University of Puerto Rico
Mayagüez Campus
Mayagüez, Puerto Rico

Department of Electrical and Computer Engineering

Hill Climbing Applications and Simulated Annealing

by:
Manuel Alejandro Umaña Rodriguez
Dahyna Gabrielle Martínez Pérez
Jan Luis Pérez de Jesús
Christopher Hans Mayens Matías

For: Profesor José Fernando Vega Riveros
Course: ICOM 5015, Section 001D
Date: March 27,2025

Abstract:

What is the effectiveness of heuristic search algorithms? To be able to answer this, in the following report 2 problems will be explored in more detail, those being the 8 puzzle problem and the 8 queens problem. The 8 puzzle problem consists of a 3 by 3 grid with eight numbered tiles (1-8) and one empty space where in the problem formulation was designated as 0. The objective is to slide the titles into the correct order from an initial scrambled state. The 8 queens problem consists of an 8 by 8 chessboard with 1 queen on each of the columns where the objective is for no 2 queens to be attacking each other. This report examines Steepest-Ascent Hill Climbing, First Choice Hill Climbing, Random Restarts Hill Climbing, and Simulated Annealing. Comparing their effectiveness in solving multiple instances of the problems mentioned. As hill climbing is computationally efficient it suffers from getting trapped into local maxima. The Steepest Ascent, First Choice, and Random Restart suffer from this drawback with a maximum success rates of 30%,20%, 90%, and 0% in the 8 queens problem respectively for each algorithm. Although that simulated annealing was expected to fare better than the hill climbing algorithms, it was the one that fared the worst due to the temperature application where the temperature cooled quicker than expected resulting in an incomplete solution for all attempts. In the case of the 8 puzzle problem none of the algorithms managed to solve the problem. The hill climbing variants algorithms with a good heuristic function served to solve some instances in the 8 queens problems. Simulated annealing showed the most promise of being able to surpass the limitations of hill climbing algorithms, however due to the randomness and poor optimization of the temperature it ended as the worst at solving the problems. In the 8 puzzle problem the restrictive nature of the solution makes it the easiest to fail at solving and get stuck in a local maxima as it requires the algorithm to purposely make bad moves to be able to unlock the solution. The Simulated Annealing's randomness did not show any significant advantage in any of the problems. This suggests that for local search algorithms to be effective at obtaining the goal state, it needs to be tailored to a specific kind of problem or else it won't be successful.

Table of Contents:

Abstract: 2
Group Collaboration: 3
I. Introduction: 4
II. Related Work: 4
III. Performance Approach and Results: 6
A. Hill Climbing Algorithm with Steepest-Ascent 6
B. Hill Climbing Algorithm with First Choice 7
C. Hill Climbing Algorithm with Random Restart 9
D. Simulated Annealing 10
E. Figures and Tables 12
IV. Conclusion: 35
References: 39

Group Collaboration:

The tasks amongst each team member were divided by each member's experience utilizing python, as follows:

-Jan Pérez was in charge of developing the Hill Climbing Algorithm with Steepest-Ascent Search Algorithm. He solved the Eight Puzzle Problem and Eight Queen Problem with said algorithm.

-Dahyna Martínez was in charge of developing the Hill Climbing Algorithm with First Choice Search Algorithm. She solved the Eight Puzzle Problem and Eight Queen Problem with said algorithm.

-Manuel Umaña was in charge of developing the Hill Climbing Algorithm with Random Restart Search Algorithm. He solved the Eight Puzzle Problem and Eight Queen Problem with said algorithm.

-Christopher Mayens was in charge of developing the Simulated Annealing Search Algorithm. He solved the Eight Puzzle Problem and Eight Queen Problem with said algorithm.

-All members of the group were included and participated in the video. Video editing was made by Jan Pérez.

I. Introduction:

Artificial Intelligence agents are designed with the purpose of solving a specific problem tailored to the environment. Selecting the correct algorithm is crucial in maximizing performance and finding the most optimal solutions. This report focuses on studying different search algorithms such as steepest ascent hill climbing, first choice hill climbing, random restart hill climbing, and simulated annealing by applying them to two relatively well known problems: the 8 puzzle and the 8 queens puzzle.

The 8-puzzle problem consists of 8 numbered tiles and a space in a 3 by 3 grid. Initially they would follow a random order, then the goal state of this puzzle is reached when the tiles are formatted in the following manner 1,2,3,4,5,6,7,8,0. For this particular puzzle it is required for the algorithms to consider the current state and how the next state affects the following states in the long run.

The 8 queens problem involves the placement of eight queens in a 8 by 8 chessboard. Initially they would be placed in a random order and the goal state would be reached if the position of the queens on the board did not directly or indirectly attack each other. This problem requires an algorithm that constantly adjusts the queens placements while minimizing conflicts. Since the search space is fairly large this problem poses a challenge for local search algorithms.

The results of these experiments were documented through figures and graphs, providing data that shows the strengths and weaknesses of each algorithm. By understanding their behaviour it is possible to find the most optimal algorithm for each problem.

II. Related Work:

For the environment of exercise 4.4 it was determined before conducting simulations to be fully observable, discrete, deterministic, sequential, static, and single agent. This applies to both the 8 puzzle and the 8 queens puzzle. The algorithms analyzed were mainly focused on hill climbing applications, which were modified from the search.py file in github [1]. Hill Climbing is a local search algorithm that finds the optimal solution by making incremental changes to an existing solution [2] . This process ends when the top or local maximum is reached. For this experiment the hill climbing algorithms studied offered an added robustness that made the hill climbing more effective. Each 8-puzzle and 8-queens puzzle were solved using the following hill climbing variants and the simulated annealing algorithm.

The first algorithm studied was Hill Climbing with Steepest Ascent. This algorithm is an enhanced version of hill climbing where instead of moving to the first neighboring node that improves the current state, it evaluates all neighbors and moves to the one that can offer the

highest improvement [2]. There is no direct mathematical equation that describes how it works, but it can be described by the following relation:

$$Best\ State\ =\ max(State\ that\ increases\ the\ improvement)$$

The performance measure was the percentage of solved and unsolved puzzles. The formulas that described this behaviour were:

$$Percentage\ of\ solved\ puzzles\ =\ \frac{Number\ of\ solved\ puzzles}{Total\ sum\ of\ puzzles}\ \times\ 100$$

$$Percentage\ of\ unsolved\ puzzles\ =\ \frac{Number\ of\ unsolved\ puzzles}{Total\ sum\ of\ puzzles}\ \times\ 100$$

Other measurements used to solve the problems were heuristics. The heuristics used were the Manhattan Heuristic Function and the Attacking Pairs Heuristic which are described as follows:

The Manhattan Heuristic uses the Manhattan distance to estimate the cost to reach a goal from a given state (current state). Manhattan distance is the sum of the absolute difference of the coordinates. Where (x,y) are the coordinates that represent the position of a state relative to the goal.

$$d\ =\ |x1 - y1|\ +\ |x2 - y2|$$

For the Attacking Pairs Heuristic there are programs that can calculate it, since there are no definitive equations. A brief explanation on how it works is as follows:

1. Loop through each pair of queens.
2. Check if the pair share the same column.
3. Check if the pair are on the same diagonal.
4. Count and return the number of attacking pairs.

All of these performance metrics were used for the rest of the hill climbing variations and the simulated annealing.

The second algorithm that was studied was the First-Choice Hill Climbing Search. In this case the algorithm starts with an initial state as the current node. A random neighbor node is generated and evaluated to the current state. If that neighbor improves the solution objective of the problem the program moves to that neighbor node immediately. These steps are repeated

until a global maximum is reached, the program gets stuck in local maximum , or the program terminates by reaching an attempt limit. That final state is then returned.

The third algorithm that was studied was the Random Restart hill Climbing Search. In this case the algorithm starts with an initial state as the current node. The regular hill climbing algorithm is applied and if a solution is reached the search stops. If the search gets stuck on a local maximum the algorithm restarts on a new random state. These steps are repeated until a solution is found or the program is terminated by reaching an attempt limit.

The last algorithm that was studied was the Simulated Annealing. The initial state (current node) was set and the temperature was set to a high value. A random neighbor was generated and evaluated with the corresponding heuristic function. If the neighbor state fared better than the current state the program moved to that state. If it proved worse the program accepted it with some probability based on the temperature. The temperature was gradually decreased to minimize the chance of selecting a worse move. These steps were repeated until a solution was found or a limit was reached.

III. Performance Approach and Results:

A. Hill Climbing Algorithm with Steepest-Ascent

The Hill Climbing Algorithm is a heuristic search algorithm used in artificial intelligence that iteratively moves towards a better solution by evaluating neighboring states and selecting the one that offers the most improvement. It is a form of local search that focuses on finding the optimal solution by evaluating whether a new solution is better than the current one [2]. The process is similar to a person climbing a mountain in the sense that they would seek to improve their path to reach the top the fastest and most efficient way. Steepest-Ascent is a variant where instead of moving to the first neighboring node that improves a state, it evaluates all neighboring nodes and moves to the one with the highest improvement [2]. The way it works is as follows:

Step 1: Evaluates initial state.
Step 2: Makes the initial state the current state.
Step 3: Repeats until the solution is found or the current state doesn't change.
Step 4: Exits function if no better state is found.

While this algorithm appears robust, it has some notable limitations. Given it has a greedy nature, it is prone to getting stuck in areas in the search space where no neighboring state offers an improvement even though the solution has not been found. In the studied experiments where this algorithm was tested, it was observed that only a few instances resulted in a valid solution.

The first test for this algorithm was the 8 puzzle. It consists of a sliding puzzle displayed in a 3 by 3 grid with 8 squares and a space. The squares are labeled from 1 to 8 and the goal is to rearrange the tiles so that they are in row-major order using as few moves as possible. The permitted moves were: left, right, up, and down. To compare results a sample of 10 tests were done utilizing the steepest-ascent algorithm and the results showed that out of 10 tests none of them completed the goal as shown in Figure 1. Graph 1 compared the search cost vs the Manhattan distance, which represents an optimal solution to the puzzle. The results showed that the search cost value was significantly larger, but it still did not show a valid result. Graph 2 showed that the percentage of solved puzzles was 0% and the percentage of unsolved puzzles was 100%.The studied cases showed that the steepest ascent algorithm's greedy nature forces it to get stuck in a solution that deems as the most effective, given its lack of foresight, it chooses the wrong option. This does not mean that this algorithm could never solve the problem. In theory, it can choose the correct solution at some point but the chances are very slim.

The second test for this algorithm was the 8 queens puzzle. The queens puzzle consists of placing 8 chess queens on an 8 by 8 chessboard so that no two queens attack each other. The permitted moves were changing the position of the queens along their column , going up or down. To compare results a sample of 10 tests were done utilizing the steepest-ascent algorithm and the results showed that the steepest ascent algorithm is significantly better at solving this puzzle compared to the previous one, as shown in Figure 2. As shown in Graph 3 in multiple cases the search cost and the optimal cost had the same value. Graph 4 displayed that 70% of the cases were unsolved and 30% were solved. This significant increase shows that the steepest-ascent algorithm is more or less effective depending on the puzzle or problem to solve.

B. Hill Climbing Algorithm with First Choice

Making use of the original hill climbing algorithm in the search.py file [1], it was modified to establish the First Choice Hill Climbing Search Algorithm. The

First Choice Hill Climbing Algorithm is a stochastic search that consists of evaluating a random neighbor and only moving towards that neighbor if it improves the objective of the solution [3]. The way it works is as follows:

Step 1: Evaluates initial state.
Step 2: Make the initial state the current state.
Step 3: Compare the current state with a randomly chosen neighbor node.
Step 4: If the neighbor state improves the solution state, make the current node the neighbor node.
Step 5: Repeat until the solution is found,the current state doesn't change, reaching local maxima, or the search limit has been reached.
Step 6: Exit function once solution found, an attempt limit was reached, or local maxima is reached.

This type of hill climbing ultimately simplifies the neighbor selection, by not evaluating every neighbor thus reducing the computation time per iteration [4]. This however, does not exclude the chance that the program may get stuck in a shoulder, local maximum or a "flat" local maximum since it will evaluate that any of the other neighbors are not as good as the current selected neighbor [2]. The algorithm as well has the chance of getting stuck due to it not being able to backtrack as it only moves to the best neighbor node that "improves" the solution. In the studied experiments where this algorithm was tested, it was observed that only a few instances resulted in a valid solution.

