Using Genetic Algorithms to
Find Stable Solutions to the Richardson Equations
Bioinspired Computing, CSE 598
Assignment 1, Spring Semester 2020
Due: Feb. 17, 2020 (3:00 pm)

1 Introduction

In this assignment we will use genetic algorithms to solve find interesting parameter sets for a family of equations known as the Richardson Equations. The Richardson model is a classic model of two-party arms races. In this assignment, we will consider a three-party version of the model, where the parties (countries) are designated as X, Y, and Z. The two weakest countries are always allied against the strongest, where current spending on arms is a proxy for strength. We assume that X is the dominant country, ie. the country that spends the most on arms. One interesting behavior (instability) in the model is a shift in alliance, e.g., when one of the weaker countries becomes the strongest, leading to new alliances. A second behavior (instability) is a runaway arms race, where countries continually increase spending on arms. The model is expressed in the following set of difference equations:

$$\begin{aligned} x_{t+1} &= x_t + (k_{11}(x_s - x_t) + k_{23}(y_t + z_t))(x_m - x_t) \\ y_{t+1} &= y_t + (k_{22}(y_s - y_t) + k_{13}(x_t - z_t))(y_m - y_t) \\ z_{t+1} &= z_t + (k_{33}(z_s - z_t) + k_{12}(x_t - y_t))(z_m - z_t) \end{aligned} \tag{1}$$

- x_s, y_s, and z_s specify the arms expenditures for each country.
- $(y_t + z_t)$, $(x_t - z_t)$, and $(x_t - y_t)$ denote the external threat from adversaries for X, Y, and Z respectively. That is, the threat to X is the sum of Y's and Z's spending on arms $(y_t + z_t)$, because Y and Z are allied against X. For countries that are allied, their threat is reduced by the amount that their ally spends on arms. **If Y or Z becomes dominant, these terms must be modified to reflect the correct external threat. i.e. the variables must be updated to reflect the switch.**
- x_m, y_m, and z_m represent the economic constraints on X, Y, and Z respectively, i.e., what fraction of their resources are available to spend on weapons. We will assume that all countries have equal total wealth, so these parameters specify the relative resources that are available on every step of the simulation.
- Rate constants: k_{11}, k_{22}, k_{33}, k_{12}, k_{23}, and k_{13} control a country's responsiveness to changes in external threat.

Iterating the equations for many steps allows us to observe the steady-state behavior of the model under different assumptions about initial conditions, parameter settings, and noise. Some scenarios will lead to unstable behavior in the forms of unlimited arms races, while others may lead to stable fixed points (arms control)

2 Assignment

There are three aspects to this assignment: Implementing the model; implementing a genetic algorithm to find parameters of the model that lead to desired behaviors; analyzing the model.

2.1 The Three-agent Richardson Model

The model is formulated as a set of difference equations, which you can simulate by supplying initial values and then iterating for a fixed number of steps to observe steady-state behavior. This is what it means to implement the model. Because the model is nonlinear, the equations may not stabilize at a single fixed point. You will need to devise a method to observe the behavior of the model. You will also need to experiment to decide how many steps you need to iterate the model to leave the initial transient behavior and reach steady state, but a small number, say 50 should be sufficient.

Once you have implemented the model, run it a few times with different sets of parameters to get a feel for its behavior.

2.2 Genetic Algorithms

A genetic algorithm will be used to find parameter settings that lead to interesting behavior. We will focus first on parameter settings that lead to a stable balance of power. Iterate your model for n time steps and then use the following fitness function to approximate stability:

$$F = |x_n - (y_n + z_n)| \tag{2}$$

For your first set of experiments, set up the GA to run with 12 parameters, e.g., k_{11}, k_{22}, k_{33}, k_{12}, k_{23}, k_{13}, x_s, y_s, z_s ,x_m, y_m, and z_m. Each of these parameters is real-valued and restricted to the interval $[0, 1]$, as are the three initial conditions: x_0, y_0, and z_0. We will first assume that the initial conditions are each fixed to be 0.0.

You will need to decide how to represent the parameters, whether as discrete bins (using the representation discussed in class), or as real-numbers constrained to the unit interval. If you choose the latter representation, you will need to design a mutation operator that works well on real values.

