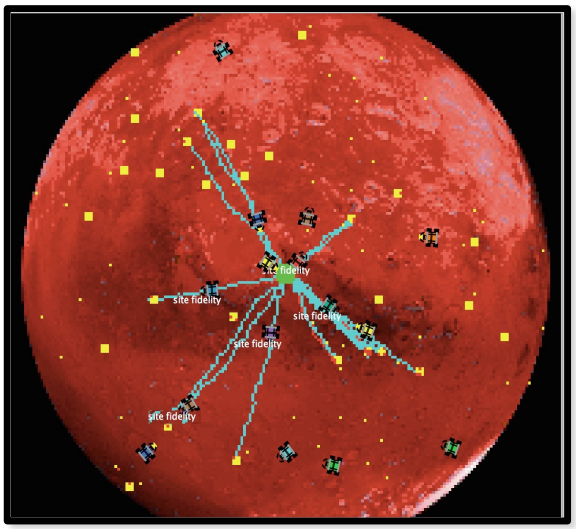
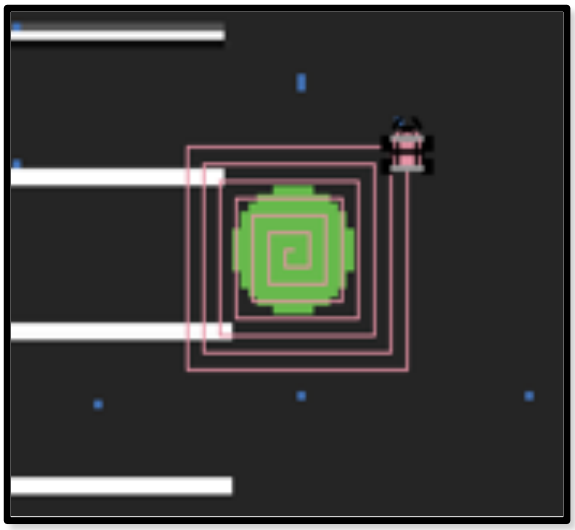
Swarmathon Module 5: Final Project

Introduction:

For this final project, you will build your own search algorithm for the robots by combining techniques introduced in Modules 1—4. You are encouraged to add your own ideas as well.





Project Overview:

We learned several ways in which a robot can effectively search for resources.

1. *Direct Recruitment (Collecting Rocks on Mars 2):*
   - **robots communicate directly with other robots**
2. *Pheromone (Collecting Rocks on Mars 3):*
   - **robots communicate through the environment by using pheromone trails** (*stigmergy*)
3. *Pattern Search (Swarmathon 1)*
   - **robots do not communicate, but move in a pre-determined way**

THE UNIVERSITY of NEW MEXICO

Advantages and Disadvantages

Think about situations where each strategy performs well, and situations where the strategy does not perform well. Consider the distribution of resources, resource distance from the base, etc. When is it better to *not* have robots recruiting?

If you get stuck, try running each program back-to-back. Look at the behavior of the robots. **There are no right or wrong answers here.** Think of this exercise as an idea generation tool to help you design an effective algorithm that combines different search strategies.

Recruitment	Pheromone	Pattern Search
Advantages	**Advantages**	**Advantages**
Disadvantages	**Disadvantages**	**Disadvantages**

THE UNIVERSITY of NEW MEXICO

Put it Together

Recommendations

1. if statements are key. When you want a robot to change behavior, say from moving in a spiral to randomly searching, an if statement can describe when that should happen. Try using different combinations for your if statement. (Number of robots in-radius…, number of resources remaining, etc. The possibilities are endless.)

2. Don't do everything at once. It's hard to tell which strategy combination is working the best if you are putting together multiple strategies at the start. Pick one base strategy that you like (such as pheromone), then try combining behaviors with it (robots switch to spiral search if…, etc.). If you're not pleased with the results, try a different base strategy, or come up with one of your own.

3. It can seem really difficult to come up with your own behaviors for searching. **How do *you* search?** How can you use your human experience to come up with ways a robot might search?

A THOUGHT EXPERIMENT

Imagine that you have lost your keys. How do you look for them?
Do you go immediately to spots where you usually put your keys?
do you search every inch of the area in front of you first?
What if you don't find them? Then what do you do?

THE UNIVERSITY of NEW MEXICO

The Rules

Getting Started

1. You will need to create your own .nlogo file, but you may use the .nlogo file provided with Module 4 as a starting point.
2. You will need to include the provided .nls file and image, or your program will not work. Look at Module 4's walkthrough for help if you don't remember how to do this.
3. You must set up your tag distribution chooser on the Interface tab just like you did in Module 4 (walkthrough #13).
4. You must use 6 robots.
5. You must have a setup procedure and a go procedure. Your setup procedure must setup the robots and world. Your go procedure must contain the robots behavior and a call to tick.
6. Your program must run until all resources are collected.

World & Interface

7. You may not change the shape or size of the world (wrapping, patch size, etc.) from the default settings provided as part of the competition.
8. You may not ask robots or patches to hatch or sprout.
9. A resource is removed by changing the color of a patch. This can only happen when a robot interacts with that patch directly. (You can't ask all patches to turn a different color to get rid of resources)
10. We will specify the distribution of rocks and number of robots. You won't know ahead of time, so your algorithm must be flexible enough to work in a variety of situations.
11. The maximum number of ticks allowed is 36,000 ticks, or 10 ticks per second for 3,600 simulated seconds, which is 60 simulated minutes or 1 hour of simulated time.

Robot Sensing

12. Robots can only sense resources in their local neighborhood, which has a radius of 1 patch (patches directly around the robot):
    12.1. EXAMPLES:
        12.1.1. In-radius-1 is ok! In-radius-2, 3, ..., etc, IS NOT
        12.1.2. Patch-ahead-1 is ok! Patch-ahead-2, 3, ..., etc, IS NOT
        12.1.3. Neighbors and neighbors4 are ok

Robot Movement

13. Robots can only take one step of length 1 per tick. Steps cannot be longer than length 1 (no speeding up robots).
    13.1. Robots may also remain stationary.
14. You may not use setxy, move-to, etc. (no teleporting)

Robot Knowledge

15. You may create and use member variables for robots.
16. Robots may not store lists of locations of resources.

THE UNIVERSITY of NEW MEXICO

Patch Knowledge

17. You may create and use member variables for patches.

Programming

18. You may not use recursive algorithms.
19. You may not remove the tick from the end of the go procedure. You may not add additional calls to tick in other procedures called by go.
20. You may create multiple procedures, but they all must be called within the go procedure.
21. You may experiment with any NetLogo commands not explicitly outlawed here: if in doubt, ask! Be creative.
22. You must use the supplied .nls files. We will run your code using the .nls files provided, so if you make any changes to the .nls, those changes will not be used in the competition.

Competition Rounds

There is a preliminary round and a semi-final round. In the preliminary round, we will run your code twice, using 2 different distributions. You will not know which distributions these are in advance. Your score for the preliminary round is the total rocks you collect in both runs. The top 4 teams will advance to the semi-final round. In the semi-final round you will get a new tag distribution. Your score will be the number of rocks collected on this new distribution only.

Sliders

Before the competition, set the slider values to the values you want to use for the competition. Be sure to save your file after setting these values. Additionally, create a Note in the dropdown menu where you record these values. You may only specify one value per slider for the entire competition.

You are not allowed to change your code, slider values, or any other interface features for the different distributions in different rounds. Thus, it is important that your algorithm is general enough to collect lots of rocks in all of the distributions.

Good luck and happy swarming!