As mentioned before the 8 puzzle is a sliding puzzle with a 3 by 3 grid with 8 labeled tiles. The objective is to be able to move each square to their corresponding goal position, while taking as few moves as possible. Ten randomly generated 8 puzzle problems were created to be able to analyze the behavior of the first-choice search algorithm. The results of implementation are presented in Figure 3 and Graphs 5-6, utilizing the Manhattan heuristic function to help solve the problems. That heuristic played the role of the most optimal cost. It is noted that none of the ten tests were able to solve the eight puzzle problem. The Graph 5 demonstrated that even with very large search costs the algorithm essentially got stuck and wasted a lot of moves until reaching a set limit, while compared to the Manhattan Distance. In the case of Graph 6 this showed that 0% of the 10 puzzles were solved resulting in a 100% of unsolved puzzles. This was expected due to the random factor of the code may lead to a neighbor node that is better than the rest of the node, but not the expected solution. This is the greedy nature of the algorithm at work, it climbs towards a better solution and continues until a local optimum is reached where no further improvements are possible [4].

This is not to say that the First-Choice search was never going to be able to solve the problem, but due to the random choice of puzzle configurations and the algorithm's ability to get stuck, the chances of it solving the problem were slim.

For the second problem, the algorithm to solve was the 8 queens problem. The problem consisted of placing 8 queens in a chess board in such a way that no queen is directly or indirectly attacking each other. Once again, ten randomly generated queen problems were made. The solutions of these tests were shown in Figure 4 and Graphs 7-8, the attacking queen pairs were used as the heuristic function to help solve the problems. The First-Choice Hill Climbing search fared better in the 8 queen problem than the 8 puzzle problem. For this experiment only two problems were solved leading to 20% of the cases being solved and 80% being unsolved as shown in Graph 8. In Graph 7 most of the search costs were similar, but the optimal solution costs was on average 3. It is important to note that the optimal path costs for this experiment was the minimum number of moves required to reach the goal state from the initial state. Compared to the eight puzzle problem where the optimal solution costs would vary between different puzzles. The search's ability to be able to solve some of the problems shows it is a non complete algorithm whose effectiveness depends on the structure of the problem.

C. Hill Climbing Algorithm with Random Restart

By using the hill climbing algorithm already mentioned above. The implementation of this algorithm was changed to be used with random restarts [2] to solve the problem of the local maximums. To have the best opportunity at finding the optimal solution in both 8 puzzle problem and the 8 queens variation of the N queens problem. The Random Restart variant of the hill climbing algorithm consists of giving the problem under evaluation a new random starting position when the algorithm gets stuck in a local maxima and giving it a better opportunity at finding a global maxima[2]. The implementation of this algorithm takes in part the already existing algorithm of hill climbing and adds the random restart on another scratch file where the hill climbing algorithm is called. The way it works is as follows:

Step 1: Evaluates initial state.
Step 2: Make the initial state the current state.

Step 3: Repeat until the solution is found or the current state doesn't change, making local maxima.
Step 4: Gives a new random starting state if no solution was found or stuck on local maxima up to 10 times.
Step 5: Exit function once solution found or number of maximum restarts is reached.

While this algorithm appears to solve some of the limitations of the previous algorithms, it does not solve the hill climbing algorithm limitations. Such as being prone to getting stuck in areas in the search space where no neighboring state offers improvement on the current state, even though the solution has not been found. In the studied experiments where this algorithm was tested, it was observed that in some instances depending on the problem resulted in a valid solution or no solution. The valid solutions appeared in the 8 queens problem where, as mentioned before, 8 queens are put on a 8 by 8 chessboard where the objective is for none of the queens to be attacking each other. On average it solved 90% of the puzzles given to it when having a limit of 10 restarts and are shown in Figure 6 and Graph 11 and 12. When the limit of restarts is decreased the performance decreases with 5 restarts giving an average of 60% solved to unsolved ratio. Increasing this limit to 20 resulted in an average of all problems being solved.

As mentioned before the 8 puzzle is a sliding puzzle with a 3 by 3 grid with 8 label tiles. The point is to be able to move each square to their corresponding goal position, while taking as few moves as possible. Ten randomly generated 8 puzzle problems were created to be able to analyze the behavior of the random restart algorithm. The results of implementation are presented in Figure 5 and Graphs 9 and 10. Where the 8 puzzle was not solved in any instance as the hill climbing algorithm often got stuck in local maxima and it required bad moves to arrive at the optimal solution. Making the hill climbing algorithm unsuited for solving this particular problem.

D. Simulated Annealing

Simulated Annealing is a robust optimization technique that mimics the physical process of annealing to find optimal or near-optimal solutions in large and complex search spaces. In Simulated Annealing, the "heat" corresponds to the degree of randomness in the search process, which decreases over time (cooling schedule) to refine the solution [5]. This algorithm was used to solve the 8-puzzle

problem and the 8-queens problem. In the search.py file, two implementations of the algorithm were created, the first filling an empty board with queens, the other having an initial state of queens already on the board where of the implementations each was adapted for one of the problems [1].

The way this algorithm works is as follows [5]:

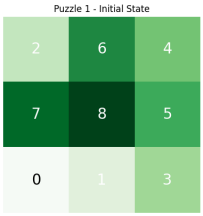
Step 1: Start with an initial solution and an initial temperature.
Step 2: Generate new solutions based on the current temperature, allowing worse solutions to be accepted with some probability to explore the search space.
Step 3: Evaluate the new solution using the objective function. If it is better than the current solution, it is accepted as the new main solution. A worse solution may also be accepted depending on the temperature.
Step 4: Decrease the temperature after each iteration according to a predefined cooling schedule.
Step 5: The algorithm terminates when the temperature reaches a low value or a predetermined number of iterations is reached.

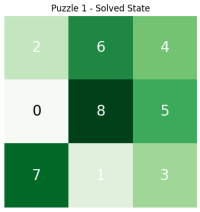
For the 8-puzzle problem, the `simulated_annealing` implementation from the `search.py` file was used [1]. Some modifications were made so that the algorithm returns the current node instead of just the state of the node. The goal of this problem is to arrange the numbered tiles in a specific order. The heuristic used for this problem was the Manhattan distance heuristic, as it is the most optimal for this type of problem. After running 10 test instances, the algorithm failed to find a single solution, resulting in a 0% success rate, as shown in Graph 14. Additionally, Figure 7 and Graph 13 illustrates that the puzzles were not even close to being solved. The nature of this algorithm makes it neither optimal nor effective for this type of problem. Since Simulated Annealing sometimes accepts worse solutions to explore the search space, it struggles with problems like the 8-puzzle, which has a constrained space and limited possible movements. As a result, the algorithm often gets stuck in a local optimum (maximum), repeating the same movements without making progress until the temperature decreases low enough to terminate the search.

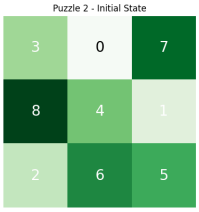
For the 8-queens problem, the `simulated_annealing_full` implementation from the `search.py` file was used [1]. Just like with the 8-puzzle problem, this implementation of the algorithm was modified to use the heuristic implemented in the 8-queens file. The heuristic used was the number of attacks heuristic. The `simulated_annealing_full` algorithm was chosen instead of the standard `simulated_annealing` used in the 8-puzzle problem because this version allows

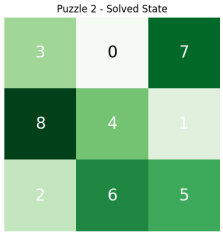
for counting the states visited (search cost), which is essential for analyzing the search performance. The simulated annealing algorithm was tested with two different variations of the 8-queens problem. The first variation is the one provided by the `search.ipynb` file [1]. In this variation, there is no initial state, and the algorithm starts by filling the board with queens that cannot attack each other, from left to right. If there is a queen missing from a column, it indicates that the algorithm couldn't find a solution before terminating. From 10 instances tested, only 2 found a solution, as shown in Figure 8 and Graph 15, resulting in a 20% success rate, as shown in Graph 16. The second variation as seen in Figure 9 is a modified version of the first one that includes an initial state. The algorithm is tested differently, although the goal remains the same, as it will go through every queen to try and find a solution for the whole board. Another 10 instances were tested, and none of them found a solution as seen in Graph 17, resulting in a 0% success rate, as shown in Graph 18. Both variations encountered issues with this problem. Simulated Annealing is not optimal for the 8-queens problem because, much like in the 8-puzzle, it is prone to getting stuck in local optima. Since the algorithm sometimes accepts worse solutions to explore the search space, it struggles to efficiently find a solution in problems with many potential configurations like the 8-queens problem. The limited movement options in this context, coupled with the randomness of the algorithm, make it difficult to consistently reach the goal.

