Group 15: Ying Fang, Mengdi Li, Mingquan Liu, Daniel Wivagg
Professor Joe Beck
CS 534 Artificial Intelligence
February 12, 2018

Assignment 1

Part 1

1. How large of a puzzle can your program typically solve within 10 seconds using A*? Using greedy hill climbing?
    a. The A* algorithm can typically solve a board of size n=7 in less than 10 seconds. If n>7, the program usually takes longer than 10 seconds. However, the time heavily depends on the difficulty of the starting position. For example, when the queens are all in position 0 at the start (meaning they are in a row at the top of the board) the program takes sufficiently longer than it does with many of the randomly generated boards. This is because those boards are usually closer to a solution just because of their randomness.
    b. As for hill climbing, typically, we can solve a board of size n=32~34 within 10 seconds. With a good random start status, sometimes we can solve a board of size n =37 in less than 10 seconds.
2. What is the effective branching factor of each approach? For this computation, perform 10 runs of a puzzle half the maximum size you calculated for step #1.
    a. A*: average effective branching factor is 8.9 with a board size of n=4 and using iterative deepening.
    b. Hill climb: average effective branching factor is 272 with a board size of n=17.
3. Which approach comes up with cheaper solution paths? Why?
    a. A*, because hill climb only looks for the best step of the current situation and does not look back to how much it has spent. Also hill climb may not form a path since it could restart which totally destroys the path. For A*, it is finding the optimal solution based on distance traveled and heuristics, which leads to the cheapest solution. In conclusion, A* always finds the optimal solution and hill climb may not find the optimal solution, so A* comes up with cheaper solution.
4. Which approach typically takes less time?
    a. Hill Climb

