Lab 4 - Local Search

Aidan Schneider

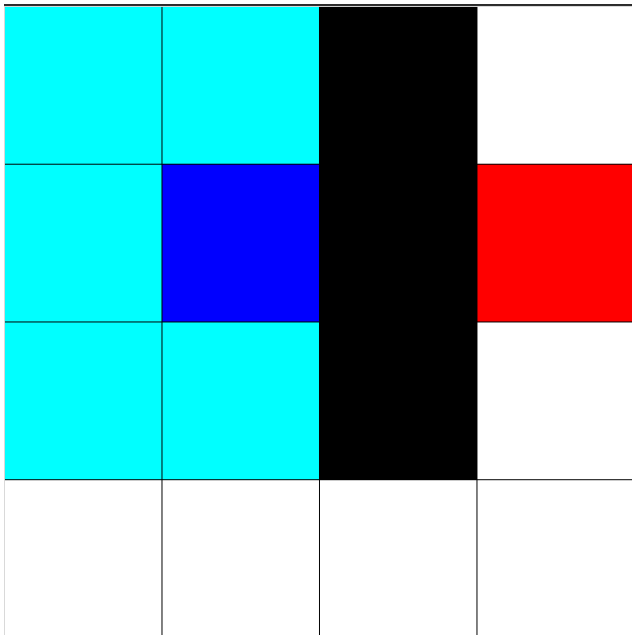
CS 2400 031

04/13/23

	BitString	Open Maze	Deceptive Maze 1	Deceptive Maze 2
Hill Climbing	100/100	100/100	28/100	21/100
Simulated Annealing	99/100	100/100	70/100	54/100

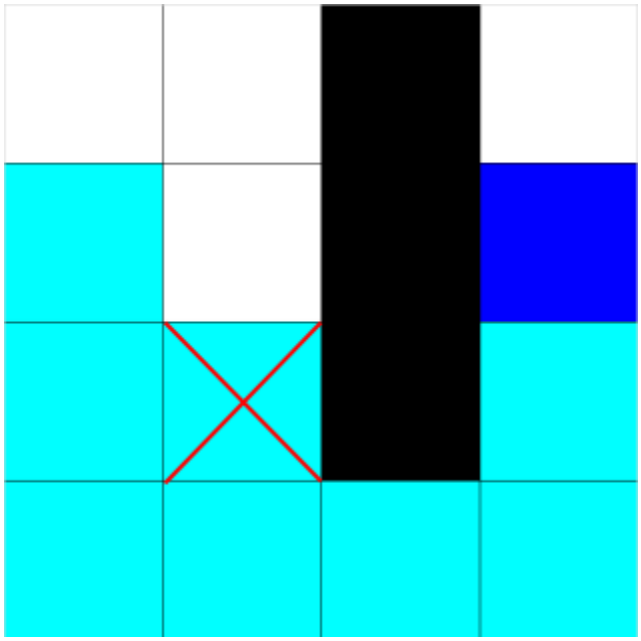
This table is showing the accuracies of the Hill Climbing and Simulated Annealing search accuracies. According to this table it seems that as a problem becomes more complicated, like with Deceptive Maze 1 and 2, the Hill Climbing method is less accurate than Simulated annealing. However, for very simple problems like BitString Hill Climbing will give the correct result every time and Simulated Annealing may not.

Hill Climbing on Deceptive Maze 1



In the above maze is a hill climbing run on deceptive maze 1. The algorithm did not find a path to the red end node and instead got stuck in a local optima. The reason the algorithm got stuck in the spot with the dark blue square is because that square has a Manhattan distance of 2 and all the neighbors around this square have Manhattan distances of 3. In this instance the restart did not help hill climbing because both restarts brought the blue square to the wrong place.

Simulated Annealing on Deceptive Maze 1



In the above maze is the simulated annealing algorithm on deceptive maze 1. This instance simulated annealing actually made it to the goal and this is likely due the fact that simulated annealing picks its neighbors at random. Because of this, when the dark blue square made it to the square with an ‘X’ in it, instead of getting stuck in a local optima the neighbor below that box was chosen causing the algorithm to get closer to the end goal.

Best Algorithm

Overall simulated annealing beats hill climbing in a landslide for the more difficult mazes (70/100 vs 28/100 respectively) and hill climbing barely edges out in the bitstring problem (100/100 vs 99/100). The reason that simulated annealing does so much better in the more difficult mazes is due to the randomness of the algorithm. This is because the successor is chosen randomly and in the simulated annealing case if the algorithm is stuck in the same local optima as hill climbing was in the first image, instead of just being stuck there simulated annealing has a chance to move out of that spot to a node next to the local optima and instead find a new way to the goal. This is shown in the second image when instead of going to the spot

with a Manhattan distance of 2, the algorithm instead went down to a further node which ended up causing it to find the end of the maze. As for the bit string problem, the reason hill climbing does so slightly better is because it's such a simple problem that doesn't contain any local optima and hill climbing will get right to the goal immediately. However for simulated annealing, the randomness that helps in the complicated problems hurts here. The reason for this is because if the goal state isn't selected as a random successor when the opportunity arises, then the algorithm could get stuck going in circles and never finding the goal.

Changing the Schedule

Lowering the 1000 to 100 in the schedule would result in a worse performing algorithm. This is because a lower number will cause the temperature of the program to go down at a much faster rate than when the schedule has 1000 in it. This means that the chances of the algorithm using a random successor with a worse fit will go down faster, which will cause the algorithm to be stuck in local optima much more often than before. This is shown when running the program with this new schedule on deceptive maze 1. Before the schedule change simulated annealing found the goal 70/100 times. However, after the change the goal is only found 13/100 times which is worse than the hill climbing algorithm.