The Computational Beauty of Nature – Spring 2020

Lab 11: Self-Organization

Download and unzip **lab11-self-organization.zip** from the class web page under Labs. This folder contains several NetLogo models that explore the idea of self-organization and emergent behavior, which we discussed in class last week. Spend some time experimenting with each of these models.

Ant Colony

The **AntColony.nlogo** model simulates the behavior of a colony of ants foraging for food. Ants wander around randomly until they find some food, at which point they pick it up and make their way back to the nest, dropping a pheromone chemical as they go. Other ants detect the pheromone trail and follow it. The rate at which an ant deposits pheromone is controlled by the *diffusion-rate* parameter. The pheromone also tends to naturally "evaporate" on its own, with the rate controlled by the *evaporation-rate* parameter. The *population* parameter controls the number of ants in the simulation.

With the right balance of parameter settings, the colony as a whole can devour the three food sources very efficiently (and often in sequence). But with too little diffusion, or too much evaporation, the trails to the food sources become unstable. How does population size affect the ability of the colony to consume the food? Try different parameter combinations to see what produces the most effective emergent behavior, including the following:

- Set *diffusion-rate* to its minimum, and *evaporation-rate* to its maximum.
- Set *diffusion-rate* to its maximum, and *evaporation-rate* to its minimum.
- Try making *diffusion-rate* and *evaporation-rate* equal.
- Run the simulation on high speed to get a better sense of the emergent colony-level behavior.

You can also track the behavior of a single ant using the **AntColonyPerspective.nlogo** model. This is just like the AntColony model, except when you click the *watch one-of turtles* button, it will highlight a particular "turtle" (i.e., ant) so you can see its individual behavior more clearly. The buttons labeled "pd" and "pu" (standing for "pen down" and "pen up") can be used to show the paths followed by a single ant, or all of the ants, over time.

Virtual Ants

The **VirtualAnts.nlogo** and **VirtualAntsLarge.nlogo** models simulate Chris Langton's time-reversible "vants". After seemingly random, chaotic behavior, a vant will eventually begin building an "ant highway". The rules followed by a vant are described under the model's *Info* tab. Try starting with just one vant, and watch what it does over time. Then try 2 vants together. You may notice that at some point one vant will start "undoing" structures that the other one has built. What happens with more than 2 vants?

Termites

The **Termites.nlogo** model simulates a population of "termites" moving wood chips around. Each termite follows a very simple rule: It wanders around randomly, and if it bumps into a wood chip, it picks up the chip and continues to wander randomly; when it bumps into another wood chip, it drops the wood chip it is carrying in a nearby empty space. Start by setting the number of termites to 1 and running the simulation on slow speed. Then speed it up. Then add more termites. Over time, a well-defined pattern of wood-chip distribution may emerge. Can you find an optimal combination of number of termites and wood-chip density for reliably producing large circular piles of wood chips within a reasonable amount of time? You can also track the behavior of individual termites with the **TermitesPerspective.nlogo** model.

Boids

The **Boids.nlogo** model is a 2-D version of Craig Reynolds' famous "Boids" model of bird-flocking behavior. Each boid follows three simple rules:

- *Alignment:* a boid tries to match the direction of movement of its neighboring boids
- *Separation:* a boid tries to avoid getting too close to its neighboring boids
- *Cohesion:* a boid tries to move toward the average center point of its neighboring boids (without getting too close to them)

In the model, you can control the size of a boid's "neighborhood" with the *vision* parameter. The larger the value, the further the distance the boid can see in all directions around it. The *minimum-separation* parameter controls how close a boid can "comfortably" get to its neighbors. The three *max-turn* sliders control how sharply a boid can turn on one time step in response to the three rules above.

Under what combinations of parameter settings does reliable flocking behavior emerge? Are there some settings that prevent such behavior from ever forming? Once a flock forms, does it persist indefinitely?

Fireflies

The **Fireflies.nlogo** model explores the emergence of synchronization among "fireflies". Each firefly is controlled by its own separate internal clock, which cycles at a periodic rate. Fireflies always flash at the beginning of their clock cycle (and the length of the cycle is the same for all fireflies), but each firefly starts out at a different random point in their cycle. Over time, a firefly's cycle is influenced by the flashes of neighboring fireflies around it, which cause the cycle point to shift slightly forward or backward in its timing.

After a while, global synchronization spontaneously emerges. With the default model settings, the effect usually becomes noticeable after about 600-700 time steps. How does changing the number of fireflies or the cycle length affect the ability of the fireflies to synchronize? What about the other parameter settings (more details about those are available under the model's *Info* tab). How reliable in general is the ability of the fireflies to synchronize? Are there just a few parameter configurations that allow "sync" to occur, or does sync seem to be pretty much inevitable?