# Data Structure and Algorithm: Final Report Solving N-Queens Problem using Hill Climbing and Simulated Annealing

1st Tingcong Jiang
*Rutgers University New Brunswick*
*ECE Department*
NJ, USA
tj215@scarletmail.rutgers.edu

2nd Zhuohuan Li
*Rutgers University New Brunswick*
*ECE Department*
NJ, USA
zl253@scarletmail.rutgers.edu

***Abstract*—8-Queens Puzzle was first introduced in 1848, in this puzzle, there is a 8*8 chess board with 8 queens. The rule is that players need to find an arrangement of placing all 8 queens on the board such that there is no two queens that share on the same column, row, or diagonal.**

**In this paper, we are going to implement N-Queens Problem in the N*N chess board with two different types of optimization algorithms: Simulated Annealing and Hill Climbing based on the idea of 8-Queens Puzzle problem. In this paper, we are going to test algorithms not only for 8-Queens, but also for different input numbers N of queens. In this case, we can observe how the algorithms work in different cases with more detailed performance analysis. The Simulated Annealing algorithm and the Hill Climbing are both optimization algorithms to find optimal solutions for certain problems. The Hill Climbing algorithm is more focusing on the improvement of the optimization. However, the idea of the Simulated Annealing is to get more exploration which is more likely to help finding the global optima rather than a local optima.**

**By implementing both algorithms-based N-Queens Puzzle, the goal of our group is to compare the performance of both algorithms-based N-Queens Puzzle under different circumstances.**

***Index Terms*—N-Queens, algorithm, Hill Climbing, Simulated Annealing**

## I. Introduction

N-Queens problem is computationally expensive because there are $8^8$ = 16777216 possible arrangements for an 8-queens problem even after we applied a simple rule that no queens can be on the same row or column. Such huge search space can easily exhaust any resource on an average computer. Thus, we need algorithms that are faster and more efficient.

Before we move to the Hill Climbing and Simulated Annealing algorithms, first w need to classify our search methodologies. We first retrieve a random starting point of the N-Queens problem such that no queen can be placed on the same column. Then, in each state, we limit ourselves that we can only move one queen at a time. Thus, between any two adjacent states, only one queen's location is changed. Then, we calculate the number of attacking pairs for each possible move, and this is where the Hill Climbing algorithm and the Simulated Annealing comes in.

The Hill Climbing algorithm always chooses the best move which is defined as a move that leads to the least number of attacking pairs. However, the problem with the Hill Climbing is that there is a chance that the algorithm reaches a local solution such that there will be no more move that can lead to less number of attacking pairs. To address this problem, a complete solver will restart the whole process with another random starting point and hope that the new starting point can lead to a global solution. Furthermore, allowing some bad moves that lead to an equal or worse situation can help with getting rid of local solutions.

On the other hand, the Simulated Annealing algorithm will choose a good move available or a bad move for a probability that is determined by two factors which are how many moves have been made and how bad the move is. In both cases, random walk can reduce the possibility of stopping at a local maximum or solution. However, we want to know which solver strategy has better efficiency and flexibility, and we will implement the Hill Climbing and Simulated Annealing algorithm to solve the N-Queens problem with several experiments.

The experiments will be conducted on the same computer to guarantee the same platform configuration which will not affect run time result and more details will be included in the experiment part.

## II. Heuristic Function

Definition of consistent heuristic function:

$$h(n) \leq g\left(n, n'\right) + h\left(n'\right)$$

where n' is a neighbor position of n and g(n, n') is the actual cost of moving from n to n'

A heuristic function is a function that ranks alternatives in search algorithms at each branching step based on available information to decide which branch to follow. For example, it may approximate the exact distance from current stage to our final stage.

In this scenario, we will take the number of attacking queen pairs as the heuristic function. The number of attacking queen pairs is calculated by counting the number of queens that are sharing the same row, column, and diagonal for each queen.

The goal of our algorithm is to reach the number of attacking queens to zero which means we find a solution in which there is no queen that can attack from each other.

## III. The Hill Climbing Algorithm

### *A. Definition*

Hill Climbing is a heuristic local search used for mathematical optimization problems in the field of Artificial Intelligence. Given a large set of inputs and a good heuristic function, it tries to find a sufficiently good solution to the given problem.