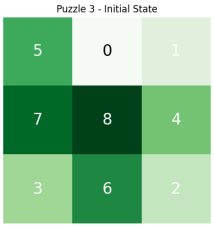
E. Figures and Tables

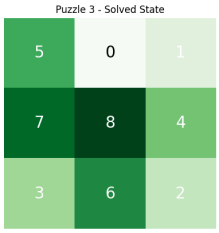


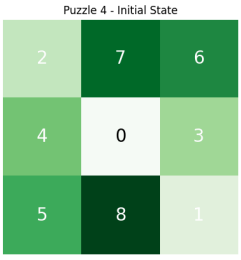


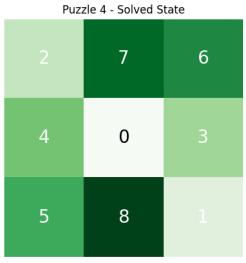


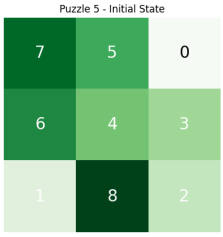

Puzzle 2 - Solved State
3 0 7
8 4 1
2 6 5

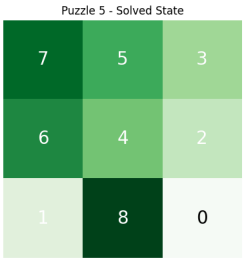

Puzzle 3 - Initial State
5 0 1
7 8 4
3 6 2

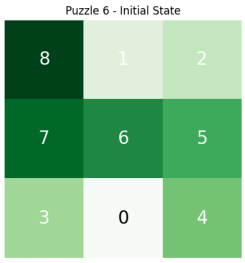

Puzzle 3 - Solved State
5 0 1
7 8 4
3 6 2

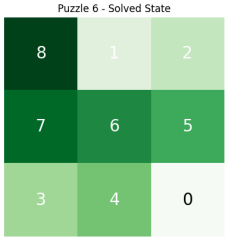

Puzzle 4 - Initial State
2 7 6
4 0 3
5 8 1

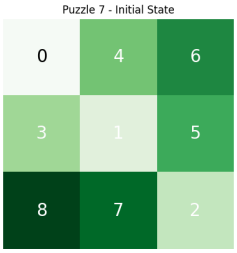

Puzzle 4 - Solved State
2 7 6
4 0 3
5 8 1

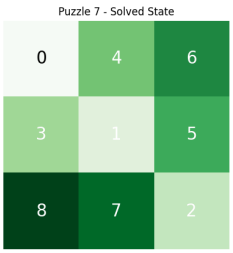

Puzzle 5 - Initial State
7 5 0
6 4 3
1 8 2

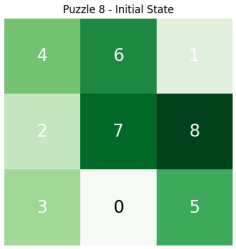

Puzzle 5 - Solved State
7 5 3
6 4 2
1 8 0


Puzzle 6 - Initial State
8 1 2
7 6 5
3 0 4


Puzzle 6 - Solved State
8 1 2
7 6 5
3 4 0


Puzzle 7 - Initial State
0 4 6
3 1 5
8 7 2


Puzzle 7 - Solved State
0 4 6
3 1 5
8 7 2


Puzzle 8 - Initial State
4 6 1
2 7 8
3 0 5

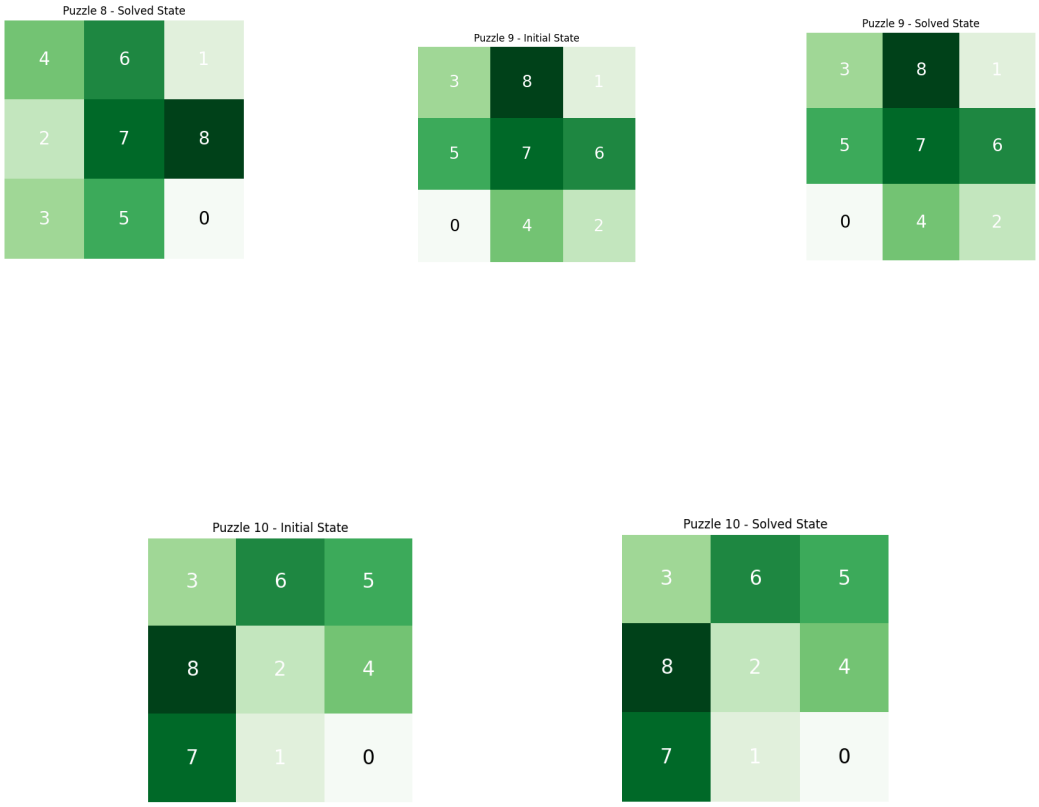
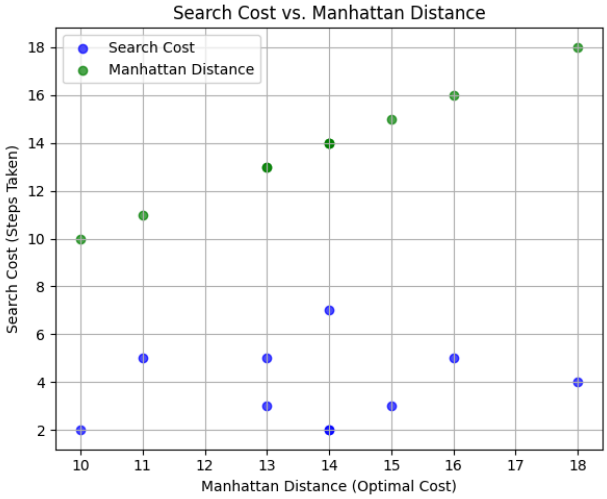
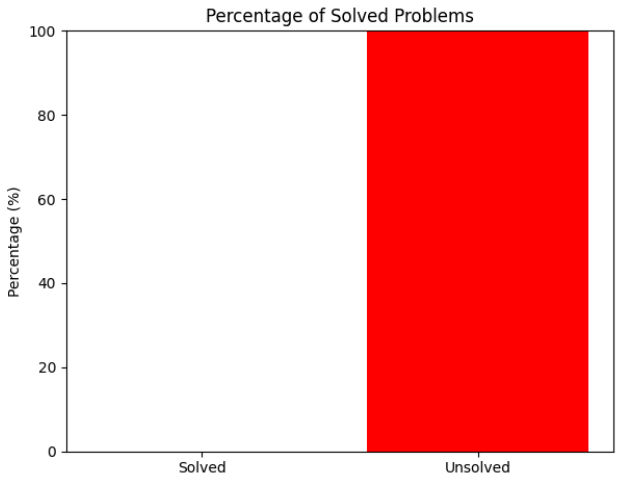


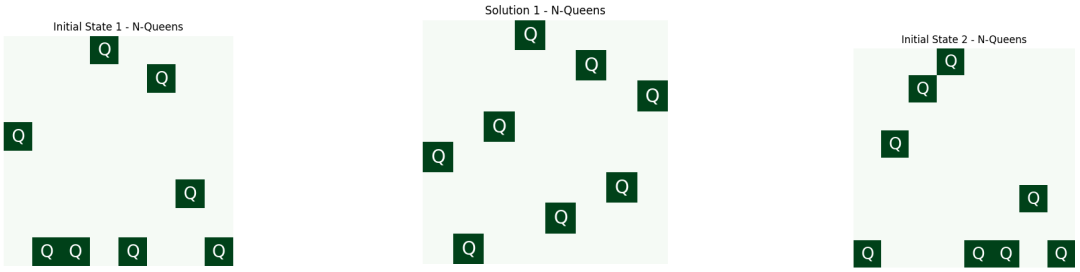
Figure 1. Results for 8 puzzle utilizing Steepest Ascent Hill Climbing



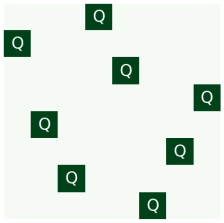
Graph 1. Search Cost vs Manhattan Distance (Optimal Cost) of Steepest Ascent Hill Climbing in 8 puzzle



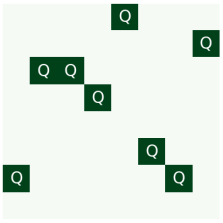
Graph 2. Percentage of solved and unsolved puzzles utilizing the Steepest Ascent Hill Climbing for the 8 puzzle



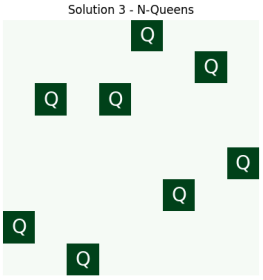
Solution 2 - N-Queens



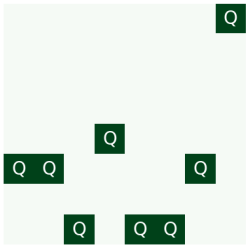
Initial State 3 - N-Queens



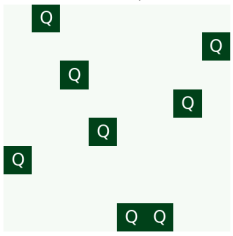
Solution 3 - N-Queens



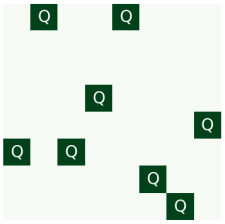
Initial State 4 - N-Queens



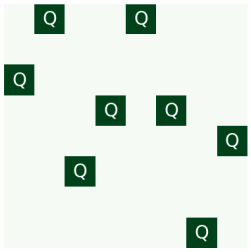
Solution 4 - N-Queens



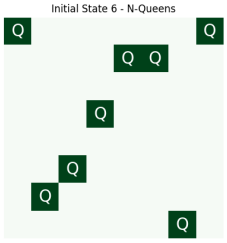
Initial State 5 - N-Queens



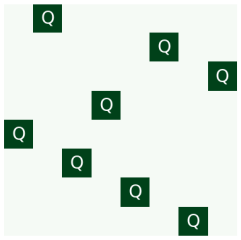
Solution 5 - N-Queens



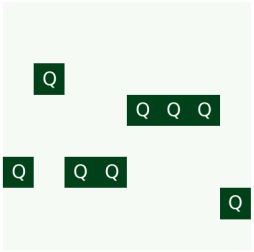
Initial State 6 - N-Queens



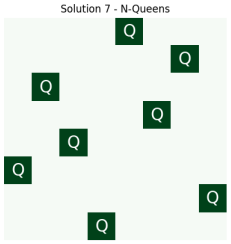
Solution 6 - N-Queens



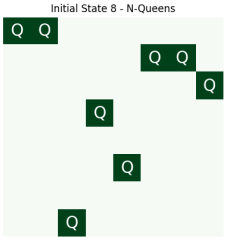
Initial State 7 - N-Queens



Solution 7 - N-Queens



Initial State 8 - N-Queens



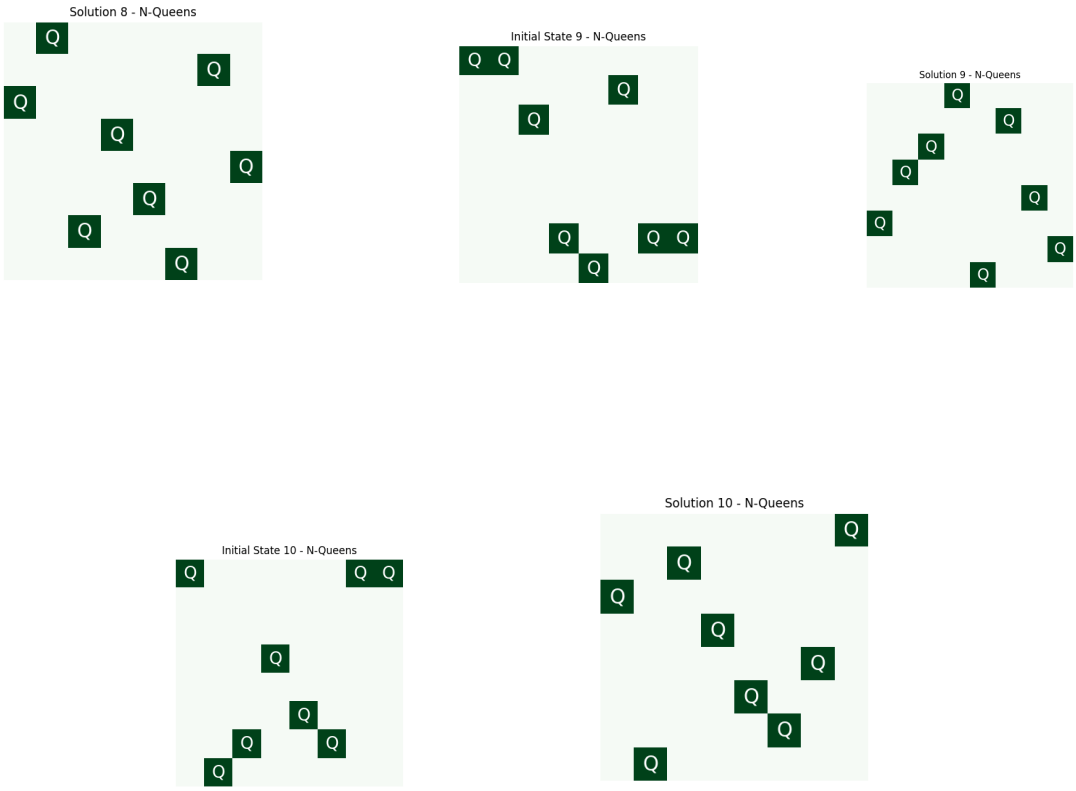
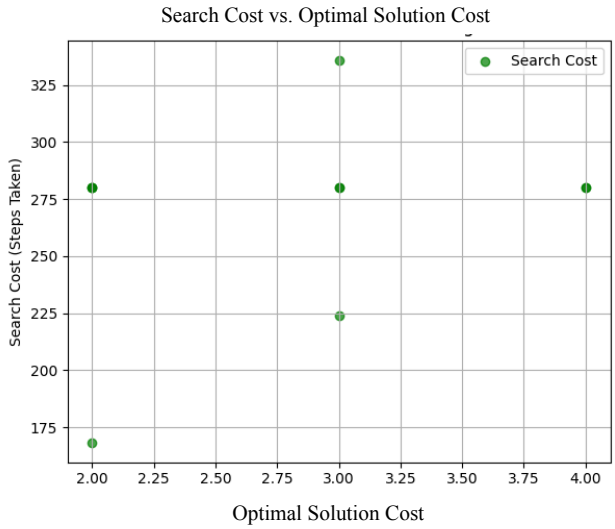
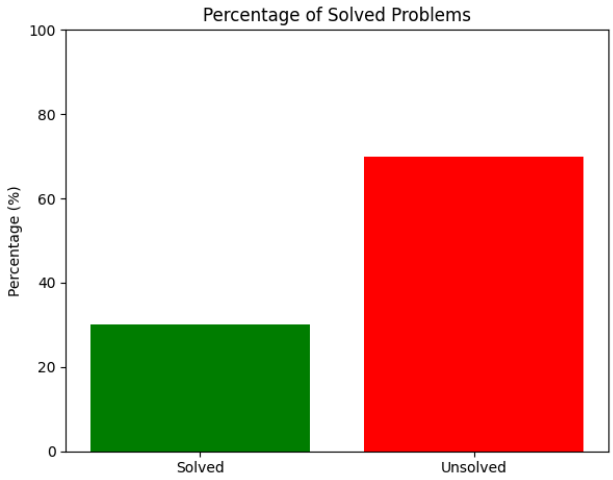


