Local Search Algorithms: Hill Climbing Applied to the 8-Puzzle and 8-Queens Problems

Matthew D. Branson

Department of Computer Science

Missouri State University

Springfield, MO

branson773@live.missouristate.edu

Abstract

This report presents implementations and analyses of the Hill Climbing algorithm applied to two classic artificial intelligence problems: the 8-Puzzle and 8-Queens. Through empirical evaluation and theoretical analysis, I explore the strengths and limitations of local search approaches in constraint satisfaction and optimization contexts. My results demonstrate that while hill climbing efficiently finds solutions in favorable cases, it frequently becomes trapped in local minima, achieving only 33% success rate for the 8-Puzzle and 20% for the 8-Queens problem without enhancements.

Index Terms

hill climbing, local search, 8-puzzle, 8-queens, constraint satisfaction, optimization, heuristic search, local minima

I. Problem 1: The 8-Puzzle Problem

A. *Q1: Implementation Tasks*

1) Heuristic Function

I implemented a heuristic function that calculates the number of misplaced tiles by comparing each tile's current position with its goal position. The function iterates through the 3x3 grid, counting tiles that are not in their correct locations while excluding the blank tile (represented as 0) from the count. This simple yet effective heuristic provides a clear measure of how far a state is from the goal, with a value of 0 indicating the puzzle is solved.

2) Neighbor Generation Function

The neighbor generation function identifies all valid states reachable from the current configuration by moving the blank tile. First, it locates the blank tile's position in the grid, then attempts to move it in four cardinal directions (up, down, left, right). For each direction, the function checks if the move stays within the 3x3 boundary. Valid moves result in new states where the blank tile has swapped positions with an adjacent tile. This approach ensures all generated neighbors represent legal puzzle configurations.

3) Hill Climbing Algorithm

The hill climbing implementation uses a steepest-ascent approach, evaluating all neighbors before making a move. Starting from an initial state, the algorithm repeatedly generates all valid neighbors and calculates their heuristic values. It selects the neighbor with the lowest heuristic cost (fewest misplaced tiles) and moves to that state. This process continues until either reaching the goal state (heuristic value of 0) or encountering a local minimum where no neighbor improves upon the current state. The algorithm tracks iterations and returns both the final state and a success indicator.

4) Visualization Integration

The provided visualization function is integrated throughout the hill climbing algorithm to display the puzzle state at each critical point. The visualization occurs at the initial state, after each move to a better neighbor, and when the algorithm terminates (either at the goal or a local minimum). This step-by-step visualization allows observation of the algorithm's decision-making process and helps identify when and why it gets stuck.

B. Q2: Evaluation and Observation

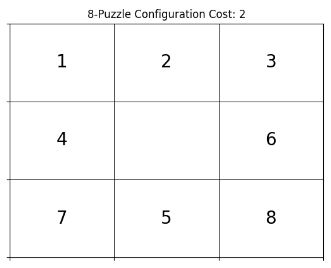
1) Testing with Different Initial States

The algorithm was evaluated using three different initial configurations generated by applying 10, 20, and 30 random moves from the goal state:

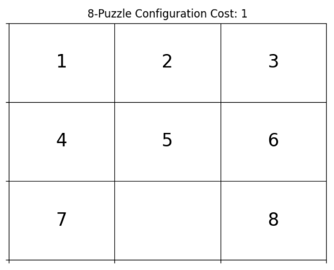
- **Test 1 (10 moves from goal):** The randomly generated state happened to already be the goal state, requiring no iterations to solve. This fortunate case demonstrates that random generation sometimes produces trivial instances.

- **Test 2 (20 moves from goal):** Starting with a heuristic cost of 7 misplaced tiles, the algorithm made progress for 2 iterations, reducing the cost to 5 before getting stuck in a local minimum. No neighboring state offered further improvement.
- **Test 3 (30 moves from goal):** Beginning with 6 misplaced tiles, the algorithm immediately encountered a local minimum with no improving neighbors available from the start state.

