COIS 4550H

Assignment 2

Winter 2025

Bachelor Of Computer Science, Trent University

COIS-4550H: Artificial Intelligence

Professor: Sheikh Ahmed Munieb

2nd March, 2025

Dikshith Reddy Macherla - Student Id: 0789055

Roman Bamrah - Student Id: 0705310

Assignment 2

Question 1

a. Given

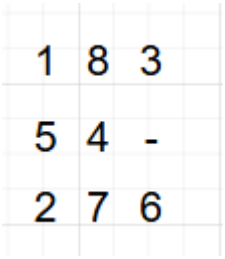
Initial State:

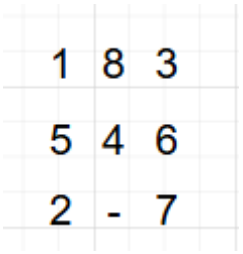
1	8	3
5	4	6
2	7	

Goal State:

1	2	3
4	5	6
7	8	

From the initial state, we can compare it to the goal state, and determine what numbers are in the correct position and what numbers need moving. Numbers (1,3,6) are in the correct positions, meaning the heuristic value is 3 and does not need moving.





The only possible moves are either to move left or to move up.
When you move up, the number of correct positions is 2 (1,3) this is rejected because the number of correct positions has gone down in comparison to the initial state (3 down to 2)

When you move left, the number of correct positions is 3 (1,3,6) this is also rejected because the number of correct positions has not changed from the initial state, so the local hill climb algorithm has reached a local maximum and is stuck as it cannot see any more moves that will increase the amount of correct positions.

b. Given

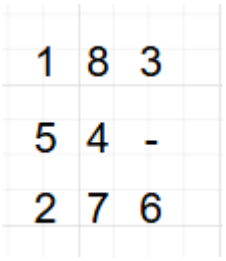
Initial State:

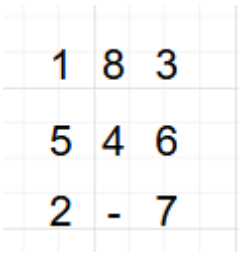
1	8	3
5	4	6
2	7	

Goal State:

1	2	3
4	5	6
7	8	

From the initial state, we can compare it to the goal state, and determine what numbers are in the correct position and what numbers need moving. Numbers (1,3,6) are in the correct positions, meaning the heuristic value is 3 and does not need moving.

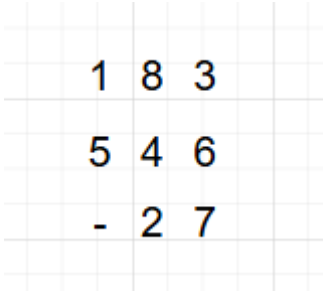


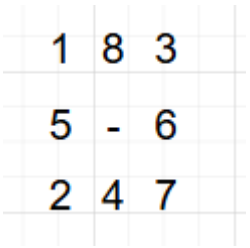


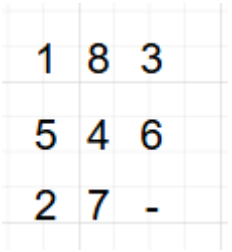
Step 1. The only possible moves are either to move left or to move up.

When you move up, the number of correct positions is 2 (1,3) this is rejected because the number of correct positions has gone down in comparison to the initial state (3 down to 2)

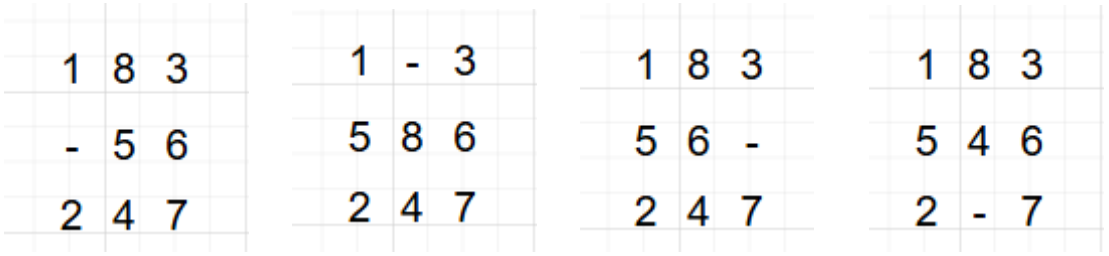
When you move left, the number of correct positions is 3 (1,3,6), since we are using a global heuristic approach, this is accepted because the number of correct positions has not changed from the initial state.