Figure 2. Results for 8 queen puzzle utilizing Steepest Ascent Hill Climbing



Graph 3. Search Cost vs Manhattan Distance (Optimal Cost) of Steepest Ascent Hill Climbing in 8 queen problem



Graph 4. Percentage of solved and unsolved puzzles utilizing the Steepest Ascent Hill Climbing for the 8 queen problem

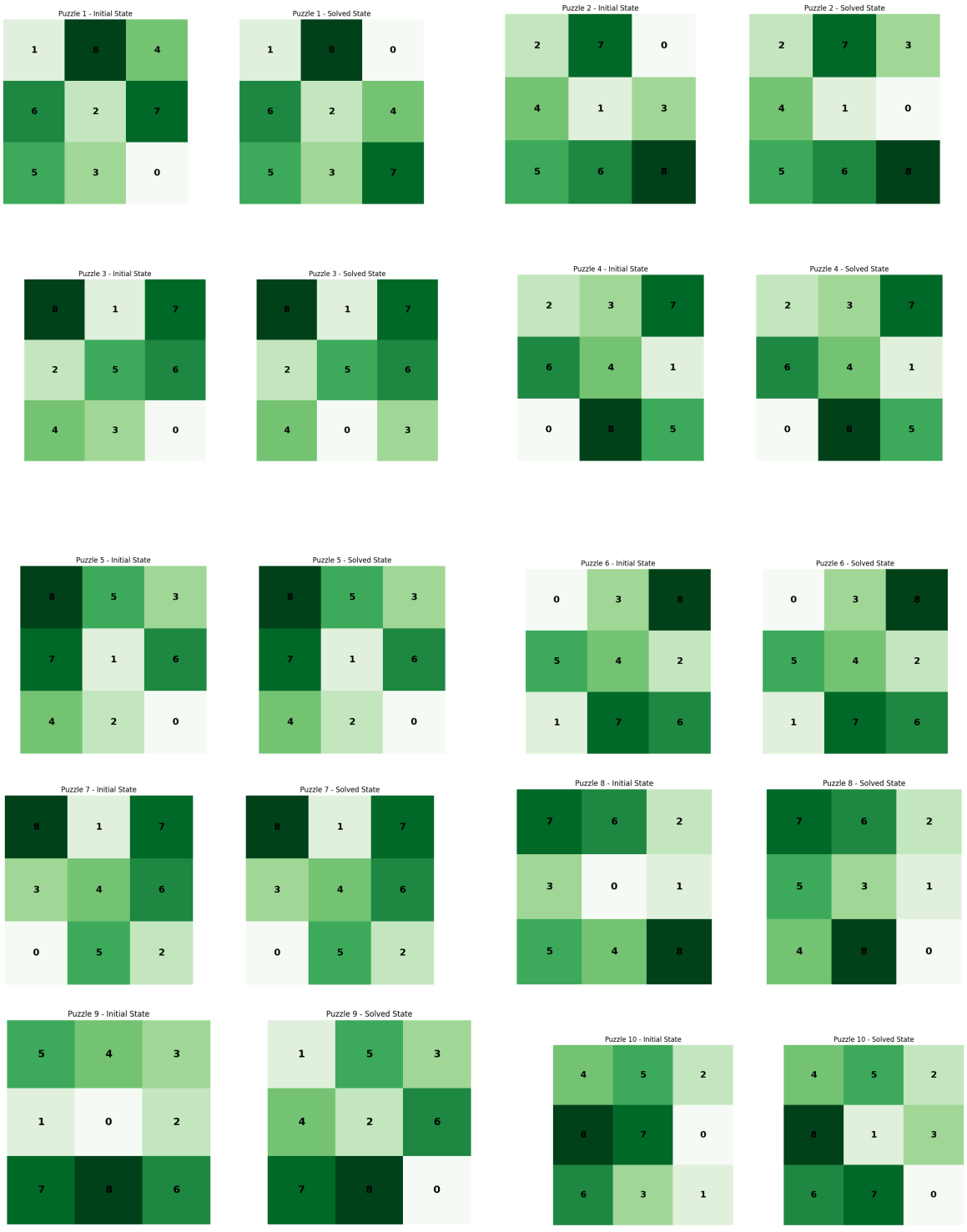
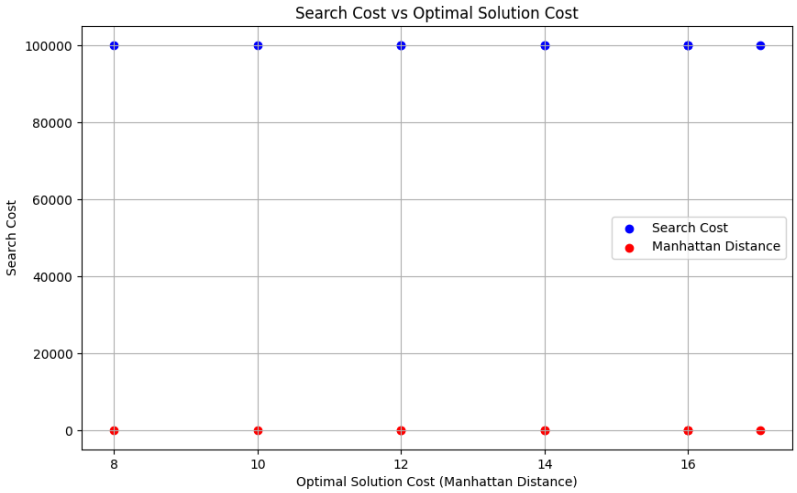
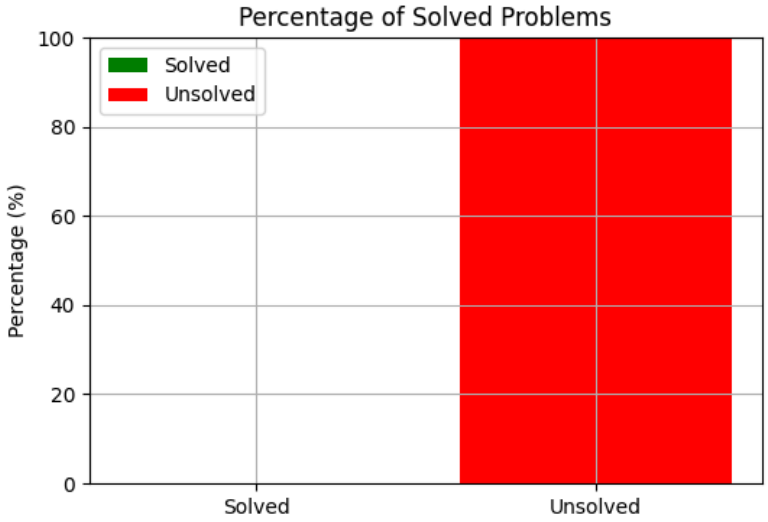


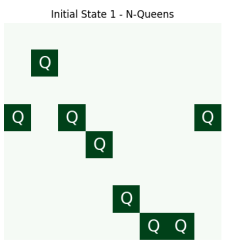
Figure 3. Results for 8 puzzle utilizing First Choice Hill Climbing

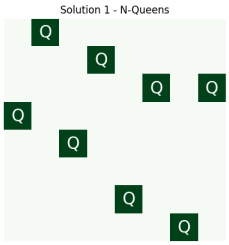


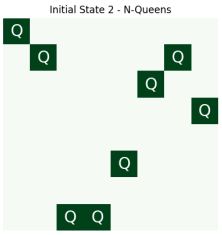
Graph 5. Search Cost vs Manhattan Distance (Optimal Solution Cost) of First Choice Hill Climbing in 8 puzzle

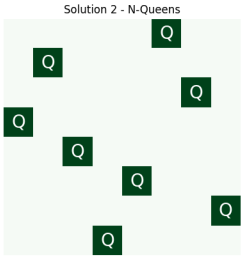


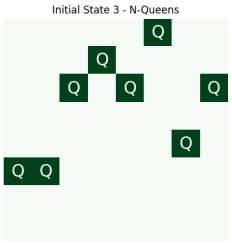
Graph 6. Percentage of solved and unsolved puzzles utilizing the First Choice Hill Climbing for the 8 puzzle