### *B. Features*

- Variant of generate and test algorithm
  - Generate a possible solution
  - Test the solution generated by step 1
  - If the solution is a global solution, then quit and return
  - Else, keep looping back to step 1 until a global solution is found

  Because of above features, we call Hill Climbing algorithm as a variant of generate and test algorithm as it takes the feedback from the test procedure. Then this feedback is utilized by the generator in deciding the next move in search space.
- Greedy Approach
  - At each state, the search algorithm only moves to a direction that will optimize the cost function trying to find the optimal solution at the end.

### *C. Process*

The process of hill climbing is like people climbing the mountain and trying to reach the highest place of the mountain step by step in ascending heights.

Simple Hill climbing : It examines the neighboring nodes one by one and selects the first neighboring node which optimizes the current cost as the next node. Algorithm for Simple Hill climbing :

- Evaluate the initial state
  - If it is a goal state then stop and return success. Otherwise, make initial state as current state
- Loop until no new operators
  - Take the initial state as current state
  - Perform following conditions to evaluate new state
    - ∗ Find all the neighbor states, and choose the best one among them and take it as the potential current state
    - ∗ If it is not better than the current state, and we have not yet reached the goal state (local optimum), back to evaluate stage
    - ∗ If it is not better than the current state, and we have reached the goal state (global optimum), return current state
- Exit if a solution is found

### *D. Advantage and Disadvantage*

- Advantage:
  The success of hill climb algorithms depends on the architecture of the state-space landscape. It can be used in conversion as well as discrete domains and can be easily implemented. Whenever there are few maxima and plateau the variants of hill climb searching algorithms work very fine.
  The Hill Climbing algorithm is also very simple to be implemented.
- Disadvantage:
  However, in real-world problems there will be a landscape that looks more like a widely scattered family of balding porcupines on a flat floor, with miniature porcupines living on the tip of each porcupine needle. NP-Hard problems typically have an exponential number of local maxima to get stuck on. Hill Climbing suffers from the following issues:
  - Foot-Hills:
    It is a state which is better than all of its neighbours but isn’t better than some other states which are farther away. At local maxima, all moves appear to make things worse. They are sometimes frustrating also as they often occur almost within sight of a solution. So, it is also called as Foot-Hills.
  - Ridge:
    It is a special kind of local maximum. it is an area of the search space which is higher than the surrounding areas and that itself has a slope. We can’t travel the ridge by single moves as the orientation of the high region compared to the set of available moves makes it impossible.
  - Plateau:
    It is a flat area of the search space in which a whole set of neighbouring states(nodes) have the same order. On a plateau, it is not possible to determine the best direction in which to move by making local comparisons. Actually, a plateau is an area of the state space landscape where the evaluation function is flat. It can be a flat local maximum or a shoulder from which it is possible to make progress.

  Therefore, there are many more advanced algorithms that have been developed to overcome the issues introduced before: such as Simulated Annealing, Local Beam Search and other genetic algorithms. For this project, we will focus on and dig into the implementation of the Simulated Annealing and conduct experiments between Hill Climbing and Simulated Annealing.

## IV. The Simulated Annealing Algorithm

Simulated Annealing is a probabilistic technique that is inspired by annealing in metallurgy, and it is used to approximate the global optimum of a given function. While annealing is a heat treatment process which alters the microstructure of a material to change its mechanical or electrical properties, it

| | | |
|---|---|---|
| Algorithms | Hill Climbing | Simulated Annealing |
| Greediness | Always Greedy | Increasing greediness while temperature drops |
| Implementation | Heuristic, Generate Initial States, Get All Neighbors | Heuristic, Generate Initial States, Get One Neighbor, Probability |
| Parameters | Goal | Goal, Temperature, Threshold, Cooling Factor, # of iterations |
| Run Time | Average | Fast |
| Solution | Local Optimum or Global Optimum | Approximation of Global Optimum |

TABLE I
HILL CLIMBING VS. SIMULATED ANNEALING

can be related to search problems. For most informed search problems, we are given an initial state and heuristic function, and the initial state of a such search problem can be seen as the initial microstructure of a given metal. In general, when temperature is high, atoms will tend to migrate in the crystal lattice and the number of dislocations reduces, and similarly, when the simulated annealing algorithm starts to explore neighbors states, we can treat this process as an atom randomly migrating between states. Furthermore, when the temperature is decreased, the metal starts to recrystallize, and we can see this as the algorithm tends to stay at one particular state. Finally, when the temperature is below a given threshold, the metal has transformed its original microstructure to another more optimal one. This transition depicts that the simulated annealing algorithm has achieved an optimal state.

