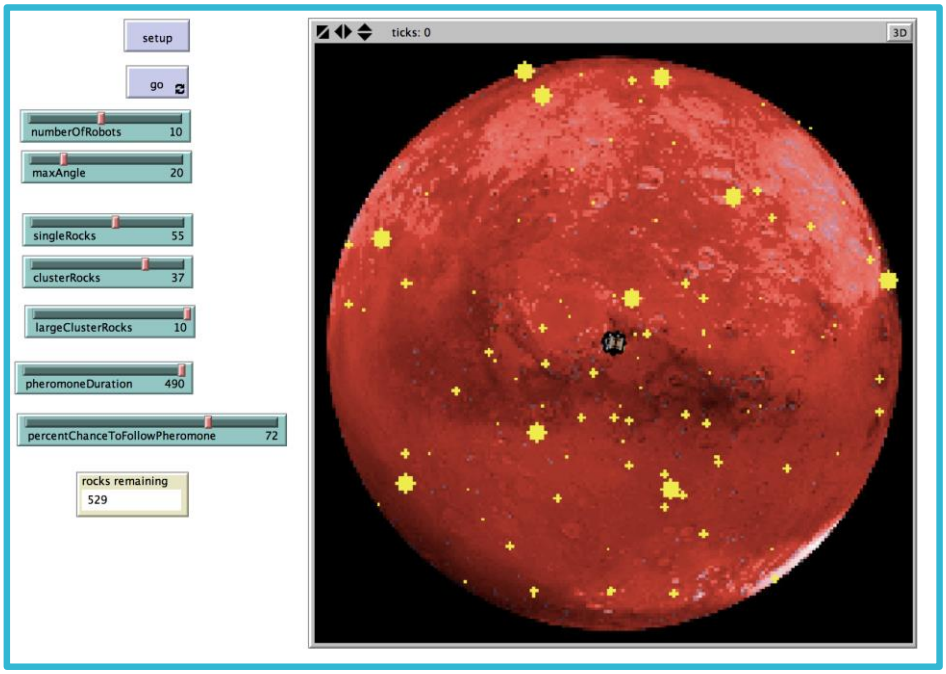
We'll use a strategy similar to how we set up the base with a green color. First, we'll choose a random patch. Then, we'll make sure that the chosen patch as well as all other patches within a radius of 3:

- are not off-world (patch color is black)
- are not already rocks (patch color is yellow)

If all of these conditions are met, we'll place a large cluster there. Each cluster has 29 rocks. Enter the code as in the picture below. Don't forget to add the new rocks to the numberOfRocks!

```
;;3) Create some larger clusters of 29 rocks.
let targetLargeClusters largeClusterRocks
while [targetLargeClusters > 0][
  ask one-of patches[
    if pcolor != black and pcolor != yellow and [pcolor] of patches in-radius 3 != black
    and [pcolor] of patches in-radius 3 != yellow[
      set pcolor yellow
      ask patches in-radius 3 [set pcolor yellow]
      set targetLargeClusters targetLargeClusters - 1
    ]
  ]
  ]
set numberOfRocks numberOfRocks + (largeClusterRocks * 29)
```

Try out the new cluster style by navigating back to the Interface. Adjust the largeClusterRocks slider value to be greater than 0 if necessary. Note that a setup button has already been created for you. Click it to see your new rock configuration!



5 MODIFYING THE GO PROCEDURE

In this section, we will complete Agenda items 4 and 5.

AGENDA 4 – 5

4) Our robot controller statements in the go procedure need to handle the state in which the robots are using pheromone. Note that an additional state, returning?, has been added for you already.
5) The go procedure should also control how pheromone evaporates on the patches.

The robots are already programmed to look for rocks and return them to the base. Note that a robot only uses the new procedure check-for-trails when it is using pheromone but not returning to base. This distinction is important because a robot does not just follow trails, it lays them. Thus, a robot who is returning? and usingPheromone? is doing the latter and should not check-for-trails to follow.

Add the controller for the robots' pheromone state by carefully reading the comments in the base code. Use the picture below to guide you.

```
;;These statements control the main behavior of the robots.
;; 1) There are two cases where a robot is using pheromone:
;;    - It found a sufficient density of rocks and is laying a trail back to the base
;;      for other robots to follow.
;;    - It picked up a trail at the base and is currently following it.
;;    The first case is handled by return-to-base.
;;    We'll need to take care of the second here:
;;    If a robot is using pheromone but is not currently returning to the base, it
;;    must be following a trail.
if usingPheromone? and not returning? [check-for-trails]
if searching? [look-for-rocks]
if returning? [return-to-base]
```

We can also write a controller for the patches. Let's do that so that every time the procedure runs (once per tick), the pheromone trails evaporate. As a trail evaporates, it changes color. Using patch color is a great way to distinguish between different stages of evaporation, as it is easy to see when running the simulation.