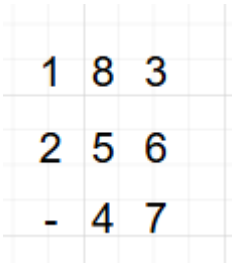




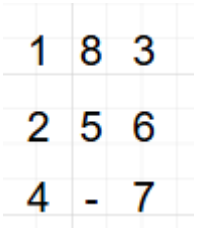
Step 2. After Step 1, the possible moves are left, up, and right. Since none of the moves increase the heuristic nor decrease it, a random path is chosen. Up will be the move chosen for step 3.



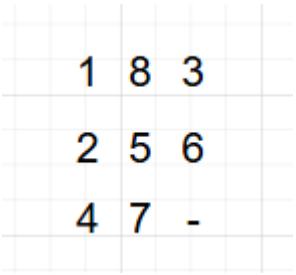
Step 3. The next possible moves are left, up, right, and down. In this position, left is the only position that will increase the amount of correct positions.



Step 4. The only moves possible in step 3 are up, down and right. Since up decreases correct positions, it is not chosen. Down and right are the only possible steps, so down is chosen



The only options are up, left and right. Up is not chosen because it lowers the amount of correct positions, so left or right can be chosen. In this scenario, right is chosen randomly.



In this position, up cannot be picked because it lowers the amount of correct positions, and left does not increase the amount of good positions In this scenario, it is stuck at a maximum, and the only things to do would be backtrack or use random restarts. The correct positions now are (1,3,5,6 and -)

Question 2

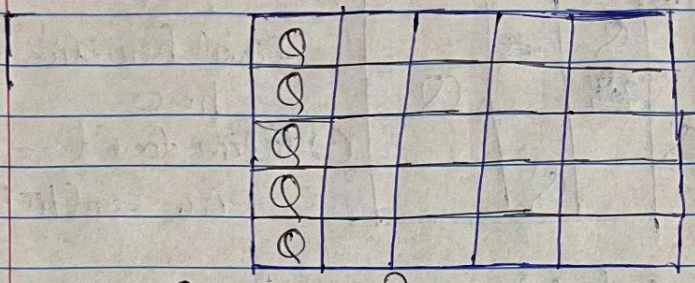
COIS 4550H - Artificial Intelligence

Assignment - 2

Question - 2

Step 1: Initial board configuration

the queens are initially placed in the first columns:



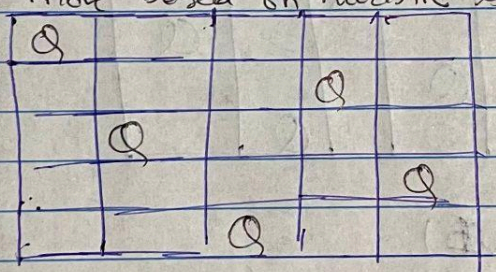
Heuristic h=10

(since all queens attack each other)

Step 2: Generating Possible Moves

each queen can be moved vertically within its respective column to minimize conflicts.

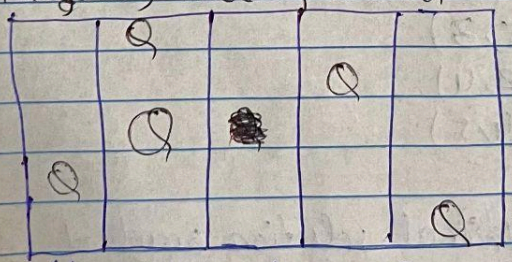
Best move based on heuristic is



new heuristic h=4

(some conflicts remain)

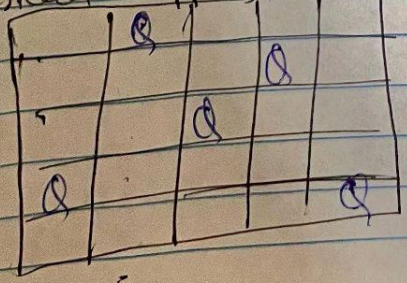
Step 3: continuing moving toward an optimal state



new heuristic h=2

Step 4: Handling local maxima

At this stage, no ~~further~~ move further reduces the heuristic instead stopping, we allow sideways moves to escape local maxima



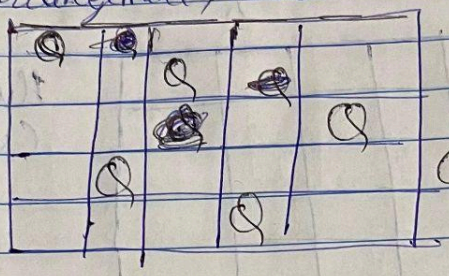
SIDEWAYS MOVE

Heuristic remains h=2

(sideways move used to explore further states)

