Genetic Algorithm Project Report

Ian Wakely

Problem 1

Implementation

For this implementation of a genetic algorithm, the population size is set to 150 members. For initialization, each of the members are issued random numbers within the ranges defined by the input xrange argument. After the population has been initialized, the fitness of the initial population is evaluated for the first time to prepare for future iterations. All of the members of the population are stored in a 2-dimensional matrix, where each of the columns represent each of the dimensions that a member is resent within.

For each iteration, the fitness values from the previous iteration are normalized in such a way to make the smallest values have the highest values, then the cumulative sum is calculated for each of the members within the population. After the population has been normalized, random members are selected from the population for reproduction using the normalized cumulative sum.

Using the new population, members are selected for crossover or mutation by looping over the population and randomly generating a number to decide if crossover will happen. The phenotype is converted into is genotype only when crossover or mutation happens to reduce on processing time. The genotypes that are used for crossover and mutation are IEEE double precision floats. Each of the values are converted into the 64-bit binary representation of the original value.

For each of the dimensions, a random integer is generated to determine where a crossover or mutation to occur. For crossover, a single point crossover occurs with the next population member at the random index that was generated. For mutation, a single bit is flipped at the random index. After crossover and mutation is complete, the genotype is converted back into a phenotype, the previous round of data is saved, and the new population is evaluated for its fitness. The population is re-evaluated until the difference between the new population and the previous population reaches a threshold of 0.000001.

Performance

For testing of the genetic algorithm, the xrange values that were used were 15, 10, and 5 for the 3-dimensions tested for the Dejong benchmark, the Ackley benchmark, and the Rastrigin benchmark.

Table 1: Dejong Benchmark Performance

Best Fitness	Time (ms)
0.0234	78.337
0.3833	65.73
0.4499	81.542
0.6715	67.18
0.0072	80.985
0.7632	75.856
0.1084	83.862
0.1501	83.524
0.2931	70.914
0.3066	71.236

Table 2: Ackley Benchmark Performance

Best Fitness	Time (ms)
2.4294	64.665
2.7754	68.068
1.2691	85.985
2.7092	73.154
3.2546	71.14
2.4644	67.293
2.8304	59.58
2.3753	69.431
2.2376	66.634
1.9214	72.06

Table 3: Rastrigin Benchmark Performance

Best Fitness	Time (ms)
7.8515	67.64
6.1758	72.687
10.6407	71.795
6.6215	81.021
7.4068	72.064
6.7165	75.204
6.3433	66.612
6.3543	82.647
7.4474	70.308
9.4431	66.022

Problem 2

The Prisoner Dilemma is a thought experiment that based on around the idea that if an individual does what they believe is best for themselves, then they will gain a more favorable outcome, when it probably won't be. The Prisoners Dilemma can be represented in a variety of different games and online, where all the players are given a choice to do an action, or to not do the action. If all the players make the same choice, then they all will be penalized. Alternatively, if some of the players take the action, and some don't, then the ones who don't will be more severely penalized.

For this instance of the Prisoners Dilemma, we are simulating a string of crimes between two partners, or players, and we are using the break down that can be found in Figure 1. For this breakdown, if both players decide to testify, then they will each receive 3 years in jail. If they both remain silent, then they will each receive 2 years in jail instead of 3. If one of the players testifies, and the other remains silent, then the silent one will receive 5 years in jail and the one who testified will be set free.

	P2 testify	P2 silence
P1 testify	P1 = P2 = 3 yrs	P1 goes free, P2 = 5 yrs
P1 silence	P1 = 5 yrs, P2 goes free	P1 = P2 = 2 yr

Figure 1: Prisoners Dilemma sentence breakdowns

In this variation of the Prisoners Dilemma, known as the Iterative Prisoners Dilemma, each of the players has some history available to them, allowing them to make informed decisions on what they should do based on what the other player(s) have done previously. For this assignment, each of the players have the history of the 2 previous decisions of the other player(s), as well as the history of their own decisions.

With this extra information, players can try to predict the other player(s) stratagy to try and beat the opponent(s) and minimize their own prison sentence. If a player is able to predict what another player's action is going to be, then you can try and take the best action for themselves. For the jail times found in Figure 1, it is adventagous for Player 1 to always testify if they are looking for the least amount of prison time, which is kind of expected. If the desired outcome is to share the same amount of time as Player 2, then it is adventagous to pick the same option that you predict Player 2 will choose. Assuming the players utilize the information that they have available to them, then they should change their behavior in such a way that should get them better results towards their goal, and act less like the original Prisoners Dilemma where the players have no historical information.

Implementation

For this implementation, each iteration has 4 possible outcomes given the decisions of the two players. As a result, each outcome is stored as an integeter from 1 to 4. The number of bits

required to represent the decisions a single player can be evaluated using Equation 1, where n is equal to the number of iterations you can remember, including the current one. Meaning that if a player can remember the 2 previous iterations, n will equal 3, and it will require 21 bits of information to represent it. For no memory, then n will equal 1, and the number of bits to represent just that decision is 1 bit to represent the 4 possible options.

$$number\ of\ bits = \sum_{i=1}^{n} 4^{i-1}$$

Equation 1: Number of bits

The fitness function for each of the inmates first locates the block of final decisions that an inmate makes, which are have already taken into account for their historical information, and counts how many of the 4 available options were selected between the final decisions. It then will multiply each occurance of the final decisions by the penities accociated with them to get a vector of the total jail time. Fitness is calculated by taking the average of the jail time from the final decisions for both player 1 and player 2. Strategies with a lower average jail time for player 1 will be more favorable over higher averages.