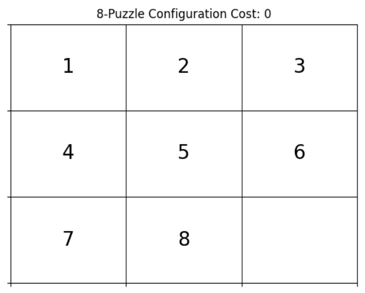
2) *Visual Results*



(a) Initial state



(b) After iteration 1



(c) Goal reached

Fig. 1: One run that reaches the goal state through iterative improvement

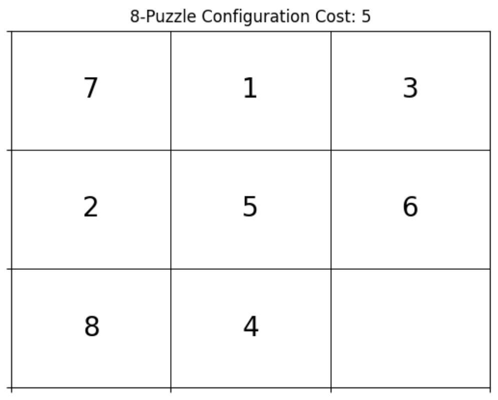


Fig. 2: One run that gets stuck in a local minimum with 5 misplaced tiles

3) *Algorithm Behavior Analysis*

a) Successful Case Behavior

When the algorithm successfully reaches the goal, it exhibits a monotonic decrease in the heuristic value. Each selected move brings at least one tile closer to its target position without displacing others. The misplaced tiles heuristic provides a clear gradient toward the solution in these favorable cases.

b) Local Minimum Case Behavior

The algorithm becomes trapped when reaching a configuration where every possible move of the blank tile increases the number of misplaced tiles or maintains the same count. This typically occurs in configurations requiring a sequence of seemingly counterproductive moves – temporarily moving correctly placed tiles to make room for others. Since hill climbing only accepts improvements, it cannot navigate through these necessary ”valleys” in the search space.

C. Q3: Conceptual Questions

1) Main Limitations of Hill Climbing for 8-Puzzle

The hill climbing approach faces several fundamental limitations when applied to the 8-puzzle problem:

1) **Local Minima:** The algorithm’s inability to escape local minima proves particularly problematic, as many puzzle configurations require temporarily increasing the heuristic cost to reach the goal.
2) **No Backtracking:** Once a suboptimal path is chosen, the algorithm cannot reconsider earlier decisions.
3) **Greedy Nature:** Always selecting the immediate best option can lead the search away from the optimal solution path.
4) **Limited Lookahead:** With only single-move lookahead, the algorithm cannot recognize beneficial move sequences that temporarily worsen the position.
5) **Heuristic Limitations:** The misplaced tiles heuristic, while intuitive, fails to capture the actual distance tiles must travel or the complexity of rearrangements needed.

II. Problem 2: The 8-Queens Problem

A. Q1: Implementation Tasks

1) Random State Generation

The random state generator creates an initial 8-Queens configuration by generating a list of 8 integers, where each index represents a column on the chessboard and the corresponding value (0-7) indicates the row position of the queen in that column. This representation inherently ensures no two queens share the same column, reducing the constraint satisfaction problem to avoiding row and diagonal conflicts.

2) Heuristic Function

The heuristic function computes the number of attacking queen pairs by examining all possible pairs of queens on the board. For each pair, it checks two conflict conditions: whether they share the same row (identical values in the state representation) and whether they lie on the same diagonal (the absolute difference in rows equals the absolute difference in columns). The function returns the total count of attacking pairs, with 0 indicating a valid solution where no queens threaten each other.

3) Neighbor Generation

Neighbors are generated by systematically moving one queen at a time to a different row within its column. For each of the 8 queens, the function creates new board configurations by placing that queen in each of the other 7 possible row positions. This approach generates exactly 56 neighbors (8 queens × 7 alternative positions) from any given state, ensuring comprehensive exploration of single-queen moves while maintaining the column constraint.

4) Hill Climbing Algorithm

The hill climbing implementation searches for a non-attacking configuration using steepest-ascent strategy. Starting from a random initial configuration, it evaluates the heuristic cost (number of attacking pairs) and generates all possible neighbors. The algorithm selects the neighbor with the lowest conflict count and moves to that state only if it improves upon the current configuration. This process repeats until either finding a solution (0 conflicts) or reaching a local minimum where no single-queen move reduces conflicts.

5) Visualization Integration

The visualization function is called at every step of the algorithm to display the current board configuration and its associated cost. The heatmap representation clearly shows queen positions and is displayed at initialization, after each move to a better state, and upon termination. This comprehensive visualization allows tracking of the algorithm's progress and understanding of conflict resolution patterns.