You are free to write your own genetic algorithms or to use a public domain package. However, if you choose to use a pre-existing package you are responsible for understanding how it works, getting it to work with the model, and analyzing results.

1. Once you have your GA code running, test it on some easy functions (like MaxOnes) to make sure it is operating the way you expect.

2. Next, run your GA on the Richardson model with the objective of finding a stable (non-degenerate) solution.

3. Now try running your GA with larger/smaller population sizes, and with different mutation and crossover rates, to find GA parameters that produce the best performance.

2.3 Analyzing the Model

Your objective is to find parameter sets that lead to stable behavior, meaning that there is not a shift in alliance over time and no run away arms race. You are likely to find *partially degenerate*

solutions, which means that some of the parameters are close to either 0.0 or 1.0. These are much less interesting than solutions that are not degenerate. Similarly, your GA search assumes one fixed set of initial conditions, which was chosen arbitrarily.

1. Run your GA and see if it can find three or four non-degenerate stable solutions, across multiple runs.

2. Find at least one example (ideally, non-degenerate) of a solution that (a) is a run-away arms race; and (b) has an alliance shift.

3. For each stable set of parameters you find in (1), experiment with different initial conditions to see how the behavior changes when the model is run with different initial values,. Specifically, run the model with different initial conditions, varying their value between 0.0 and 1.0. You can accomplish this by systematically varying each of the three initial values and recording how and if the behavior changers. Are some solutions (parameter settings for the model) sensitive to particular initial conditions (meaning that a slight change in input values produces a significantly different steady-state behavior)? How fine-grained does your step size (of initial conditions) need to be?

4. Next try encoding the initial conditions on the genome (adding three more parameters to your representation) and rerunning the GA, so that the initial conditions are evolved together with the model parameter settings. How do your results change?

Extra Credit

1. If you find a solution (set of parameters that is stable with inputs 0, 0, 0, but not stable with other inputs), make a plot of its *basin of attraction*. There are several ways to do this, but a simple way is make two-dimensional slices through the three-dimensional space. That is, you can choose two (of the three) input variables, assign one color to stable and a second color to unstable, then make a 2-d plot indicating at each point whether it is stable or unstable. Three such plots will reveal all of the variable combinations.

2. Now that you have explored varying the initial conditions, we would like to know how robust the model is to the model parameters. Since there are 12 parameters, it is infeasible to vary all of them in combination (that is why we used a GA to search for good parameter settings). However, you can choose 1 or 2 parameters at a time and perturb them varying amounts, e.g., by adding or subtracting σ= 0.0, 0.001, 0.003, and 0.005 to the parameter and seeing if it changes the behavior significantly. One interesting plot is to choose two interesting parameters, say k_{11} and z_s, vary them systematically, and then plot which country is strongest for each parameter setting. But, you will need to experiment to find which parameters are most sensitive and produce interesting changes in the model behavior. Test your model with one set of initial conditions (0,0,0), and make plots of the

3 Reporting results

Please hand in a 4-5-page (approx.) report using the ACM format, written in appropriate academic style with proper citations (if you use citations). Please print a copy of your assignment paper and

bring it to class on Feb. 17 and submit a .pdf of the writeup and a .zip file with your code and instructions for running it through Canvas. If you miss class that day, then submit on Canvas, so I know you completed the assignment, but still bring a printed copy the next time you come to class. Your paper should describe the following:

1. The problem you are trying to solve (in your own words);
2. How you set out to solve the problem, e.g., which GA software you used, its basic algorithms (what kind of selection, crossover, mutation, etc.);
3. The representation (how did you represent the model parameter values to the GA?);
4. The basic parameters for your runs (population size, generation time, crossover rate, mutation rate, etc);
5. Experimental design (what experiments you ran and why);
6. Experimental results, as detailed above. In addition, your report should include:
   (a) Evolution curves: These are plots that show generation vs. fitness for one sample run, or more if you have interesting GA behavior to show.
   (b) Statistical analysis: How much does the performance of the GA vary from run to run? How different are the solutions that your GA finds on different runs? Document how you aggregate performance across many runs.
7. A listing of code that you wrote, as an appendix of the report if it is compact and standalone (not counted in the page estimate), in addition to the submitted source and executables.