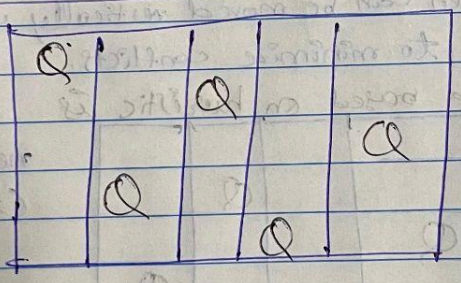
Step 5: Reaching an optimal solution

After a few more sideways move, we find an arrangement, where no queens attack each other



final heuristic
h=0
(solution found = no conflicts)

Final Board solution:



Queen 1 = (1,1)
Queen 2 = (3,2)
Queen 3 = (5,3)
Queen 4 = (2,4)
Queen 5 = (4,5)

This is an optimal solution reached using hill climb with sideways moves, successfully avoiding local maxima.

Question 3

PYTHON CODE

```
import random

def calculate_attacks(board):
    """Calculate the number of attacking pairs in the board."""
    attacks = 0
    n = len(board)

    for i in range(n):
        for j in range(i + 1, n):
            if board[i] == board[j] or abs(board[i] - board[j]) ==
abs(i - j):
                attacks += 1

    return attacks

def get_best_neighbors(board):
    """Find the best neighboring states with the minimum heuristic
value."""
    n = len(board)
    min_attacks = calculate_attacks(board)
    best_moves = []

    for col in range(n):
        for row in range(n):
            if board[col] != row:
                new_board = list(board)
                new_board[col] = row
                attacks = calculate_attacks(new_board)

                if attacks < min_attacks:
                    min_attacks = attacks
                    best_moves = [new_board]
                elif attacks == min_attacks:
                    best_moves.append(new_board)

    return best_moves, min_attacks

def hill_climbing_with_sideways(n, max_sideways=100,
max_restarts=10):
    """Solve the N-Queens problem using Hill Climbing with Sideways
Moves and Backtracking."""
    for _ in range(max_restarts):
        board = [random.randint(0, n - 1) for _ in range(n)]  #
Random initial state
        heuristic = calculate_attacks(board)
        sideways_moves = 0

        while heuristic > 0:
            best_neighbors, best_heuristic =
get_best_neighbors(board)

            if best_heuristic >= heuristic:
                # Allow sideways moves, but limit them
```

```
                if sideways_moves >= max_sideways:
                    break  # Restart if stuck in local maximum
                sideways_moves += 1
            else:
                sideways_moves = 0  # Reset sideways move counter if
improvement

            board = random.choice(best_neighbors)  # Move to a random
best neighbor
            heuristic = best_heuristic

        if heuristic == 0:
            return board  # Found a valid solution

    return None  # No solution found after max restarts

def print_solution(board):
    """Print the chessboard with queens placed."""
    if board is None:
        print("No solution found")
    else:
        n = len(board)
        for row in range(n):
            line = ["Q" if board[col] == row else "_" for col in
range(n)]
            print(" ".join(line))
        print("\n")

# Solve the 5-Queens problem
solution = hill_climbing_with_sideways(5)
print_solution(solution)
```

Question 4

a. The methods used by Beam Search and Hill Climbing to explore search spaces are different. While Hill Climbing only investigates one path at a time, making it more vulnerable to local traps, Beam Search keeps several candidates (beam width) at every step, decreasing the likelihood of becoming trapped in local maxima. Beam Search makes a compromise between efficiency and exploration, but requires greater memory. On the other hand, Hill Climbing, a greedy algorithm frequently identifies local optima rapidly but could overlook the optimal solution.

b. The methods used to find the best answers in Beam Search and Hill Climbing are different. By analyzing several candidates at each stage and keeping just a certain amount (k) of the best ones, Beam Search helps to prevent local maxima and guarantees a more thorough investigation of the search space. This is in contrast to Hill Climbing, which is faster but more likely to become trapped in local optima because it only takes one path, always heading toward the best immediate neighbor. While Hill Climbing is more memory-efficient, it may overlook the best answer, while Beam Search uses more memory because it keeps track of several possibilities. While Hill Climbing is frequently used in optimization and robotics, where speed is more significant than global optimality, Beam Search is frequently used in AI applications like natural language processing and speech recognition, where locating a globally good solution is essential.