COMP SCI 5401 FS2017 Assignment 2b

Stuart Miller
sm67c@mst.edu

November 18, 2017

1 Overview

Assignment 2b provides an improvement to the Iterative Prisoner's Dilemma (IPD) problem by employing a Genetic Programming Search across single agents evaluated by a tit-for-tat strategy search.

2 Algorithm Strategy

The algorithm works by generating a population of random trees (50% full depth trees, 50% grown tress), then iteratively generating children based on tree crossover from parents. Children are occasionally mutated, then simulated by playing tit-for-tat to obtain a fitness rating and added to the population. To obtain this fitness rating, each agents is plays a set number of rounds on a tit-for-tat strategy (the agent's opponent chooses what the agent itself did last). An option is also present to score each agent's fitness across multiple runs of IPD, with multiple randomized initial memories. Doing so allows for a more accurate fitness rating. After each generation, the population is thinned by removing poor agents chosen in a k-tournament.

3 Analysis

The main problem with playing IPD tit-for-tat is that the agent very quickly learns that it can get into a pattern since the opponent is always guaranteed to do what it did last. It is very easy to an agent to get into a back-and-forth rhythm with its opponent with they simply alternate confessing and defecting. In such a way, all the agent has to do is choose what the opponent did an odd number of turns ago and it is guaranteed to score a 3 every time. Since there is a set number of startup round before payoffs are counted, the agent can establish this rhythm no matter what the initial memory is. This is evident in that the global best solution will almost always be O[1,3,5,...]. The agent is discouraged from branching out further because this would simply decrease its fitness do to the parsimony coefficient. The only way to break out of this would be to ditch the tit-for-tat strategy and implement a co-evolutionary approach. Perhaps even playing the against against a randomized opponent would fare better.

This fallacy is visible in Figure 1. Notice how the best run simply approaches 3 (minus the minimal parsimony deduction for have a very small tree). Also notice how useless the boxplot is due to the non-diverse final population. There are only three possible fitness values. The all scored the same but had different parsimony penalties for size 1, 2, or 3 trees.

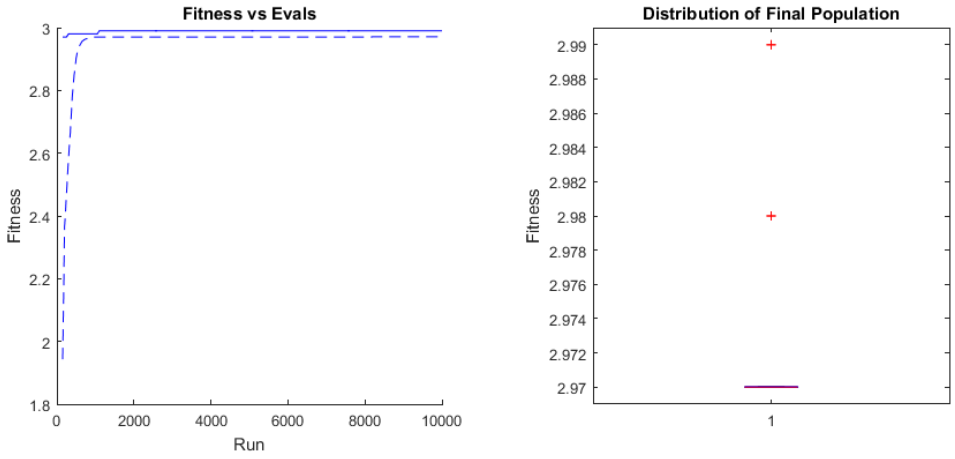

Figure 1: Plot of Best Run

4 Analysis vs. Random Search

Figure 2 shows the statistical analysis for the average final best fitness per run. Keep in mind that the Assn2b (Variable 2) results inclued the parsimony coefficient whereas the Assn2a (Variable 1) are raw average of the payoff. Assn2a saw higher averages as it was occasionally able to get up to a 5 payoff at the end, where Assn2b only ever settled on a 3. Conversely, though Assn2b got to a 3 (minus parsimony) every single time while Assn2a often couldn't get past a best of 1. These averages prove that the approaches are roughly equal. Assn2b is more stable, but suffers from the restrictions discussed earlier in Section 3.

F-Test Two-Sample for Variances

	Variable 1	*Variable 2*
Mean	3.10752	2.979333333
Variance	2.039359092	8.91954E-05
Observations	30	30
df	29	29
F	22863.94858	
P(F<=f) one-tail	2.44135E-56	
F Critical one-tail	1.860811435	

t-Test: Two-Sample Assuming Unequal Variances

	Variable 1	*Variable 2*
Mean	3.10752	2.979333333
Variance	2.039359092	8.91954E-05
Observations	30	30
Hypothesized Mean Difference	0	
df	29	
t Stat	0.491639913	
P(T<=t) one-tail	0.313337346	
t Critical one-tail	1.699127027	
P(T<=t) two-tail	0.626674692	
t Critical two-tail	2.045229642	

Figure 2: Statistical Analysis