## Machine Problem 1:

## Genetic Poker Hands

Due 24 Sept, 11:55 PM via Latte

The goal of this assignment is for you to experiment with and experience evolutionary computation. You start with a population (say 50 or 100) of random worthless card hands, and through mutation and crossover, with feedback from a fitness function, gain better and more valuable hands over time, perhaps ultimately leading to a 4-of-a-kind or a straight or royal flush. Here are some of the elements you will be writing:

1. Representation. You make up your own representation for a card and for a hand of 5 cards. For example, a bit string of 52 0’s with 5 1’s indication the 5 cards by position. Or 5 groups of 17 bits representing each card. Or 6 bits per card instead of 17 (with 4 bits indicating the card and 2 bits indicating the suit). Or 5 real numbers with the integer part being the card and the fractional part being the suit. Your choice of representation will affect the mutation, crossover, and evaluation functions. You also have to decide whether to keep each hand sorted.
2. Mutation Operator. You should be able to mutate a hand by changing one of the cards to a completely different card, or shifting number or suit of an existing card. This might involve changing bits or deleting a card and dealing a new one. You also need a LEGAL function to see if the new hand is legal, e.g 5 cards, no duplicates.
3. Crossover: taking two hands and picking a number from 1 to 4 then choosing the first n cards from one parent and the remainder from the other parent. Again you need a LEGAL filter on the new hand.
4. Fitness. This is often the hardest part of a GA project. You need a function which can score a poker hand and/or compare two hands. The hierarchy is well known but in comparing hands there are many subtleties:
   1. Royal flush
   2. Four of a kind
   3. Straight Flush
   4. Full House
   5. Flush
   6. Straight
   7. Three of a kind
   8. Two Pair
   9. Pair
5. The Loop. Genetic algorithm is a loop usually for a limited number (say 100 or 200) or until the maximum fitness of the population or average fitness of the population passes a minimum threshold. First you initialize the population with random hands, eliminating any hands which already have high scores.
   1. Evaluation: Apply fitness function to each member of population. It will help with proportional fitness if the evaluation returns larger numbers, like 100 for a straight flush versus 2 for a pair.
   2. Selection: Using proportional fitness or tournament selection, select parents for next generation.
   3. Reproduction: Populate next generation with a combination of crossover (80%), mutation (10%), and elitism (10%). Elitism means copying a selected hand with no mutation, an attempt to prevent backsliding
6. After each generation, print out the generation number, the maximum fitness across the population, and the hand which achieves it.

What to hand in with 3 or 4 files through Latte:

1. Your source code including documentation and citation of components you didn’t write yourself.
2. Description of what you did and how it worked. This is not a well-known problem so it may not be possible to ever reach 4-of-a-kind or a straight flush. Why?
3. A run (text file) showing typical behavior of your genetic algorithm
4. (Optional) A graph averaging max fitness over 50 runs