Homework Module:

Evolving *Colonel Blotto* Strategies using a Genetic Algorithm

Purpose: Gain experience with genetic algorithms and coevolution.

1 Background

Colonel Blotto is an abstract war game that generalizes to many situations involving competing strategies of resource distribution. In the military context, it goes as follows:

> Colonel Blotto has a limited number of soldiers, S, and a fixed number of battles, B, to be fought during a war (against one opposing colonel). In each battle, the commander who shows up with the most troops wins, and the commander who wins the majority of the B battles wins the war. The colonel's job is to decided, prior to the first battle, the distribution of troops for **all** the battles.

A parallel version for pacifists:

> The country of Acirema is divided into B bastions known as *setats*. During the presidential election - which usually boils down to a 2-person competition between the top members of the two main political parties - each candidate is given a fixed amount of money, S, which (s)he can distribute among the local compaigns in the B setats. Normally, there is an extremely tight correlation between campaign investment and election success in each setat, and the winner of the presidential election is the candidate who wins the most setats.

A representation for a Blotto strategy is simply a list of B non-negative integers (that sum to S). A competition (e.g. war) between two strategies entails comparing corresponding strategy entries (e.g. opposing troop allocations for a battle), tallying battle wins, and crowning the war winner as the strategy with the most battle wins. Importantly, the **degree** of battle victory does not count, only the binary act of winning or losing. So to beat the opponent 7 against 6 in a battle is just as good as winning 25 to 6, and, in fact, given a fixed resource amount, S, a 7-to-6 victory is preferable, since it means that more of your resources can be distributed among the other battles.

Table 1 displays three different Blotto strategies, where B = 4 and S = 20. In a war between Colonels Bull and Rawhide, the latter would win battles 1 and 4, lose battle 3, and battle 2 would be a tie. Hence, Rawhide

would win the war. If Colonels Bull and Grit square off, the former wins 3 of the 4 battles. However, when Grit meets Rawhide, it's a draw, as both win 2 battles.

Colonel	Battle 1	Battle 2	Battle 3	Battle 4
Bull	8	1	6	5
Rawhide	9	1	0	10
Grit	6	6	4	4

Table 1: Three different Blotto strategies for three different colonels

2 Assignment

Your task is to enhance the evolutionary algorithm (EA) written for an earlier homework module to evolve Blotto strategies. Each strategy will compete (i.e., have a war) with every other strategy in the population. Each war win is worth 2 points, and each tie is worth 1; losses are worth 0. Fitness should be directly based on the total number of war points.

Each GA chromosome (i.e. genome) should code for the strategy. To simplify this representation, particularly the need for all resources to sum to S, use the following coding:

> If a war consists of B battles, let each genome have B binary genes, each of which codes for an integer between 0 and 10. These can represent *weights* for each battle, not absolute resource allocations. Simply normalize these weights (so that the normalized values sum to 1) and then multiply each normalized weight by S; now the weights will sum to S and can effectively represent resources, although they will not all be integers. To simplify even further, just assume that S = 1 and use the list of normalized weights as the strategy. Nothing is lost by assuming that S = 1 instead of, say, 1000.

As added twists to the problem, incorporate these two factors into your code for simulating a war:

1. Troop redistribution - If Colonel A wins a battle with X troops versus Colonel B's Y troops, then the (X - Y) *extra* troops can be *reployed* (i.e. re-deployed) in all succeeding battles (in Colonel A's strategy), by evenly distributing R_f(X-Y) resources among the remaining battles, where $0 \leq R_f \leq 1$ is the *reployment fraction*.
2. Strength reduction - Each strategy has a *strength* factor that begins each war at 1.0 but decreases by L_f (the loss fraction, $0 \leq L_f \leq 1$) each time the strategy's army loses a battle. This is akin to the defection of soldiers from an army that appears to be losing the war. In any battle, each strategy's resource is multiplied by the strength factor before comparing it to the opponent (and determining the battle winner).

Both R_f and L_f are parameters that the Blotto-EA user can tune.

