Bubbler-Project

Furkan Toprak

January 24, 2020

Introduction

My project revolves around building neural networks that emulate the decision-making skills of Sand Bubbler Crabs. Sand Bubbler Crabs are crabs that are located in the coasts of Southeast Asia, and are known for the beautiful patterns they make in the sand. Sand Bubbler Crabs create these patterns when they search the sand for nutrients, by balling up sand, sucking the nutrients out of the sand-ball, and continuing the process.

Figure 1: An example of the patterns the Sand Bubbler Crab makes in the sand

Fitness: General Overview

To represent their trajectory, I have simulated "crabs" on a two-dimensional plane using neural networks. Each crab has a 1000-step trajectory. The crab calculates a unit vector at every step that represents the next step to take. Dependening on the specific trajectory the crab takes, the crab is assigned a fitness using the fitness function F. By the end of the trajectory, we would like to maximize F. The fitness function is as follows:

$$\frac{alive}{1+e^{-a*(E-b)}}$$

where $a = 8, b = 0.5$, E represents efficiency (the number of unique locations visited in the trajectory divided by the total number of steps taken), and *alive* is a boolean (1 or 0) that represents if the crab is alive (see below).

Fitness: Predation

Finding the probability of being alive is a bit more complicated, and requires us to simulate a *Predatory Event*. These crabs are subject to predatory pressure from seabirds. At any given step, there is a $\frac{1}{1000}$ probability that there will a predatory event (an expected 1 predatory event per simulation). If a crab survives all predatory events by the end of its trajectory, the crab survives ($alive = 1$); otherwise, it is dead ($alive = 0$).

The further the crab is from the burrow, the more likely the crab is to die. Let us introduce a constant called the *threshold radius*, r_0. If the crab's distance from the burrow (r) is greater than r_0, predatory pressure is high. Otherwise, predatory pressure is low. We'll discuss the importance of this later.

The process is as follows:
1) Draw a line from the crab's current position to the burrow. The length of this line is r.
2) Count the number of times the line overlaps with the trajectory and divide that number with the distance from the burrow (r). This number is henceforth referred to as v.

2.a) v_{inside} symbolizes the proportion of previously visited positions the crab must cross within the threshold radius.

2.b) $v_{outside}$ symbolizes the proportion of previously visited positions the crab must cross outside of the threshold radius.

2.c) u_{inside}, the proportion of unvisited (free) positions the crab must cross inside of the threshold radius is $1 - v$.

2.d) $u_{outside}$, the proportion of unvisited (free) positions the crab must cross outside of the threshold radius is $1 - v$.

Below is the probability function for staying alive upon a predatory event:
$P_{alive} = (1 - P_{far})^{u_{outside}+2*v_{outside}}(1 - P_{close})^{u_{inside}+2*v_{inside}}$

Explanation
Let us discuss the probability function in detail:

- P_{far} is the probability of the crab dying at any given step outside of the threshold radius ($r > r_0$) when attempting to return to the burrow.
- P_{close} is the probability of the crab dying at any given step inside of the threshold radius ($r \leq r_0$) when attempting to return to the burrow.
- $(1 - P_{far/close})$ represents the probability of survival at any given step.
- $u_{inside} + 2 * v_{inside}$ represents the "time" it takes to get from the crab's position from inside of the critical radius to the burrow.
- $u_{outside} + 2 * v_{outside}$ represents the "time" it takes to get from the crab's position from outside of the critical radius to the critical radius.
- The reason for the coefficient to $v_{inside/outside}$ is because it takes the crab twice as long to cross space it's previously visited.

Maximizing F
With there being a probabilistic aspect to the fitness function, it is of course impossible to find a deterministic solution; instead, I want to find all of trajectories that result in a maximal *expected* fitness. The question is, how can we compute the best trajectory? Brute-forcing the issue could take 1000! possibilities to compute. Well, we know that the possibility space that we neeed to consider collapses as we use a greedy algorithm. However, a greedy algorithm must be revised, because we need to find the best overall trajectory rather than the best next step at every step. I just don't know exactly *how* it collapses. Also, I'm unsure how to theoretically find this ideal trajectory without actually computing it, as the fitness function changes throughout the trajectory (as E is variate). This is where I need your help.

If we could find the most ideal trajectory, I could then compare it to how close my neural networks are to finding the most ideal solution or being stuck in a local optima. This could allow me to have a benchmark/better way of tracking neuroevolution in neural networks of different sizes.