

Introduction to Artificial Intelligence

Solving the N-Queens puzzle
using different heuristic search
strategies – local and systematic –
comparing and analyzing
performance

Submitter: Abdel-Qader

Introduction

The n-queens puzzle, in which the aim is to find starting from initial randomly generated state a configuration of the n queens on the n columns in which there are no two queens threatening each other, thus, no two queens are on the same row, column or diagonal.

History

The puzzle was originally proposed in 1848 by the chess player Max Bezzel, and over the years, many mathematicians, including Gauss, have worked on this puzzle. The first solutions were provided by Franz Nauck in 1850. In 1874, S. Günther proposed a method of finding solutions by using determinants.

Edsger Dijkstra used this problem in 1972 to illustrate the power of what he called structured programming. He published a highly detailed description of the development of a depth-first backtracking algorithm.

Clarifications Regarding the Implementation

There are two classes, the first is NQueens.java and the second is NQueensStar.java, in the first I implement the Random Restart Hill Climbing and the simulated annealing algorithm, in the second class I implement the A* algorithm and Best First Search (BFS).

A board state is represented by nxn matrix – two dimensional array –, each class has as a field Node variable that represents the initial state, which is generated randomly in the class constructor, in addition I have a field represents the dimension of the board, the class also has an inner class represents Node, I use as heuristic evaluation the number of pairs of queens that attack each other.

Random restart Hill-Climbing -- Local Heuristic search

In hill-climbing the algorithm chooses the best option from the neighbors of this state and updates current to be this neighbor, the problem of this approach is that the algorithm can get stuck with local optimum, so what can we do? The answer is random restart hill climbing, if at the first time the algorithm failed, try it again until a global optimum is found.

Simulated annealing -- Local Heuristic Search

The basic idea behind the simulated annealing (SA) algorithm is similar to the state of having a ping pong ball placed on a surface with local and global minimas, and we want to get to the global minimum, first we start to shake the surface hard without getting the ball out side the surface, while shaking we reduce the shaking temperature , so that if we reduce the temperature slowly enough, we end up with the ball in the global minimum when the temperature is zero, the scheduling of the temperature I used is linear in the time and obeys to the formula :

$$T(t) = (- t)/\text{iterationsNumber} + 1 \quad .$$

Best First Search – Systematic Heuristic Search

In best first search, we implement the general graph search with priority queue, the nodes representing the states are ordered according to the heuristic function, that is we choose to expand the node (state) with the lowest heuristic value (assuming the heuristic represent the estimated distance from the goal state), the heuristic thus is $f = h$, before any node is expanded we check that it is not in the previously expanded nodes list, that is done by saving a list of expanded nodes.