The visualization of each EA run must involve (at least) 3 things:

1. A fitness plot, showing:

- The fitness of the best individual in the current generation
- The average fitness across the whole population
- The standard deviation of the fitness across the whole population

2. A plot of the average strategy entropy (see below) in the population (for each generation)
3. A listing of the winning strategy for each generation

Feel free to add any other visual aids that help you analyze the runs.

The *strategy entropy* for strategy s, denoted H(s), is computed as follows:

$$H(s) = -\sum_{i=1}^{B} p_i log_2(p_i) \tag{1}$$

where p_i is the fraction of the total resource that is devoted to the ith battle - these are fractions from the original strategy, not those modified by reployment or strength fractions. Be sure to note the minus sign in front of the summation. The significance of entropy in this context should become clear as you perform various runs and observe the dominant strategies.

Once implemented, your Blotto-EA system should be run with various combinations of values for the 3 key *war parameters*: B, R_f, and L_f. B should be tested at several points in the range [5,20], while the two fractions should be tested at 0, 1 and a few intermediate values. You should not need a population size greater than 20 (strategies) to get interesting evolutionary behavior, but you'll want to run for several hundred generations each time.

In total, you must do at least 18 *base* runs involving all combinations of:

1. B = 5 and B = 20
2. $R_f = 0$, 1 and at least one intermediate value
3. $L_f = 0$, 1 and at least one intermediate value

Come up with a simple set of descriptors for each run, such as:

1. Convergence to one stable strategy
2. Periodic shifting between 2 (or a small set of) powerful strategies
3. Continous shifting between a whole host of (very) different strategies.

Entropy measurements may help in this description as well.

For each of the 18 cases, document the 3 war parameter settings and the simple descriptor, and include all 18 in a single table.

Be sure to look carefully at the output of each run, since some strategies may appear slightly different but be essentially equivalent. For example, the allocation of 0.25 resources to battle 1 and 0.13 to battle 2 is

probably about the same as a 0.23-0.15 allocation to those same two battles. So be prepared to detect patterns in the series of strategies that are streaming across your screen.

Take 3 of the 18 base cases that are significantly different from one another and describe them in detail. In your report, include fitness and entropy plots, along with your descriptions, for each of these 3 *signature* runs.

Note that this is an example of **competitive coevolution**, since every member of the population competes directly with all others in order to assess it's fitness. As with most coevolutions, the absolute value of the population-best fitness will not increase monotonically, even when elitism is employed in the selection mechanism. So don't expect a nice *staircase* fitness progression. Instead, look to see shifts in the dominating strategies, at least in most scenarios (i.e. with most settings of the 3 war parameters).

3 Deliverables

1. A sketch and description of the representation that you used for the GA chromosome, and how you convert it into a Blotto strategy. **(1 point)**
2. A discussion of the significance of strategy entropy, H(s). What general characteristic is it measuring in these strategies, and why is that of interest? **(1 point)**
3. A table showing the settings of the key EA parameters (population size, mutation rate, crossover rate, degree of elitism, etc.) that you found most useful for this problem. These may vary among the different runs, but the default values (found by your experimentation) should be documented. **(0 points, but helpful for us in analyzing your results)**
4. The table summarizing the 18 (or more) base runs, along with an indication of and brief description of the more important data points in the table.**(1 point)**
5. Fitness and entropy plots, along with informative descriptions, of the 3 signature cases. **(2 points)**
6. A brief discussion of what you have learned about **coevolution** from doing these experiments. **(1 point)**

Your report should be 5-10 pages long, including fitness plots and other diagrams.

3.1 Warnings

In general, the programming of EA solutions to hard problems involves two significant phases:

1. Coding the genome representation, genetic operators, genotype-phenotype mapping, fitness function, etc.
2. Tuning the parameters of the system to actually solve the problem.

The second phase, tuning, often requires **much** more time than the first phase. Be aware of this when scheduling time for this (and other) EA assignments. Just getting the system to run, bug-free, is normally half (or less) of the total work.

Depending upon the actual course and semester - your instructor uses this module in various courses - you may or may not be required to do any or all of the following:

1. Demonstrate this module to the instructor or a teaching assistant.
2. Upload the report and/or code for this module to a particular site such as *It's Learning*.

Consult your course web pages for the requirements that apply.