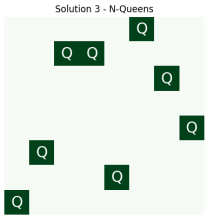

Initial State 1 - N-Queens

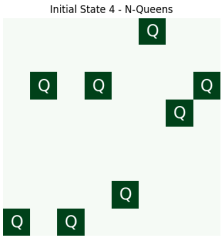

Solution 1 - N-Queens

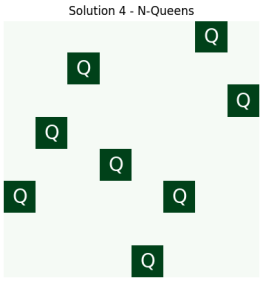

Initial State 2 - N-Queens

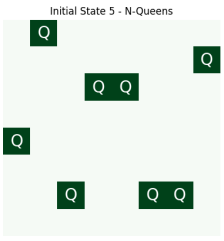

Solution 2 - N-Queens

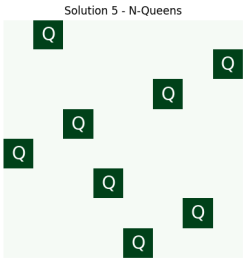

Initial State 3 - N-Queens

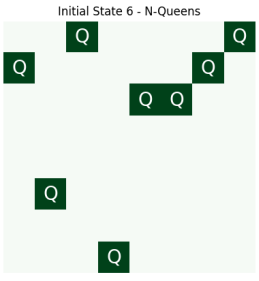

Solution 3 - N-Queens

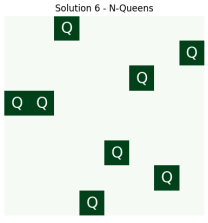

Initial State 4 - N-Queens

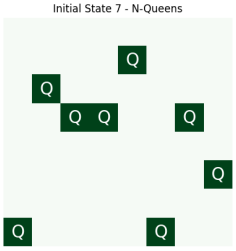

Solution 4 - N-Queens

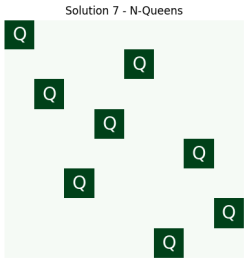

Initial State 5 - N-Queens

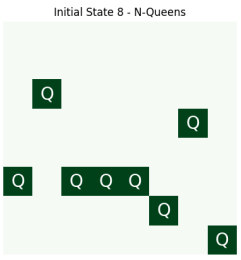

Solution 5 - N-Queens


Initial State 6 - N-Queens


Solution 6 - N-Queens


Initial State 7 - N-Queens


Solution 7 - N-Queens


Initial State 8 - N-Queens

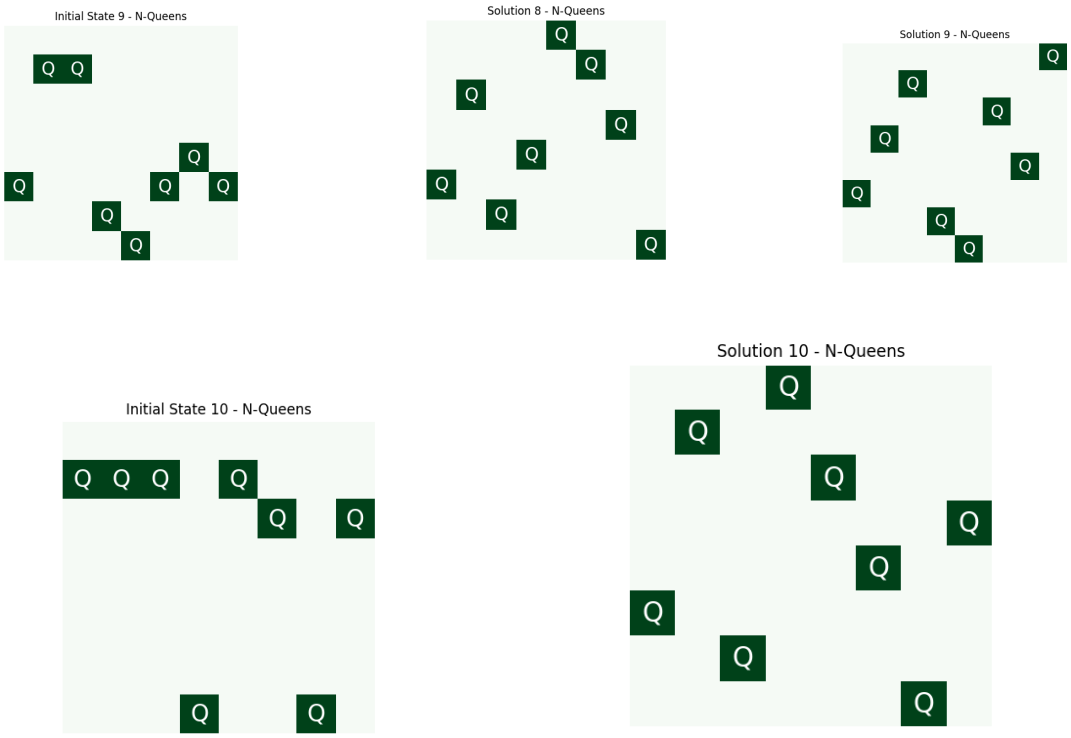
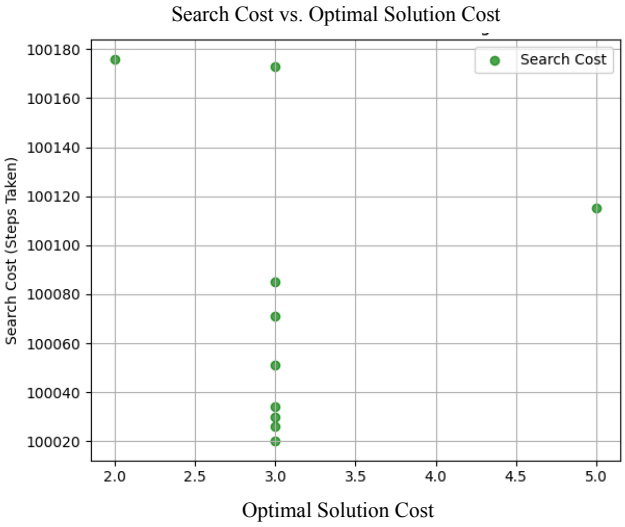
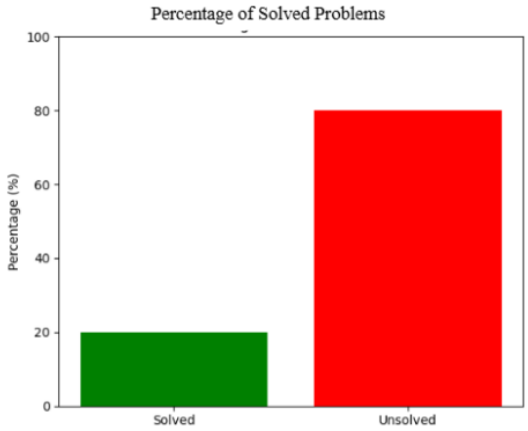


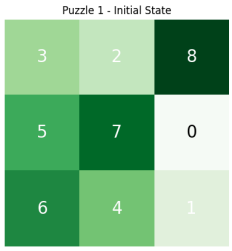
Figure 4. Results for 8 queen puzzle utilizing First-Choice Hill Climbing

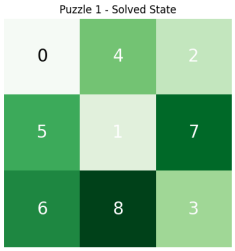


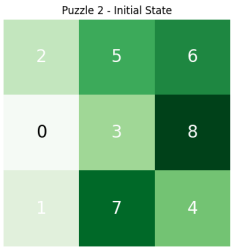
Graph 7. Search Cost vs Optimal Solution Cost of First Choice Hill Climbing in 8 queen problem

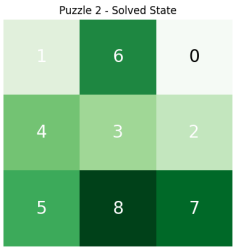


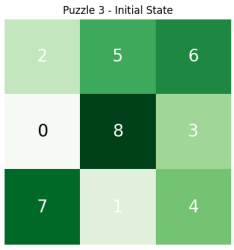
Graph 8. Percentage of solved and unsolved puzzles utilizing the First Choice Hill Climbing for the 8 queen problem

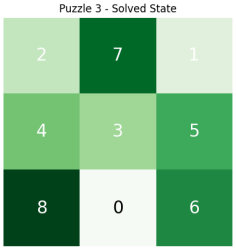


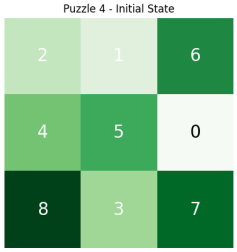


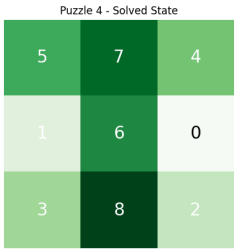












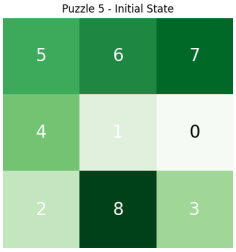
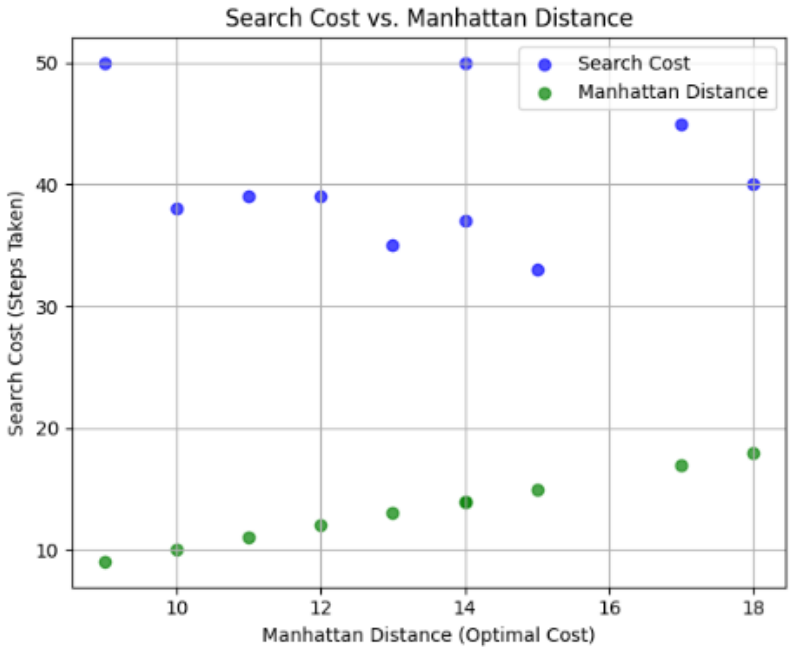
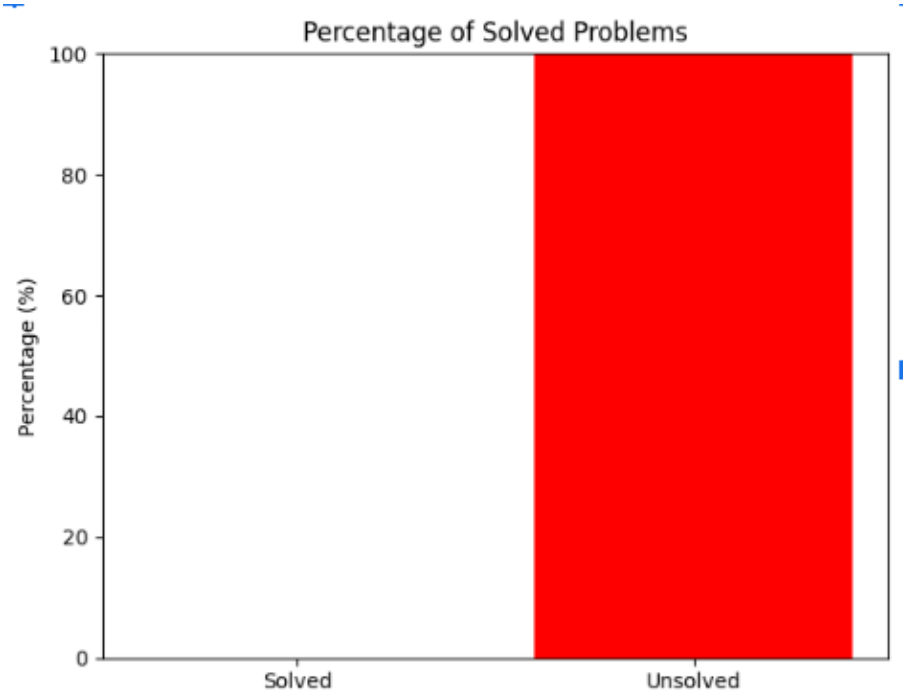


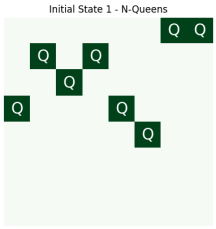
Figure 5. Results for 8 puzzle utilizing Random restart Hill Climbing

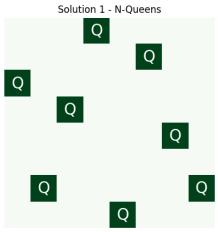


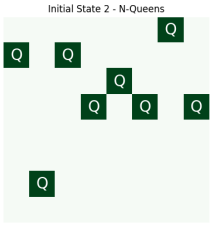
Graph 9. Search Cost vs Manhattan Distance (Optimal Solution Cost) of Random Restart Hill Climbing in 8 puzzle

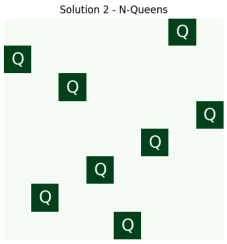


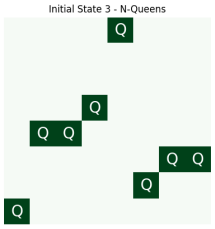
Graph 10. Percentage of solved and unsolved puzzles utilizing the Random Restart Hill Climbing for the 8 puzzle