In short, simulated annealing algorithm makes guesses based on current state and choose the the guesses that has the most improvement. However, what differs simulated annealing from hill climbing is that simulated annealing accepts bad moves. This is to say that if it reached an local maximum, ridge, or plateau, it can get away from these situations by accepting neighbors states that have the same or worse heuristic value based on a acceptance probability. Furthermore, simulated annealing has several important parameters: Initial temperature, Cooling Rate, Number of iterations between each temperature drop, Threshold. Each of these parameters plays an significant role in simulated annealing algorithm. Initial temperature decides the maximum tolerance of accepting bad moves, and cooling rate decides the greediness of algorithm over time. Furthermore, the number of iterations between each temperature drop decides how many guesses that the algorithm need to consider for each given temperature. Lastly, the threshold decides how close the outputted approximated solution is to the global solution. For each search problem, these parameters should be fine tuned to fit the problem's needs.

## *A. General Procedure*

- Take a initial state
  - Randomly generate an initial state.
- Make Guesses
  - Between Each Temperature Drop, make several guesses, and accept guesses as current state based on an acceptance probability. If a guess reaches goal, return the guess.
- Final Output
  - Repeat previous step until temperature drops below a given threshold, and return current state.

## *B. Acceptance Probability*

Simulated annealing algorithm has a acceptance probability that is defined as:

$$P_{acceptance} = e^{\frac{-\triangle E}{T}}$$

where $\triangle E$ = change in heuristic value

$T$ = current temperature

This acceptance probability is inspired by transitions of a physical system, and it has 2 intuitions.

- When temperature is fixed, the higher the change in energy is, the lower the probability is.
  - Tends to accept a better move among the bad moves.
- When the change in energy is fixed, the lower the temperature is, the lower the probability is.
  - The greediness of simulated annealing is inverse proportional to its current temperature.

All of these parameters and acceptance probability contributes to the outstanding performance of simulated annealing. Furthermore, simulated works with problems such that has an unknown optimal solution. However, simulated annealing also has flaws. Simulated annealing cannot guaranty to output the global solution of a given function since it only outputs an approximation of the global solution. Furthermore, these parameters has no clear rule to set them up. Therefore, fine tuning these parameters is burdensome. But, it still has many application in engineering problem such as sales man route scheduling, large-scale aircraft trajectory planning, and many more.

## V. EXPERIMENT

## *A. Set Up*

Each experiment consists of solving n-queen problem with n increasing from 6 to 29 with increment 1, and each experiment is repeated 20 times for each algorithm. In this case, we will use hill climbing and simulated annealing as the solvers to solve n-queen problem. The solvers run on a computer with i9-10900KF, and each of them only utilizing one thread at any given time. Therefore, the impact of computer performance is minimized. For each round of the experiment, we will record all execution time of the 20 iterations, and calculate its average and standard deviation to form an error bar graph.

