```
import random

def hill_climbing_n_queens(n):
    # Step 1: Initialize a random board with one queen in each column
    board = generate_random_board(n)

    while True:
        # Calculate the current number of conflicts
        current_cost = calculate_conflicts(board)

        # If solution is found (no conflicts), return the board
        if current_cost == 0:
            return board

        # Step 3: Find the neighbor with the lowest number of conflicts
        next_board, next_cost = get_best_neighbor(board)

        # Step 4: Check if we've reached a local minimum
        if next_cost >= current_cost:
            # If stuck in local minimum, restart with a new board
            board = generate_random_board(n)
        else:
            # Move to the better board configuration
            board = next_board

def generate_random_board(n):
    # Generates a random board with one queen in each column
    return [random.randint(0, n - 1) for _ in range(n)]

def calculate_conflicts(board):
    # Counts the number of pairs of queens that are in conflict
    conflicts = 0
    for i in range(len(board)):
        for j in range(i + 1, len(board)):
            if board[i] == board[j] or abs(board[i] - board[j]) == abs(i - j):
                conflicts += 1
    return conflicts

def get_best_neighbor(board):
    n = len(board)
    best_board = board[:]
    best_cost = calculate_conflicts(board)

    # Try moving each queen in each column to a different row
    for col in range(n):
        original_row = board[col]
        # Check each possible row for this column
        for row in range(n):
            if row != original_row:
                board[col] = row
                cost = calculate_conflicts(board)

                # Keep track of the best board with minimum conflicts
                if cost < best_cost:
                    best_cost = cost
                    best_board = board[:]

        # Revert the queen to its original row
        board[col] = original_row

    return best_board, best_cost

# Example usage:
n=int(input("No of queens: "))  # Change n to desired board size
solution = hill_climbing_n_queens(n)
print("Solution for", n, "queens:")
print(solution)
```

```
No of queens: 4
Solution for 4 queens:
[1, 3, 0, 2]
```

+ Code + Text