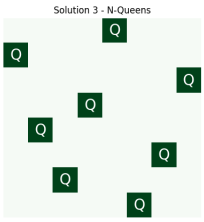

Initial State 1 - N-Queens

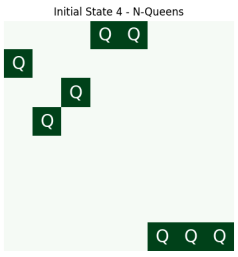

Solution 1 - N-Queens

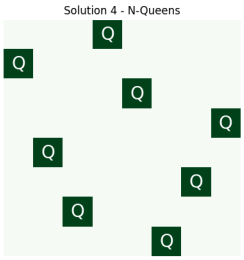

Initial State 2 - N-Queens

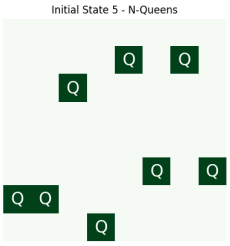

Solution 2 - N-Queens

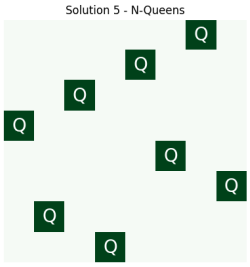

Initial State 3 - N-Queens

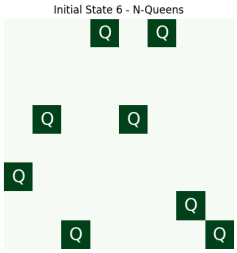

Solution 3 - N-Queens

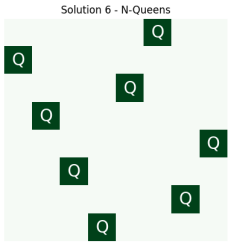

Initial State 4 - N-Queens


Solution 4 - N-Queens


Initial State 5 - N-Queens


Solution 5 - N-Queens


Initial State 6 - N-Queens


Solution 6 - N-Queens

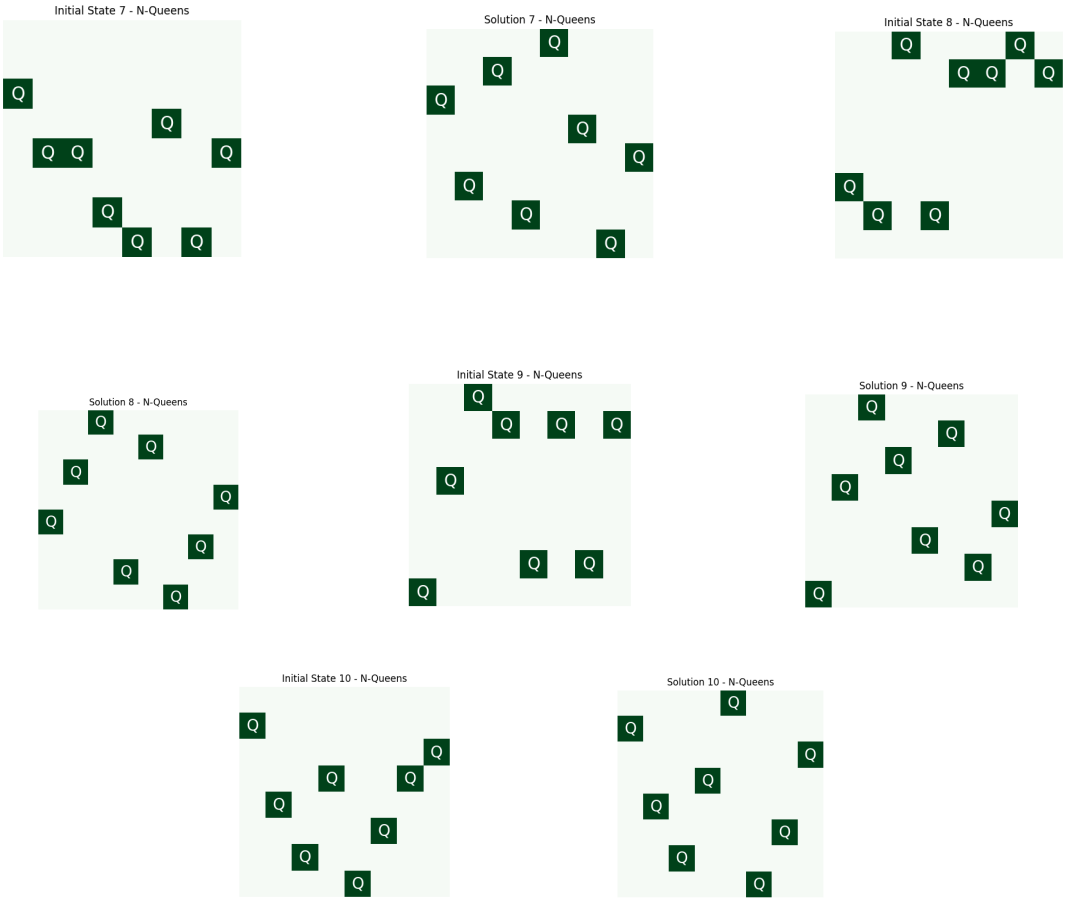
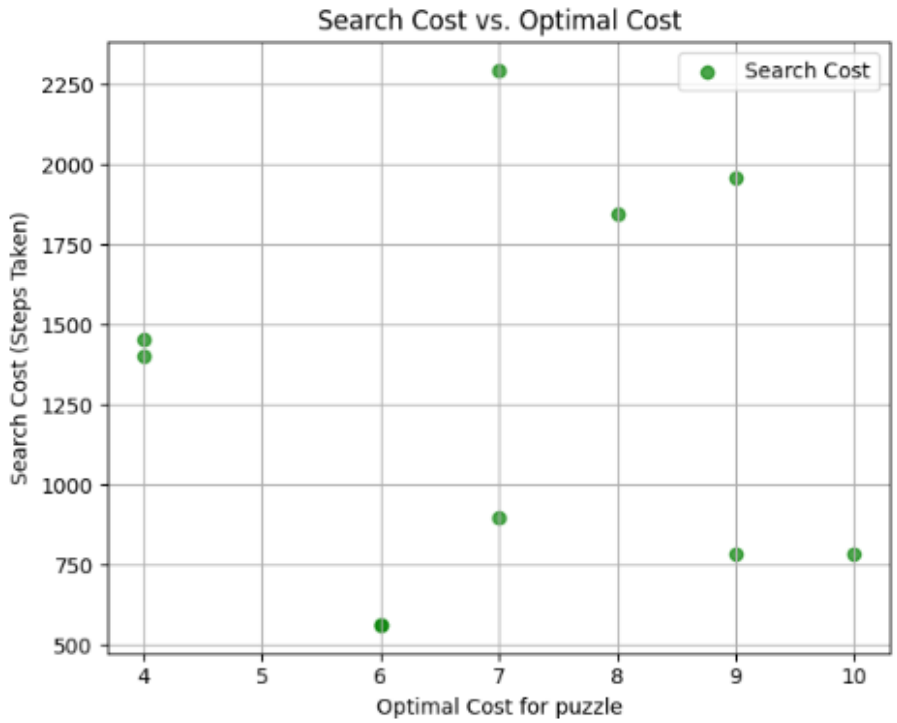
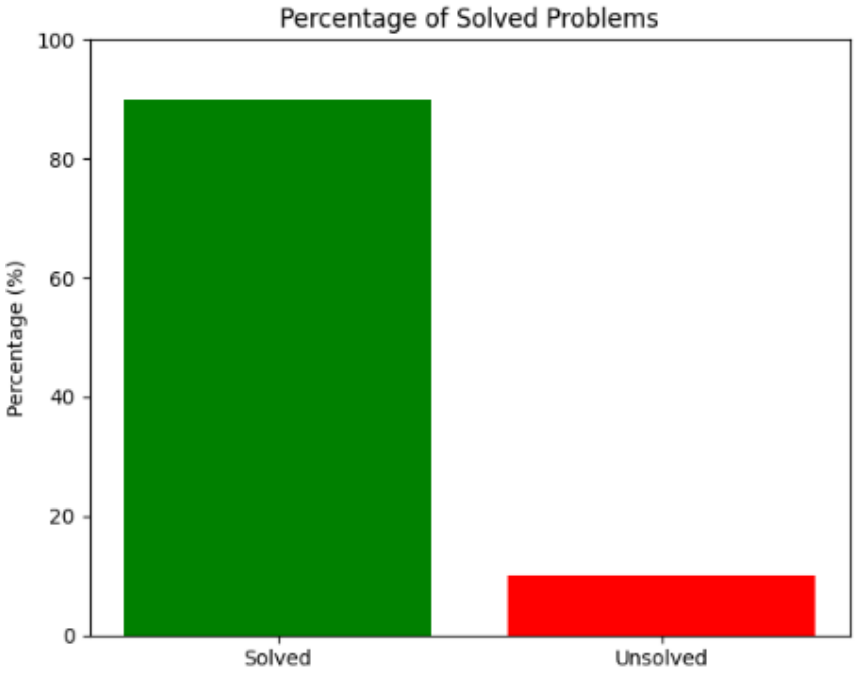


Figure 6: Results for 8 queen puzzle utilizing Random Restart Hill Climbing



Graph 11. Search Cost vs Optimal Solution Cost of Random Restart Hill Climbing in 8 queen problem



Graph 12. Percentage of solved and unsolved puzzles utilizing the Random Restart Hill Climbing for the 8 queen problem

Puzzle 1-Initial State

2	3	6
1	5	8
4	7	

Puzzle 1-Solution State

4	7	2
5	1	8
6		3

Puzzle 2-Initial State

1	5	2
4	3	6
	7	8

Puzzle 2- Solution State

3		8
5	1	2
4	7	6

Puzzle 3-Initial State

2		3
1	4	6
7	5	8

Puzzle 3-Solution State

5	6	3
2	1	
4	8	7

Puzzle 4-Initial State

1	3	6
	7	2
5	4	8

Puzzle 4- Solution State

6	7	3
1	5	
4	8	2

Puzzle 5-Initial State

4	5	
7	6	1
3	2	8

Puzzle 5-Solution State

2	4	5
7	6	
3	8	1

Puzzle 6-Initial State

2	5	3
1	7	6
	4	8

Puzzle 6- Solution State

1	3	5
2	7	6
	4	8

Puzzle 7-Initial State

4	1	2
5	3	6
7		8

Puzzle 7-Solution State

3	4	6
2		5
7	1	8

Puzzle 8-Initial State

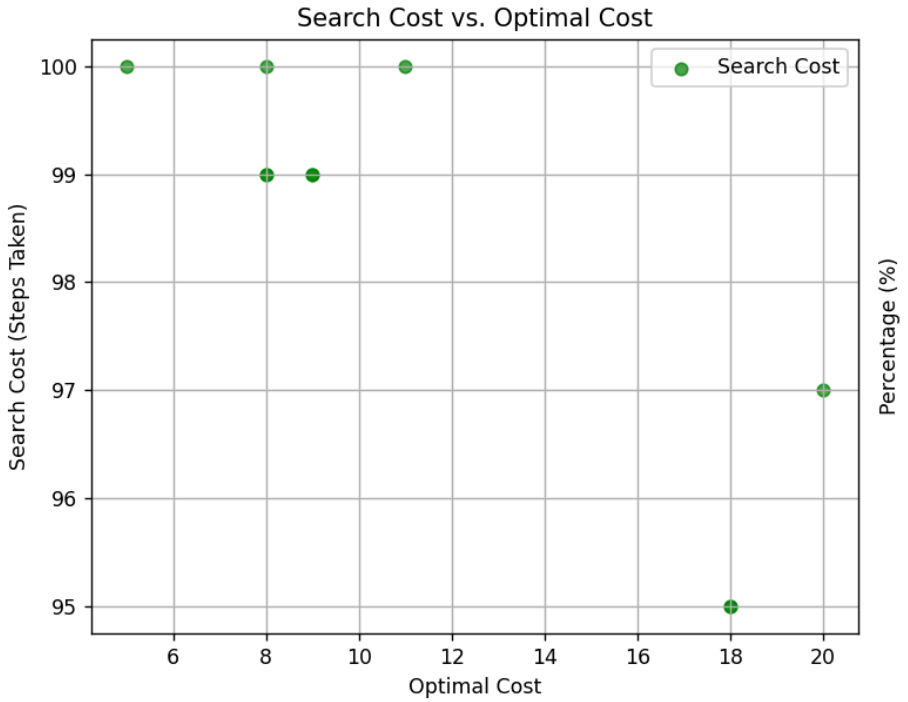
3	4	2
1	7	6
5	8	

Puzzle 8- Solution State

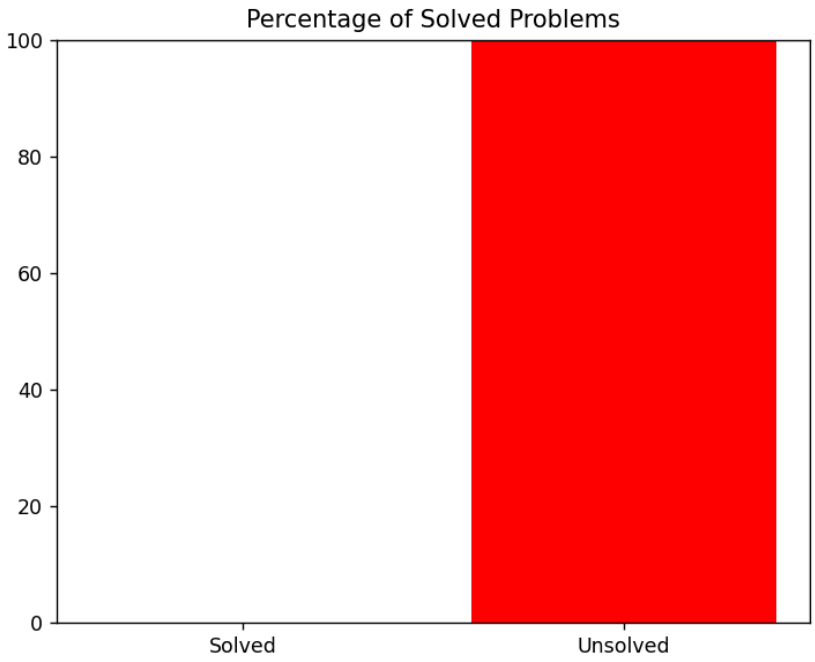
7		2
4	3	5
1	8	6

Puzzle 9-Initial State			Puzzle 9-Solution State			Puzzle 10-Initial State			Puzzle 10- Solution State		
5	1	3	6	3	4		8	3	7	2	5
4	2	6	2	7	1	2	5	6	1	3	8
7		8	5	8		7	1	4	4		6