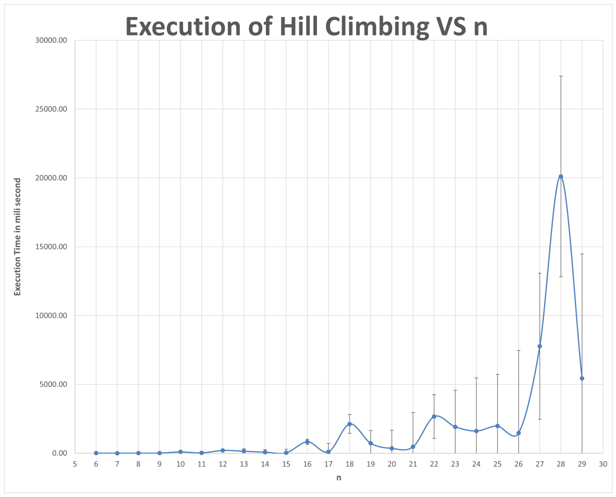

Fig. 1. Running result of Hill Climbing

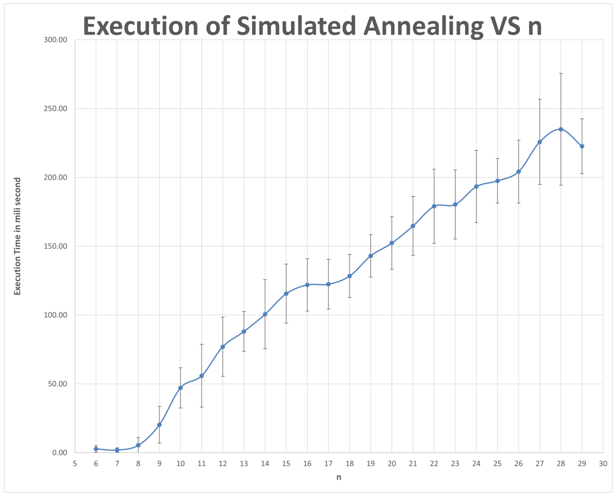

Fig. 2. Running result of Simulated Annealing

### B. Experiment Result

The experiment result for hill climbing is shown in Figure 1, and the experiment result for simulated annealing is shown in Figure 2. Overall, simulated annealing is much faster than hill climbing for most of the cases. The reason for simulated annealing outperformed hill climbing is that simulated annealing can escape from local optimum to reach global optimum which is want we trying to find. Unlike simulated annealing, when hill climbing algorithms faces an local optimum, it will takes another randomly generated initial state as input and try to climb to a reachable optimum again. Therefore, in the worst case, if every randomly generated initial state leads to an local optimum, the execution time of hill climbing is infinity. This is to say that, the performance of hill climbing significantly relies on the initial state, which is unpredictable. Therefore, the performance of simulated annealing is stable, predictable, and fast, and the performance of hill climbing is fluctuating.

Furthermore, from Figure 2, we can see that the simulated annealing has a jump in execution time for every increase of 3-5 in size n. On the other hand, from Figure 1, we can see that the hill climbing has a pattern of execution that is for each increase of 4-5 of n when n is greater than 18, the execution time will go up, down, down, down, and down. However, this is only an hypothesis that cannot be conclude from our study. To verify such hypothesis, the size of n should growth larger to see that if the pattern still exists. However, due to the limitation of computing power available to us, it takes tremendous amount of time to conduct such a experiment with n greater than 30. Therefore, verifying this hypothesis is our future work.

## VI. Conclusion

The characteristic of both hill climbing and simulated annealing is shown in TABLE I. First of all, hill climbing is essentially an greedy search algorithm, and hill climbing is easier to implement compared with simulated annealing because the procedure of hill climbing is more straight forward and simple. However, hill climbing is suffering from the problem of foot-hills, ridges, and plateau, which leads to an very fluctuating performance. On the other hand, simulated annealing can escape from these situations by accepting bad moves to improve its performance. However, simulated annealing can only offer an approximation of the global solution. Furthermore, simulated annealing has a lot of parameters that are need to be fine tuned for each given problem so that it can achieve an outstanding performance. But, for problems that has an unknown optimal solution, simulated annealing is a good candidate of solvers. In practical, simulated annealing is much faster than hill climbing, and the approximated solution is reasonably acceptable. In our experiment of solving n-queen problem by using the two algorithms, simulated annealing is also much more stable than hill climbing. This is to say that the execution time of simulated annealing for a given board size n is more predictable than the execution time of hill climbing. Overall, simulated annealing significantly outperformed hill climbing algorithm in our study.

## VII. GitHub Link

https://github.com/jasonqwerty100987/DSA_21_group8

## VIII. Contribution

Tingcong Jiang worked on implementing the Simulated Annealing algorithm and conducted all the experiments using one single computer. Tingcong Jiang also wrote all the report about the Simulated Annealing algorithm part, Experiment part and Conclusion part.

Zhuohuan Li worked on implementing the Hill Climbing algorithm and combine all the experiment data and plot the experiment results. Zhuohuan Li also wrote all the report about the Introduction part, the Hill Climbing algorithm part and the rest.

In all other project parts, Tingcong Jiang and Zhuohuan Li all collectively and equally contributed to the research, development of this project and presentation creation.

## References

[1] A. Tversky and D. Kahneman, “Availability: A heuristic for judging frequency and probability,” Cognitive Psychology, Volume 5, Issue 2, September 1973, Pages 207-232

[2] E.-G. Talbi and T. Muntean, “Hill-climbing, simulated annealing and genetic algorithms: a comparative study and application to the mapping problem,” Proceedings of the Twenty-sixth Hawaii International Conference on System Sciences, 8-8 January. 1993