Kunal Purohit

011877606

Cs 420 – Project 2

I implemented both the Hill Climb Random Restart and the Genetic Algorithm as per the textbook. The Random Restart Algorithm randomly generates a board state of n (in this case 21) queens, and then iteratively generates the children states and pushed them into a priority queue. Once a minimum state is achieved, the process restarts at another random state, and the program keeps track of the most optimal state. The process continues till the number of restarts is completed, or an optimal solution is found. The genetic algorithm generates a random initial state, and then generates the possible children states as the new population. The algorithm then randomly selects two states and breeds them by choosing a random index to split both states. The algorithm can then alter the resulting state through mutation, which randomly selects a queen and mutates its value. This process continues till the resulting state has the optimal fitness or the number of set iterations is completed.

For the random restart, the program will run 100 different instances of the algorithm, and the algorithm will run 40 random restarts per instance. In general, the algorithm performs proficient under basic parameters. For 21 queens, with 40 restarts per run, a total of 100 runs indicated that the algorithm reached an optimal solution 35% of the time. Moreover, the average solution cost is roughly 1, which indicates that the algorithm consistently reaches close to the optimal solution, but runs into local minimums for a majority of runs. This makes sense, as the end state that can be yielded is significantly dependent on the initial state that is generated. The Average run time is 284.89 milliseconds per run, which is relatively proficient considering the number of operations per run.

For the Genetic Algorithm, the program also ran on 100 instances of the algorithm. The algorithm’s parameters were set to have a 50% mutation rate and also run the algorithm to a max depth of 50. With the given configuration, the algorithm yielded close to a 95% passing rate. The average run time of each run was less than one millisecond, and the search cost was close to 1700. The Genetic Algorithm cost more in space compared to the Random Restart algorithm, but executes much faster and more efficiently. The passing rate of the algorithm is surprising; the idea that randomly generating children states will yield the optimal solution is counter intuitive. Nevertheless, this makes sense, as the mutations and random selection of parent states lead to less chances of running into local minimums.