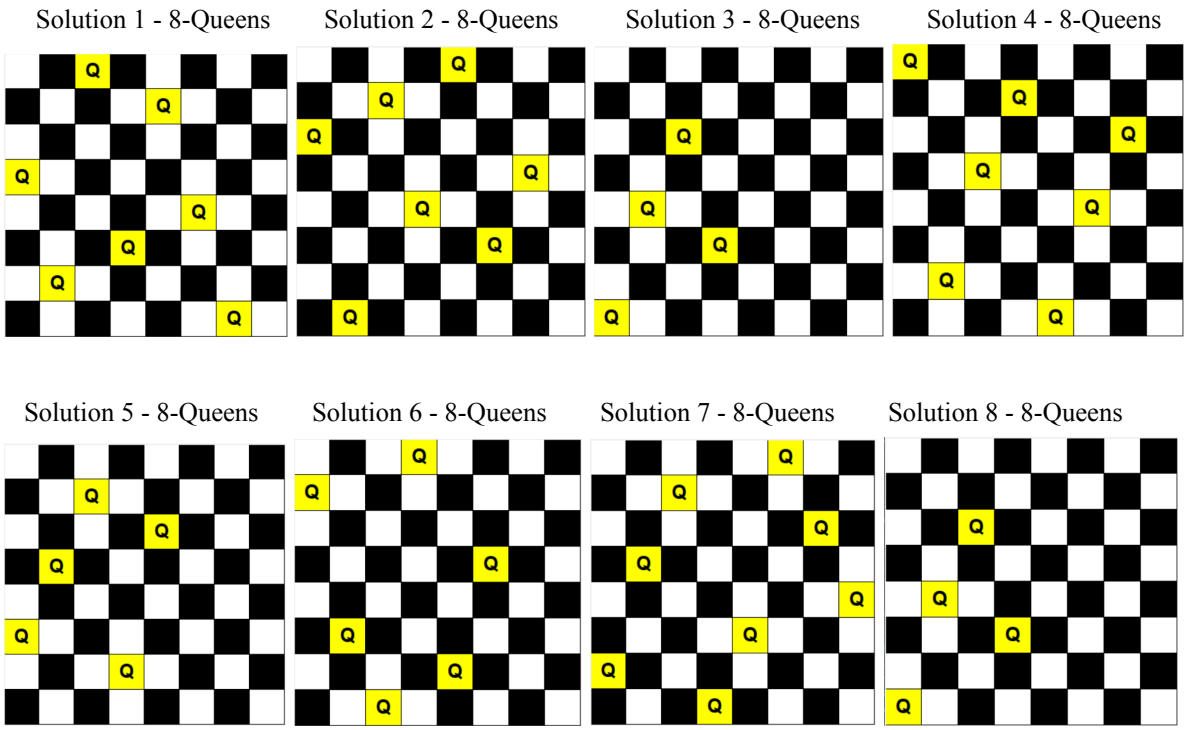
Figure 7. Results for 8-puzzle utilizing Simulated Annealing



Graph 13. Search Cost vs Optimal Solution Cost of Simulated Annealing in 8-puzzle



Graph 14. Percentage of solved and unsolved puzzles utilizing Simulated Annealing for 8-puzzle



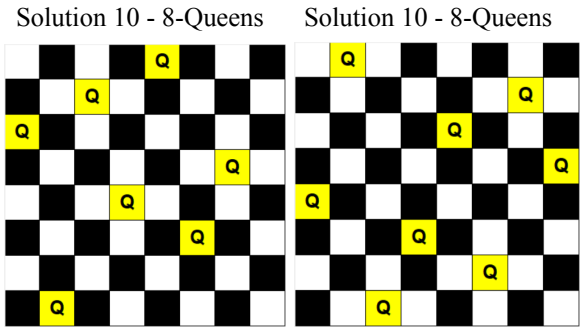
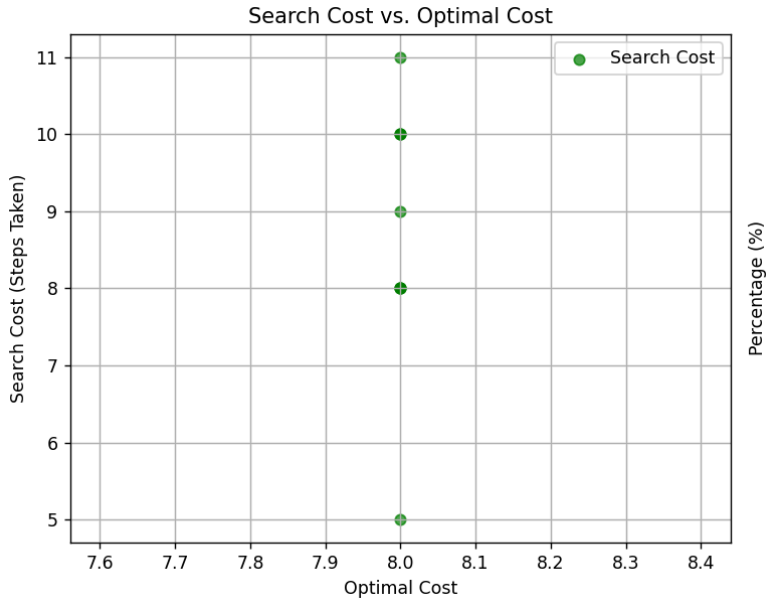
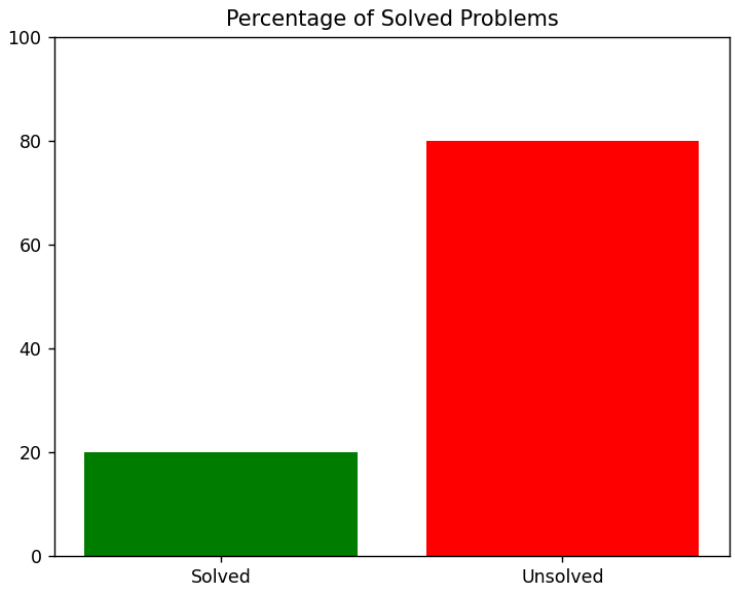


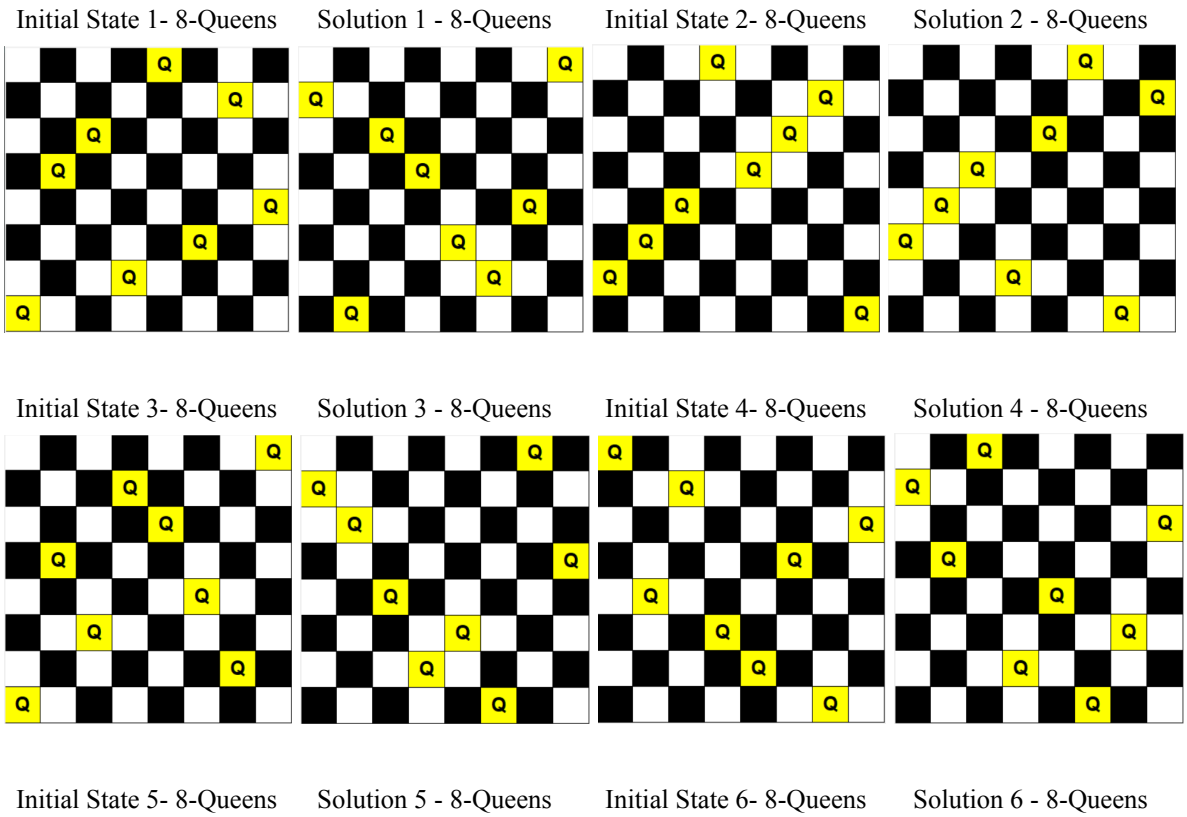
Figure 8: Results for 8-queen puzzle utilizing Simulated Annealing without an initial state

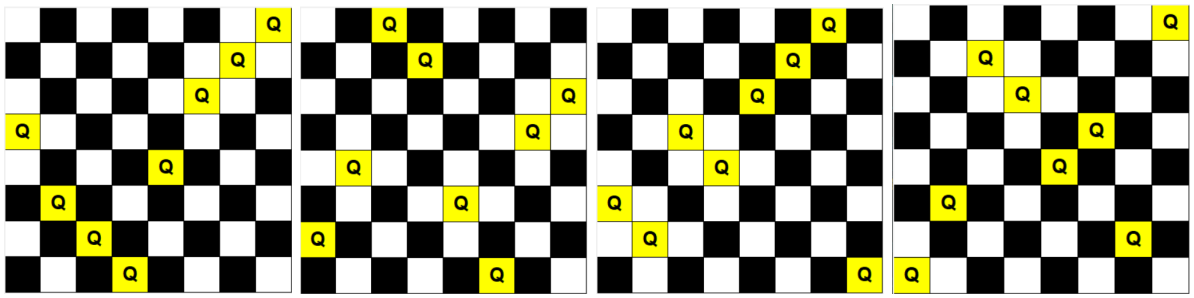


Graph 15. Search Cost vs Optimal Solution Cost of Simulated Annealing in 8-queen problem (Without initial state)

