Alan Huang
Dr. Daisy Tang
CS 4200

CS 4200 Project 2 Report

Approach:

The steepest-ascent hill-climbing algorithm is very simple to implement as a coding project. In the beginning, I was a little confused about how to solve the n-queen board. I thought I supposed to solve each row's queen one by one with their total hitting number, so I went to professor office hours to discuss how should I approach to solve the 8 queen board. After discussed to the professor I realized that I should generate all 56 possible locations and select one of the best cost locations as the chosen node to replace the current state node. The while loop will run until the current location became a solution puzzle, or while the loop will run until child node cost is higher than the current node then the program will exit and return a false to indicated that no solution found. After successfully testing the code I let the program run 100 times by itself and calculated to the success rate. I was surprised to see that percentage is very close to 14%, so that means my steepest-ascent hill-climbing algorithm was correctly implemented.

I choose the simulated annealing algorithm as my second part of the project. The simulated annealing algorithm is pretty simple for my node's class function to implement it, and I don't have to modify the node class too much. The hardest part to implement this algorithm was to understand the meaning of the pseudocode code from our textbooks. It took me a while to figure out what the schedule mapping function supposed to do when the time increased. After watching the lecture and reading the textbook I realized I only need to decrease the temperature when the time increase. I just set the temperature to 1000 and use temp --; to slowly decreased it when the for loop is running.

The program will run and selected a random queen to move and see if the random location has less attacked queen. When the delta E is greater than 0 it will change the current node to the best cost random node. When delta E is negative the exponential function will stay lower than 1, and when the temperature number is high the exponential function number is very closed to 1. When the temperature starts to decrease the exponential function number will start getting close to zero. I first set my program to select worth cases when the probability is greater than 0.8 and my program success rate is 0 percent. I thought I did something wrong then I started to play with the number then I accidentally figured out that if I see the number very very close to 1 then the program success rate will increase. In the end, I chose only when the probability is greater than 0.9999 to accept the worth cases because the success was over 97%.

Analysis:

In the textbook, it showed that we are supposed to let the simulated annealing program exit when the temperated is zero, but I also made an exit in my program to see if the success rate could be changed by this factor.

Table: Hill climbing compare to Simulated Annealing algorithm (Simulated Annealing algorithm with exit)

	Average Percent		Average Cost		Average Time	
N	Hill climbing	Simulated Annealing	Hill climbing	Simulated Annealing	Hill climbing	Simulated Annealing
100	16%	97%	175	973	45281 ns	425366 ns
500	17%	97.2%	185	931	57406 ns	368519 ns
1000	15.2%	97.7%	183	866	46245 ns	331084 ns
1500	14.3%	97.3%	182	909	44726 ns	343598 ns
2000	14%	96.3%	180	1013	42683 ns	377723 ns
2500	15.16%	97.08%	178	925	42050 ns	348397 ns
3000	15.1%	97.4$	182	904	41887 ns	343728 ns

From the data, we can see that the hill-climbing success percentage is very close to what the textbook predict which is around 14%. The hill-climbing is much quicker than the simulated annealing algorithm, but the success rate is low to solve most of the n-queen problem. On the other hand the simulated annealing success rate is always above 97%, so this simulated annealing algorithm is much better for the n-queen puzzle. Since we are only working on the fixed size n-queen problem, so efficiency is not too important for both of the algorithms. If the n-queen board size increase then the simulated annealing could waste too much execution time to find a solution. Also, from above I let my simulated annealing program exit as soon as it finds the solution.

Table: Hill climbing compare to Simulated Annealing algorithm
(Simulated Annealing algorithm without exit)

	Average Percent		Average Cost		Average Time	
N	Hill climbing	Simulated Annealing	Hill climbing	Simulated Annealing	Hill climbing	Simulated Annealing
100	16%	99%	175	9999	56876 ns	3319518ns
500	17%	98%	185	9999	69307 ns	3288192ns
1000	15.2%	97.7%	183	9999	41965 ns	3319387ns
1500	14.3%	97.6%	182	9999	43117 ns	3305038ns
2000	14%	97.3%	180	9999	42600 ns	3298906ns
2500	15.16%	97.3%	178	9999	40841 ns	3279259ns
3000	15.1%	97.4$	182	9999	40470 ns	3282224ns

The table above is for simulated annealing finished when the temperature is zero. We can see that the success rate didn’t change too much if I let the program exit when the temperature is zero, but the execution time is about 10 times larger than the previous one. The simulated annealing with an early exit could be a better solution to solve the n-queen program, but I am not sure if my program success rate is correct compared to the real algorithm.

Finding:

The simulated annealing algorithm is very interesting because the textbook didn’t mention how we should preset the temperature, and under what probability we should take bad cost neighbor. In the beginning, I let the temperate set to 100, but my success rate is only below 15%. After I changed the temperate and loop counter to a higher number, my program success rate start to increase. I also noticed when I set a probability accepted number close to 1 could also affect the success rate. I tried many different numbers for the temperate, loop counter, and probability factor I was able to find a sweet spot to increase my success rate to above 97%. This simulated annealing is a very interesting and challenging algorithm to learn.