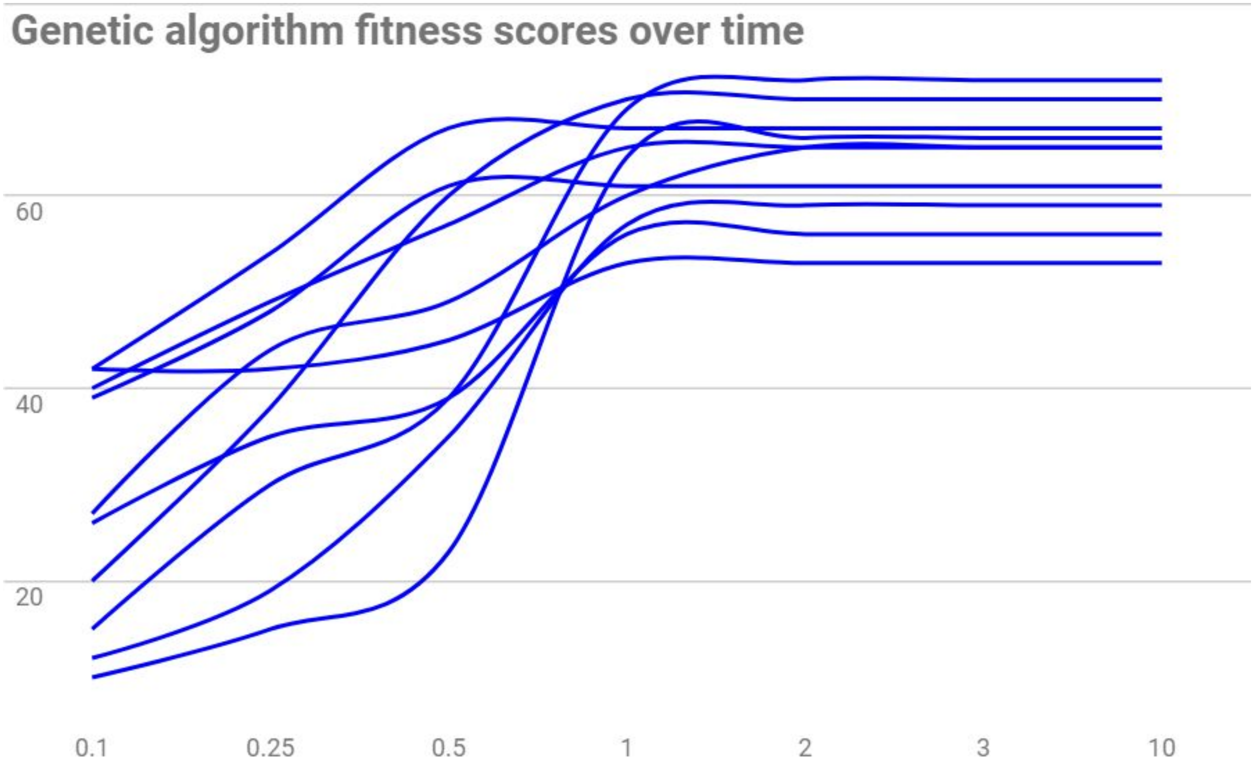
Part 2

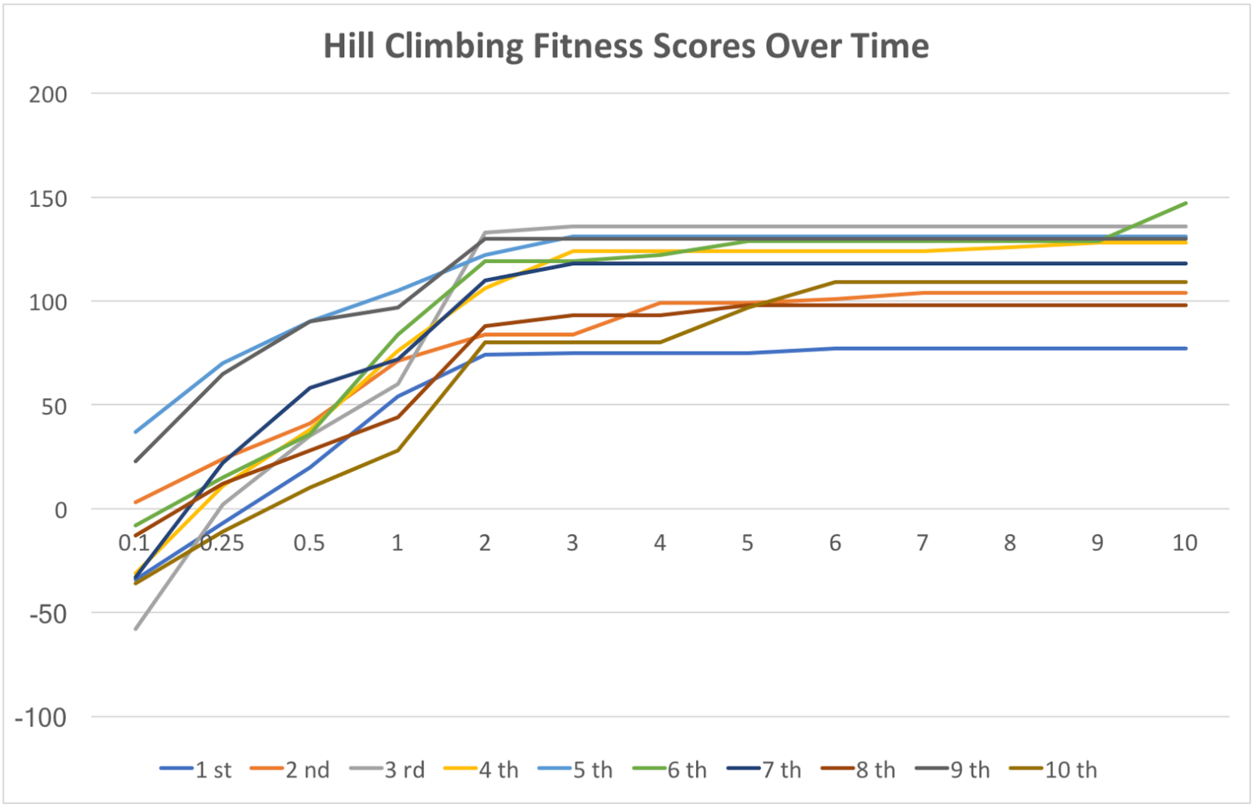
1. Explain how your genetic algorithm works. You must describe your selection, crossover, elitism, culling, and mutation approaches.
    a. The crossover function is a partially mapped crossover (PMX) algorithm. This method preserves the correct number of industrial, commercial, and residential

sites in each map, so that the crossover does not create illegal combinations. In order to use this algorithm, the map dictionary must be converted into a strand of "DNA" in the form of a list with each space on the map assigned a unique label. Then, the algorithm takes a section of each parent and places it in that section of the child's DNA. Finally, it fills in the remaining spaces by mapping each missing value of a child to its corresponding place in the opposite parent.

b. For selection, we calculate the probability distribution of selecting parents based on its fitness score. In the actual calculation, we use the minimum fitness score as the floor of the values. The difference between current value and minimum value is taken for every parent and that value added to a running total. Then each of difference is divided by the total count to get the probability of each parent to be selected.
c. As for elitism, we keep at least one, at most 2% of the current population for the next generation.
d. As for culling, we remove at least one, and at most 2% of the current population for the next generation
e. As for mutation, we give it 2% of change to happen during crossover. When a board is mutated, the entire board is regenerated in order to inject as much randomness as possible into the process.

2. Create a graph of program performance vs. time. Run your program 10 times and plot how well hill climbing and genetic algorithms perform after 0.1, 0.25, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10 seconds. If you only had 0.25 seconds to make a decision, which technique would you use? Does your answer change if you have 10 seconds?





If we only have 0.25 seconds to make a decision, we will use Genetic Algorithm. While, if we have 10 seconds to make a decision, Hill Climbing performs better.

3. How do elitism and culling affect performance?
    a. Elitism keeps the elite offsprings of current population, which could keep genetic algorithm not getting worse.
    b. Culling removes the bad offsprings of the current population, which reduces the possibility of getting bad children in next generation.
4. How did you perform selection and crossover for your population? Find some known method for doing selection, other than the one described in class. Explain what you did.
    a. As for selection, we used fitness proportionate selection as the select method. Fitness proportionate selection uses fitness score to calculate the proportion of each individual. This is achieved by dividing the fitness of a selection by the total fitness of all the selections, thereby normalizing them to 1. Then we randomly select the parent based on the resulting probability distribution.
    b. As mentioned above, crossover was done using a PMX algorithm, which created children that were “partially mapped” to each of the parents. This method involved copying a portion of each parent to the corresponding child, and filling in the missing pieces from the opposite parent, making for a shuffled map that still had all the required items in it.