For this project I implemented two variants of MC-RAVE based on the stock MCTS to be used on Gomoku. One without exploration bonus (MC-RAVE), and one with exploration bonus (UCT-RAVE). I was able to successfully determine an ideal k parameter = 0.8 for MC-RAVE given the number of simulations to be 2000. This was mostly done manually. (mc-rave game 2000 0.8) performed reasonably well when pitched against itself, a human player, or minimax. However, I was unable to determine a good combination of k and c parameters for UCT-RAVE (2000 simulations).

In addition, I implemented a simple genetic algorithm to determine what may be the best value of k for MC-RAVE (2000 simulations) on 7\*7 board. The fitness of evolution was determined by the outcomes of 10 matches against Minimax (cutoff = 1), 5 of which have MC-RAVE as the firsthand player and 5 have Minimax as the firsthand player. Losing to Minimax would result in decreased fitness whereas winning would result in increased fitness (drawing would neither deduce from or add to fitness). All fitnesses in this generation would then be scaled to add up to 1, and the individuals with the highest fitnesses would get to reproduce themselves by a slight mutation (the difference caused by mutation is sampled from a Gaussian distribution with mean = 0 and standard deviation = a free parameter manually specified) for a number of times proportional to their scaled fitness values. The number of individuals after reproduction was capped by the population size for each generation, which meant that low fitness individuals who didn’t get priority to reproduce would just die out. The algorithm took 3 free parameters — the number of generations, the population size for each generation, and the standard deviation value used in mutation, i.e. (driver num\_gen pop\_size mut\_sd). I tried to run (driver 5 5 0.02) via ssh; however, after only finishing the first generation, my connection was cut, presumably because of safety mechanisms. Here’s what’s left in the log file: *0,(1/5 2/5 3/5 4/5 1),(10 9 8 13 11),(0.6287039 0.39158812 0.4138195 0.20123567 0.19979233)*, which are, in the order of presentation, generation number, k values tested in this generation, fitness values for each k (in the same order), and the k-values that would participate in the next generation. Even though the genetic algorithm run could not be finished in time, I’m fairly confident the algorithm itself is bug-free since I’ve tested it extensively on downsized problems. Please see my comments in evol-alg.lisp for more details.

My work therefore is mostly concentrated in MC-RAVE.lisp and EVOL-ALG.lisp, all clearly commented.