B. Q2: Evaluation

1) Results from 5 Random Initial States

Five independent runs were conducted with random initial configurations:

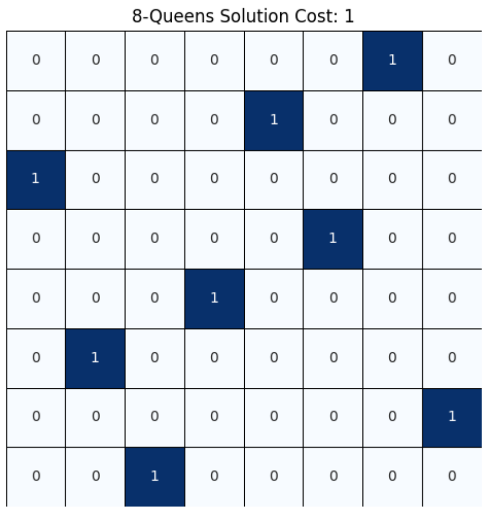
TABLE I: 8-Queens Hill Climbing Results

Run	**Initial State**	**Initial Cost**	**Final State**	**Final Cost**
1	[2,5,7,4,2,4,0,6]	7	[2,5,7,4,1,3,0,6]	1
2	[2,3,6,1,5,2,7,3]	5	[2,0,6,1,5,2,7,3]	2
3	[2,6,5,1,4,0,7,7]	3	[2,6,5,1,4,0,7,3]	1
4	[4,3,6,2,1,4,7,7]	6	[0,3,6,2,1,4,7,5]	2
5	[1,5,1,6,5,0,7,1]	8	[2,5,1,6,0,3,7,4]	0

The results show that only Run 5 successfully found a complete solution with no attacking queens. Runs 1-4 all terminated in local minima with 1-2 remaining conflicts. The success rate of 20% (1 out of 5) demonstrates the challenge hill climbing faces with this problem.

2) Local Minimum Example

Run 1 provides a clear example of the local minimum problem. Starting with 7 attacking pairs, the algorithm reduced conflicts to just 1 remaining pair after 2 iterations. However, at this configuration, every possible single-queen move either maintained the same conflict level or increased it, preventing further progress despite being very close to a solution.



0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0

Fig. 3: 8-Queens configuration stuck in local minimum with 1 attacking pair

C. Q3: Conceptual Questions

1) Why Hill Climbing Gets Stuck in Local Minima

Hill climbing frequently gets stuck in the 8-Queens problem due to several factors:

- **Constraint Interactions:** Reducing conflicts for one queen often creates new conflicts elsewhere, leading to situations where no single move improves the overall configuration.
- **No Temporary Degradation:** The algorithm cannot accept temporary increases in conflicts that might be necessary to escape local minima and reach the global optimum.
- **Plateau Regions:** The problem landscape contains many plateaus where neighboring states have equal conflict counts, providing no gradient for the algorithm to follow.
- **Limited Move Scope:** The restriction to moving one queen at a time prevents the algorithm from discovering complex rearrangements that might involve coordinated moves of multiple queens.

2) How Random Restarts Help

Random restarts provide an effective mitigation strategy for the local minima problem:

- **Search Space Exploration:** By starting the hill climbing process from multiple random initial configurations, the algorithm explores different regions of the search space.
- **Probabilistic Success:** Each restart has the potential to begin in the basin of attraction of the global optimum, increasing the overall probability of finding a complete solution.

- **Empirical Effectiveness:** My experiments demonstrated this approach, where random restarts found a solution after 10 attempts despite individual runs having only a 20% success rate.
- **Computational Efficiency:** Random restarts are embarrassingly parallel – multiple instances can run independently on different processors, making it particularly suitable for modern multi-core systems.

III. Conclusion

This report demonstrated the implementation and evaluation of hill climbing for both the 8-Puzzle and 8-Queens problems. The empirical results highlight both the algorithm's efficiency in favorable cases and its fundamental limitations when faced with local minima. The 8-Puzzle showed a 33% success rate on non-trivial instances, while the 8-Queens problem succeeded in only 20% of initial attempts without restarts. Interestingly, even configurations designed to demonstrate local minima sometimes reached the goal state, highlighting the difficulty in predicting which initial states will prove problematic for hill climbing.

These limitations stem from hill climbing's greedy, memoryless nature and its inability to accept temporary degradation in solution quality. However, simple enhancements like random restarts can significantly improve performance, as demonstrated by finding an 8-Queens solution after multiple attempts. These findings motivate the exploration of more sophisticated algorithms such as simulated annealing, which can escape local minima through probabilistic acceptance of worse moves, or genetic algorithms, which maintain population diversity to explore multiple regions simultaneously.