TRINITY COLLEGE DUBLIN, SCHOOL OF MATHEMATICS

Assignment #2, Module: MA5611

Gustavo Ramirez

January 11, 2017

1 PROBLEM DESCRIPTION

The goal of this assignment is to develop a framework for implementing genetic algorithms in parallel using MPI. Then using this framework you will develop strategies for playing the iterated prisoner's dilema.

1. Task 1

   Write some serial C code that implements the functionality required for a genetic algorithm. Steps involved include

   - Selection of parents based on fitness
   - Crossover of parents
   - Random mutation

   Assuming you use the memory efficient method of representing chromosones, the crossover and mutation operations will involve bitwise arithmetic and so you will need to become comfortable with the following operators ~& ^ |

Use the simple example from class (number of 1s in a string) to test your code with different string lengths, population sizes, mutation rates etc. Make sure to support the case where the chromosone is longer than 32 bits - ie you need to use an array of ints.

Hint: A useful trick is to graph the total fitness of the population over time. This should increase. If it is not, then there may be something wrong with your code!

2. Task 2

   Modify the fitness function so the code plays the iterated prisoner's dilema. Your strategy for playing should be based on the outcomes in the previous two games. As noted in class, this gives 16 possible combinations which can be stored in an int. You will need to figure out a way of mapping the outcomes of previous games to the bits in the int.

   You will also need to come up with a method to determine your tactics in the first two games. This can either be hardcoded, random or added to the chromosome encoding of the game.

   Allow command line arguments to be given specifying

   - Size of population
   - Number of generations
   - Number of iterations of PD to play
   - Crossover rate (usually high)
   - Mutation rate (usually low)

3. Task 3

   Using the Master-Slave model, implement the genetic algorithm developed above in MPI. The master process should handle the selection, crossover and mutation. Each slave should handle the fitness evaluation of members of the population.

   Do you see the overall fitness continue to increase or does it do something else? Can you explain what is going on?

4. Bonus Tasks

   Investigate the island model for genetic algorithms and add this functionality to your code. Add an extra command line argument to switch between using the island model

and the normal model. Change the game from Prisoners' Dilema to Rock-Paper-Scissors where the strategy is just a single choice that is always played. Make a win worth 10, draw 2 and loss 0. Graph the fitness per chromosone type (Rock, Paper or Scissors). Explain what you see. Submit a tgz with your code files, graphs and a 2(ish) page writeup of what you did and any observations you have made on the behaviour and performance of your code.

1.1 SOLUTION

- **Task 1**

  For the implementation of the genetic algorithm:

  1. generate the population with its values randomly set. This is only done once
  2. calculate the fitness for all the members of the population combined and separated
  3. calculate the probabilities of being selected, each of them proportional to their fitness, and normalized. These probabilities are useful in the next step, in the sense that when a chromosome is to be selected, it is done through the wheel roulette selection.
  4. crossover step. X% of crossover rate means that percentage of the next generation is composed of crossover offspring (and for creating two new elements of the offspring, two elements of the current generation are chosen through wheel roulette selection and then crossed), and the rest of the empty places are filled in doing wheel roulette selection again. The two chromosomes from the current generation crossover after first bit to give the new offspring. **If the population is n, and the fractional crossover rate is c_r, then this step loops nc_r times; in each iteration of this loop, a pair of current chromosomes is selected, crossed, and the offspring created and stored. Additionally, it loops $n - nc_r$ more times, to fill the empty places.**
  5. mutation step (acts over the just created offspring). Y% of mutation rate means to flip each bit with a probability Y%. **If the population is n, the chromosome size s, and the mutation rate m_r, then this step loops nsm_r times, but without flipping two times the same bit. For this flipping process, all of the ns bits of the population are taken into consideration.**

  Perform steps 3, 4 and 5 n_g times, where n_g is the number of generations.

- **Task 2**

- **Task 3**