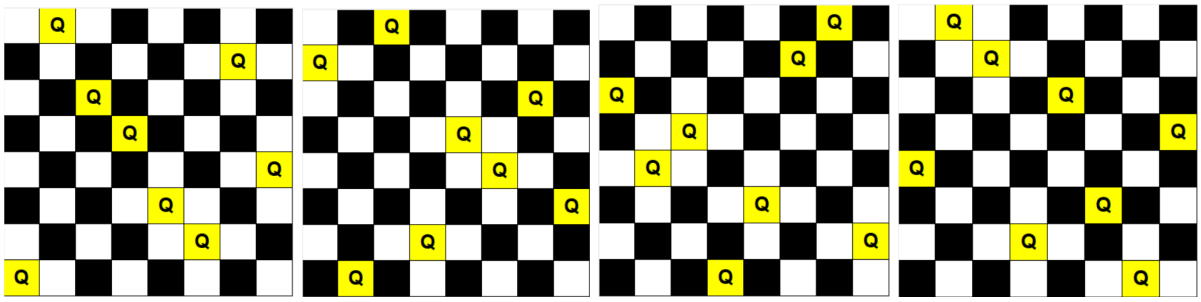


Graph 16. Percentage of solved and unsolved puzzles utilizing Simulated Annealing for 8-queen problem (Without initial state)





Initial State 7- 8-Queens Solution 7 - 8-Queens Initial State 8- 8-Queens Solution 8 - 8-Queens



Initial State 9- 8-Queens Solution 9 - 8-Queens Initial State 10- 8-Queens Solution 10 - 8-Queens

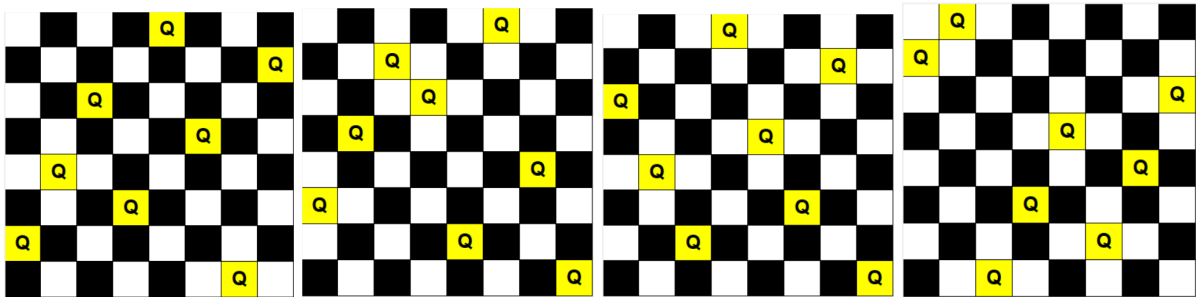
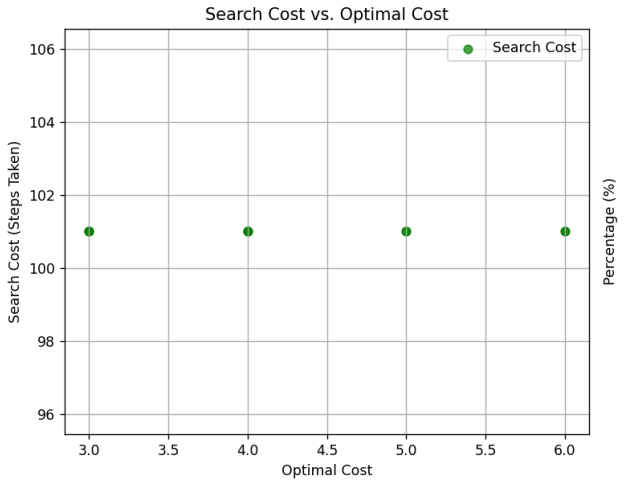
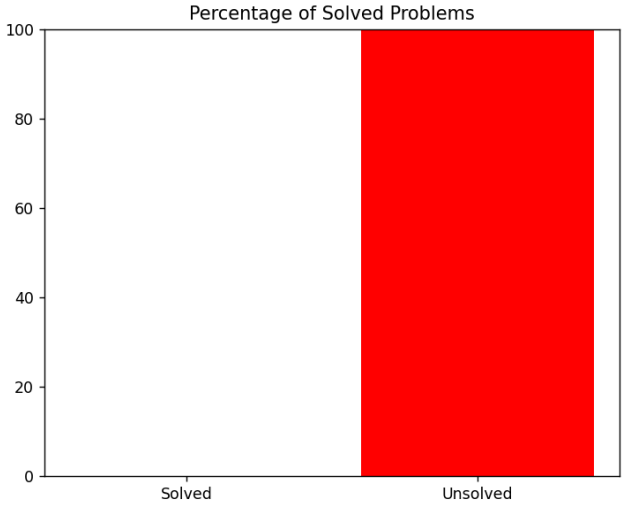


Figure 9: Results for 8-queen puzzle utilizing Simulated Annealing with an initial state



Graph 17. Search Cost vs Optimal Solution Cost of Simulated Annealing in 8-queen problem (With initial state)



Graph 18. Percentage of solved and unsolved puzzles utilizing Simulated Annealing for 8-queen problem (With initial state)

IV. Conclusion:

The hill climbing algorithm is a local search algorithm that attempts to find the correct solution by making an incremental change to the potential solution. It is widely used in artificial intelligence, but despite its robustness it has noticeable shortcomings. Searching for any move that improves the current state is expensive and inefficient, making it particularly susceptible to

getting stuck in local optima. To address these challenges, various modifications of the hill climbing algorithm have been developed. The following conclusion will examine these variations and evaluate their effectiveness in solving the 8 puzzle and 8 queens problems.

Steepest-ascent hill climbing is a variant of hill climbing that always moves towards the state that improves the current state. By evaluating all possible moves it selects one that significantly improves the heuristic value. However, the tests for 8 puzzle showed that steepest ascent is not the most efficient algorithm to solve disjointed and complex problems. Heuristic values do not always guide the algorithm to the correct solution. Moving to a different state might improve the state temporarily but in the process there is a high chance of getting stuck and never reaching a solution. In contrast, tests on the 8 queens puzzle showed significantly better improvement, being able to successfully solve multiple instances. This is largely due to the fact that the state space of the 8 queens problem is smooth and structured in a way that allows consistent improvement of the state. Unlike the 8 puzzle where the algorithm must compare states against a fixed goal, the 8 queens problem benefits from a flexible evaluation method, minimizing queen attacks. Additionally dead ends were relatively easy to escape given a queen can move freely through reducing the risk of stagnation. Steepest-ascent hill climbing is ineffective for problems that require backtracking or temporary moves that worsen the current state, a necessity in solving the 8 puzzle.

The First-Choice hill climbing is another variant of the hill climbing search that picks a random neighbor node and evaluates if it is better than the current state. If it is, that is the successor that the search moves to and the process is repeated until no better successor is found. This type of search allows for faster execution due to it not requiring to search through all the neighbor (successor) nodes. However, this does not entail that the search is complete. This variant is still subject to being caught in an infinite loop by getting stuck in a shoulder, local maximum, or a “flat” local maximum. To analyze the performance of this algorithm two problems were used, the 8 puzzle problem and the 8 queens problem. For both problems, 10 randomly generated problems were created and tested. In the 8 puzzle problem the search cost of each choice was compared to the Manhattan heuristic function. This heuristic compared the distance from the current position to the goal. Even with this function, it did not always guide the search to the correct solution. For the 8 queens problem, however, the heuristic used was the number of attacking queen pairs. We wanted to lower this value, thus since the first-choice was made using a more “descent” the search proved more efficient at solving this problem with a 20% solvability. Still, a very low value since the search would often get stuck in an infinite loop. This in part as mentioned beforehand had to do with the flexibility of the eight queens problem structure compared to the eight puzzle structure. First-Choice hill climbing is an inefficient search algorithm for problems that have similar heuristic values, difficult to find an improving move, getting stuck in local minima, and lacking irreversible moves that may lead to dead ends. In the end there was still a chance that the algorithm could have solved the eight puzzle problem, but these were very slim due to the configurations of the problems.

Random Restart hill climbing is another variation of the hill climbing algorithm. This algorithm creates a new starting position for the problem when it gets stuck on a local maxima to be able to locate the global maxima. As mentioned before, the problems 8 puzzle and 8 queens were analyzed and 10 instances were created to study its pattern and the effectiveness of the algorithm under test. With the specifics with random restart the problem would be given 10 restarts to start with and increasing or decreasing those opportunities of restarting to find the effectiveness on the number of restarts had on the problem. In the Eight Puzzle problem the search cost of each choice was compared to the Manhattan heuristic function. This heuristic compared the distance from the current position to the goal. Even with this function, it did not always guide the search to the correct solution. For the Eight Queens problem, however, the heuristic used was the number of attacking queen pairs. With the goal to lower this value. It can be seen that using this heuristics the results could be accurately calculated and compared with the optimal solution found. When performing the tests, hill climbing was found as not the best algorithm for solving the given problems. As even with restarting the algorithm on a random place it could not solve the problem for 8 puzzle even when reaching the number of allowed restarts to be 100. It would get stuck on a local maxima and be unable to improve its current state with its heuristic function not finding a better path of improvement. Meanwhile in most cases the problem for 8 queens was solved by 90% on average with 10 restarts when increased to 20 it would average 98% efficiency in solving the 8 queens problem.

Simulated Annealing, while a powerful optimization technique, proved to be ineffective for both the 8-puzzle and 8-queens problems. Despite using suitable heuristics, the algorithm struggled to find solutions. The 8 puzzle test resulted in a 0% success rate, primarily due to the algorithm's tendency to get stuck in local optima, given the limited movements and space. Similarly, for the 8-queens problem, Simulated Annealing had a 20% success rate in one variation of the code where it was filling the board with queens and 0% in another where the queens were already on the board and needed to be accommodated where no queens where attacking each other, highlighting its challenges with finding solutions efficiently in a problem with many potential configurations. The inherent randomness of the algorithm and its tendency to accept worse solutions to explore the search space hindered its ability to consistently converge on the optimal solution, making it less suitable for these problems compared to other specialized algorithms. These results indicate that while Simulated Annealing may work well for problems with large, complex search spaces where local optima is abundant, it may struggle with problems that have more constrained or deterministic solution spaces.

While hill climbing and its variants offer a promisingly robust approach to solving optimization problems, their effectiveness highly depends on the problem structure and starting conditions. Unlike the 8 puzzle problem which proved challenging, the 8 queens problem benefited from how flexible these algorithms are. The results showcased high success rates with methods like Random Restart, which was the most successful at solving this kind of problem. Simulated Annealing's randomness did not show any significant advantage in any of the

problems. This suggests that for local search algorithms to be effective at obtaining the goal state, it needs to be tailored to a specific kind of problem or else it won't be successful.

References:

[1] Aimacode, “aima-python,” *github.com*, Feb. 1, 2016. [Online]. Available:https://github.com/aimacode/aima-python . [Accessed Mar. 20, 2025].

[2] U. Rawat, "Hill Climbing in Artificial Intelligence," *geeksforgeeks.org*, Oct. 10, 2024. [Online]. Available: https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/. [Accessed: Mar. 25, 2025].

[3] Cyfuture Cloud, "Introduction to Hill Climbing in Artificial Intelligence,"*cyfuture.cloud*, [Online]. Available: https://cyfuture.cloud/kb/general/introduction-to-hill-climbing-in-artificial-intelligence. [Accessed: Mar. 25, 2025].

[4] Z. Akhtar, "Hill Climbing Algorithm in Artificial Intelligence," *databasetown.com* , Aug. 6, 2023. [Online]. Available: https://databasetown.com/hill-climbing-algorithm-in-artificial-intelligence/. [Accessed: Mar. 25, 2025].

[5] D. Dewcom, "What is Simulated Annealing,", *geeksforgeeks.org,* Sept. 12, 2024. [Online]. Available: https://www.geeksforgeeks.org/what-is-simulated-annealing/. [Accessed: Mar. 26, 2025].

Code Repository:

https://github.com/Dahyna-Martinez/Agents-and-Task-Management