We don't want robots to sense the faintest trail, but we'll implement that behavior a little bit later on. For now, fill in the patches

controller by carefully reading the comments and using the following picture.

```
    ;;2) Manage the pheromone on the patches.
    ask patches
    [
      ;; Handle the case where the pheromoneCounter is down to 1 separately
      ;; so that we don't go into negatives.
      if pheromoneCounter = 1[
        set pheromoneCounter 0
        set pcolor baseColor
      ]
      ;;Colors denote trail strength.
      if pheromoneCounter > 1 and pheromoneCounter <= 50 [set pcolor cyan - 13]
      if pheromoneCounter > 50 and pheromoneCounter <= 100 [set pcolor cyan - 10]
      if pheromoneCounter > 0 [set pheromoneCounter pheromoneCounter - 1]
    ]
  ]
```

There will be no visible changes at this point if you click setup and go. This is because we've haven't set the usingPheromone? to true yet! Next, we'll code the conditions under which a robot turns on pheromone as well as when it follows it.

6 FINAL STEPS

In this section, we will complete our Agenda!

AGENDA 6 – 8

6) The look-for-rocks procedure will need to be modified. When a robot finds a rock, it should check if there are other rocks in the immediate area and turn on pheromone if there are. By doing this, it will be able to lay a trail back to the base to signal to other robots that there are rocks at the end of the trail.
7) The return-to-base procedure will need to be modified. While a robot who turned on pheromone is returning to the base, it should lay pheromone. Also, if a robot is at the base after dropping off a rock, it should check to see if there are any trails it can follow.
8) State switching is managed.

Robots shouldn't lay a pheromone trail every time they find a rock. If they had just found a single rock, then laying a trail wouldn't help other robots at all. Let's write the condition so that 2 additional rocks must be in the area for the robot to turn on pheromone.

6.1 LAYING THE TRAIL

Doing this is surprisingly easy! Just scroll to the look-for-rocks procedure and enter this if statement:

```
;; 1) Now count the yellow patches (rocks) around the robot.
;; If that number is greater than or equal to 2, the robot
;; asks itself to set usingPheromone? to true.
if count patches in-radius 1 with [pcolor = yellow] >= 2[
  ask myself [set usingPheromone? true]
]
```

Note that in the look-for-rocks procedure, **all** robots who find a rock set their state so that they will return-to-base, regardless of whether they are usingPheromone? or not.

Now that we've set the **condition** under which the robots will lay trails back to the nest, let's code that **behavior**. Scroll to the next procedure, return-to-base. Note that there are two areas to fill in, 1) and 2), and that the first one is at the *bottom* of the procedure.

Let's start with 1). We will have robots that found 2 or more additional rocks in the area where they picked up a rock lay a trail back to the base. Let's make the trail by changing the patch color under the robot to cyan. Recall that in our pheromone controller in the go procedure, we specified that cyan was the strongest (most recently laid) trail.

IMPORTANT! Do not change yellow patches to cyan—or you are eliminating rocks!

Enter the code from the picture below to complete 1).

```
;;  1) Lay a pheromone trail back to the base if we are using pheromone.
;;  Other robots can pick it up.
;;  Be careful not to knock out rocks with the trail!
;;  Have the patch set its counter for how long the pheromone lasts.
if usingPheromone? [
  ask patch-here[
    if pcolor != yellow [
    set pcolor cyan
    set pheromoneCounter pheromoneDuration
    ]
  ]
  ]
```

Go back to the interface and click setup and then go. Robots begin to lay trails!

6.2 SENSING AND DECIDING TO FOLLOW THE TRAIL

Now that the robots are laying trails, let's get other robots to follow them. In nature, however, ants don't follow trails 100% of the time. They may decide to do something else. Let's give our robots the same independence by using **probability**.

PROBABILITY PRIMER

If you're not familiar with probability, think of it as a chance you'll do something. Let's say that you have a 0.5 (50%) probability to have pizza for lunch on any given day. I would expect, then, that when I came into the lunchroom, I'd see you eating pizza for lunch about half the time.

Scott Semegran

The percentChanceToFollowPheromone slider will allow us to control the robots' **probability** of picking up an existing trail from the base.

Scroll up to 2). Use what you've just learned about probability to understand the code in the picture below. Enter the code into your program.

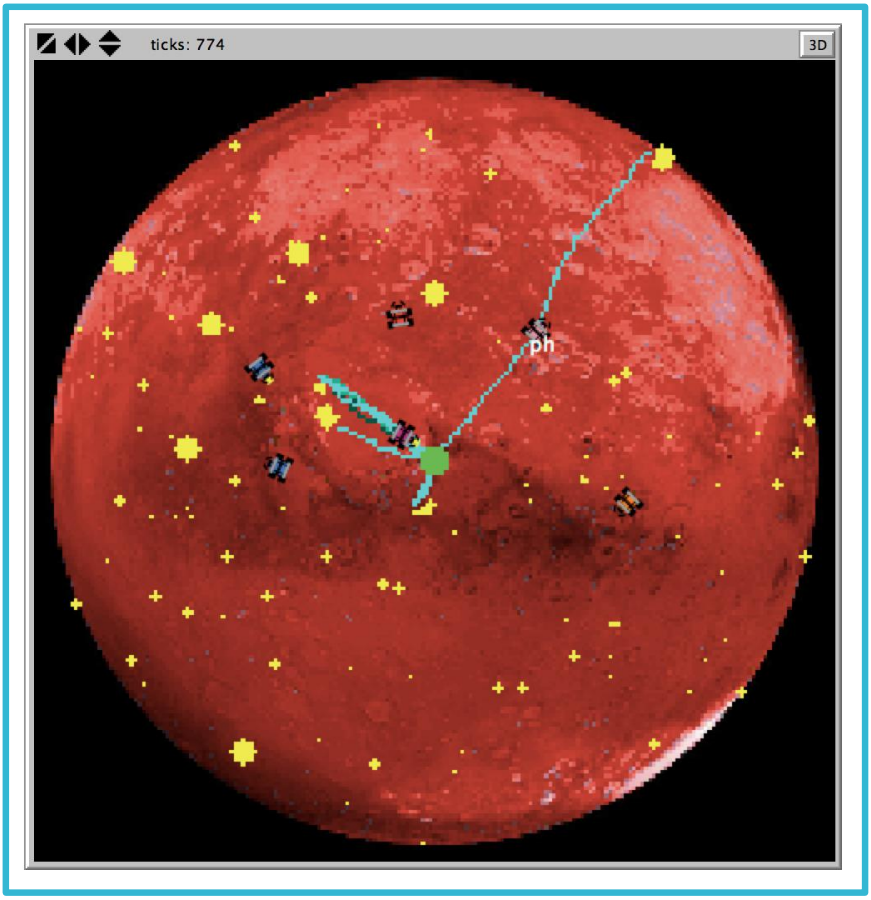
```
;; 2) Set pheromone detection on with probability equal to the slider value.
;; If detection is activated, turn on pheromone, turn off searching, and
;; check-for-trails.
 set usingPheromone? false
 if random 100 < percentChanceToFollowPheromone[
   set usingPheromone? true
   set searching? false
   check-for-trails
 ]
```

Now we've implemented the behavior that a robot will use the function check-for-trails with a probability determined by our slider value. check-for-trails also has a sub-procedure, switch-to-search-from-pheromone. Let's fill in sections 1) — 6) in check-for-trails and switch-to-search-from-pheromone to complete Swarmathon 2!

NOTE: The following code is advanced. It's okay if you don't understand everything. Read comments carefully. Notice that you can use with and of to string code together.

```
;-------------------------------------------------------------------------------
;;;;;;;;;;;;;;;;;;;;;;
;; check-for-trails ;;
;;;;;;;;;;;;;;;;;;;;;;
to check-for-trails

  ;; 1) Use an ifelse statement.
  ;; Sense if there are any? trails near the robot (in-radius 5).
  ;; Robots cannot sense the faintest trails, so only check for the strongest and slightly evaporated trails:
  ;; these trails have color cyan or cyan - 10.
  ifelse any? patches in-radius 5 with [(pcolor = cyan) or (pcolor = cyan - 10)]
  [
    ;; 2) If there is at least one patch that has a trail on it, create a variable called target to hold the one that's farthest
    ;; from the origin.
    let target one-of patches in-radius 5 with [(pcolor = cyan) or (pcolor = cyan - 10)] with-max [distancexy 0 0]

    ;; 3) Use a nested ifelse statement.
    ;; Compare the distance from the origin of the target patch to that of the robot.
    ;; If the patch is farther away than the origin than the robot, then set the robot's
    ;; label to "ph" to indicate that it's using pheromone.
    ifelse [distancexy 0 0] of target > [distancexy 0 0] of self [
      set label "ph"
      face target
    ]
    ;; 4) Else, the trail must be evaporating, or it is behind us.
    ;; Call the sub procedure to take us back to search mode.
    [ switch-to-search-from-pheromone]
    ]
  ;; 5) There aren't any trails near us.
  ;; Call the sub-procedure to take us back to search mode.
  [ switch-to-search-from-pheromone ]

end

;; 6) Fill in the sub-procedure.
;-------------------------------------------------------------------------------
to switch-to-search-from-pheromone

  ;; Turn off usingPheromone?
  set usingPheromone? false

  ;; Turn on searching?
  set searching? true

  ;; Set the label back to empty.
  set label ""

end
```

Return to the Interface and click setup and go. The robots should both lay down and follow pheromone. Additionally, robots that are currently following a trail should show the label "ph."



Try experimenting with the slider values and consider the following:

- How does changing the initial rock setup affect how the robots use pheromone? Do certain rocks cause them to use pheromone more or less? Why do you think that is?

- How does changing the probability affect how the robots use pheromone? Is a lower or higher probability more effective? Why do you think that is?
- How does changing the robots' max walk angle change their search patterns?

GREAT JOB! You completed SWARMATHON 2.

BUG REPORT? FEATURE REQUEST?

Email sherbet@unm.edu with the subject SW2 Report

NEXT UP
SWARMATHON 3: Intro to Deterministic Search