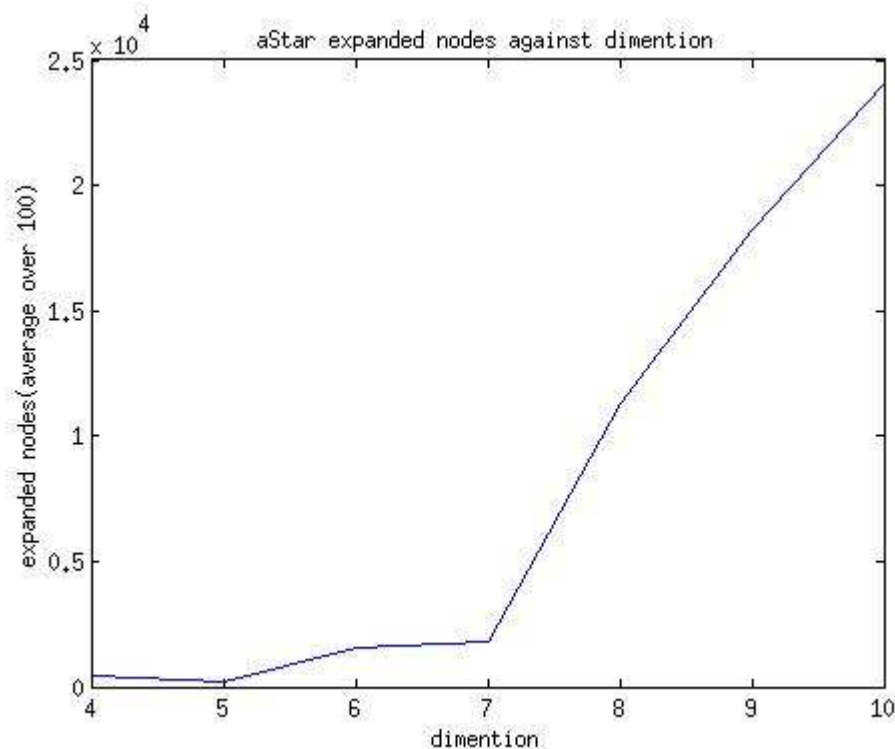
A* -- Systematic Heuristic Search

In A* we run the same algorithm like in BFS, with one difference, the heuristic function now is the cost from the initial node to the current node in addition to the estimate cost from current to the goal, that is $f=g+h$, where g is the cost till now and h is the estimated cost from now till the goal.

Analysis

A*

First I will explore the average number of expanded nodes against the dimension of the board, thus, for each dimension I run the A* 100 times and sum the total number of expanded nodes then take the average over 100, the graph I got is below :

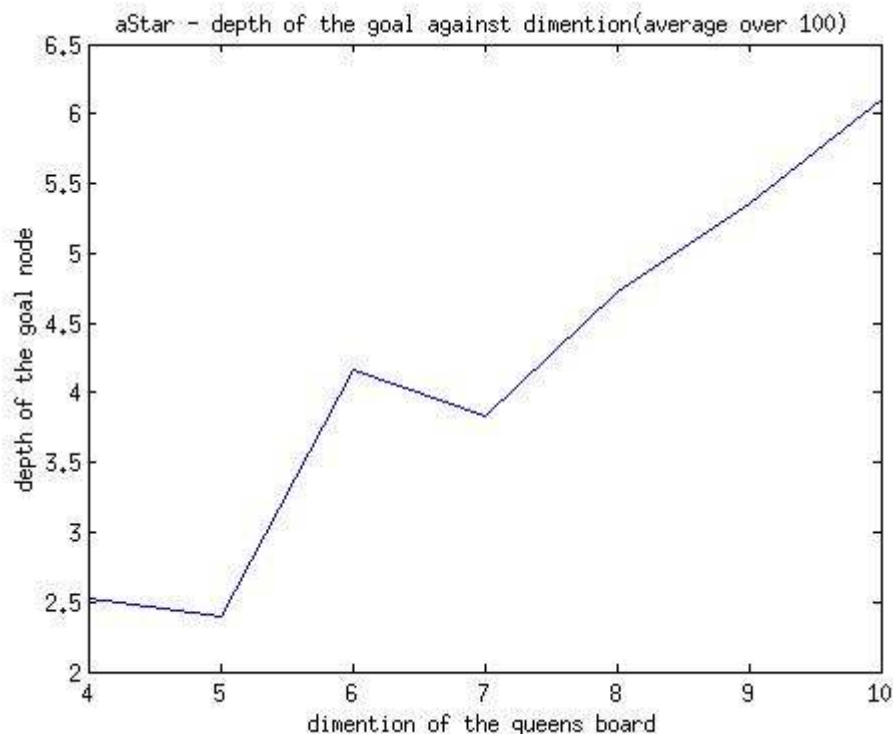


I notice that the expanded nodes number is increasing with the increase of the board dimension (number of queens), it is also possible to notice that from dimension 7 the expanded node number increases linearly.

I plotted until dimension 10 since for larger dimensions, for 100 times the run time is quite large.

Second I will examine the average depth of the founded goal node in the search tree against the dimension of the board, that is, for each dimension I run A* 100 times and sum all of the founded goal node depths, then will average it over 100, and will plot it against the dimension.

The graph I got is in the figure bellow:

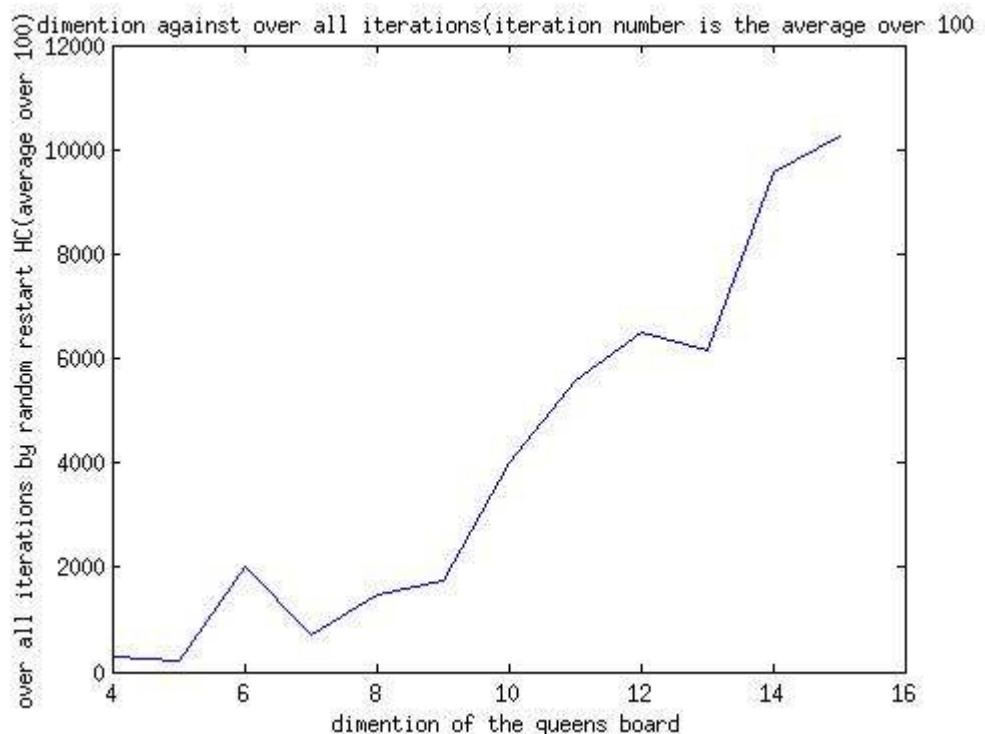


I notice that the average node depth is increasing with the increase of the board dimension, and again from dimension 7 it increases linearly, and due to run time issue I take until dimension 10.

I tried to do the same for BFS algorithm, but I get to conclusion that when running BFS 100 times with dimension equal or larger than 6, it will face one initial state for which I will get an out of memory exception.

Random Restart Hill Climbing

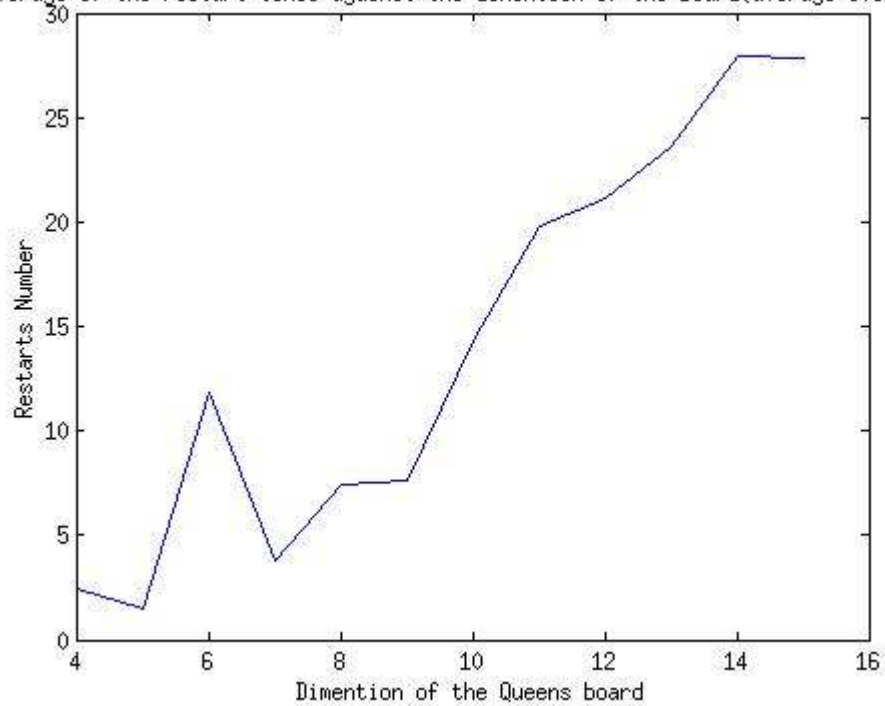
First I explored the average over all iterations number against the dimension of the board, average is taken over 100, that is I run RRHC 100 times for each dimension and take average over 100, I took the dimension from 4 to 15, the graph I got is below :



We can see that the over all behavior of the graph is that the over all iterations average increases with the increase of the dimension of the board, it starts with few hundreds for dimension 4, and it reaches about 12000 iterations for dimension 15 .

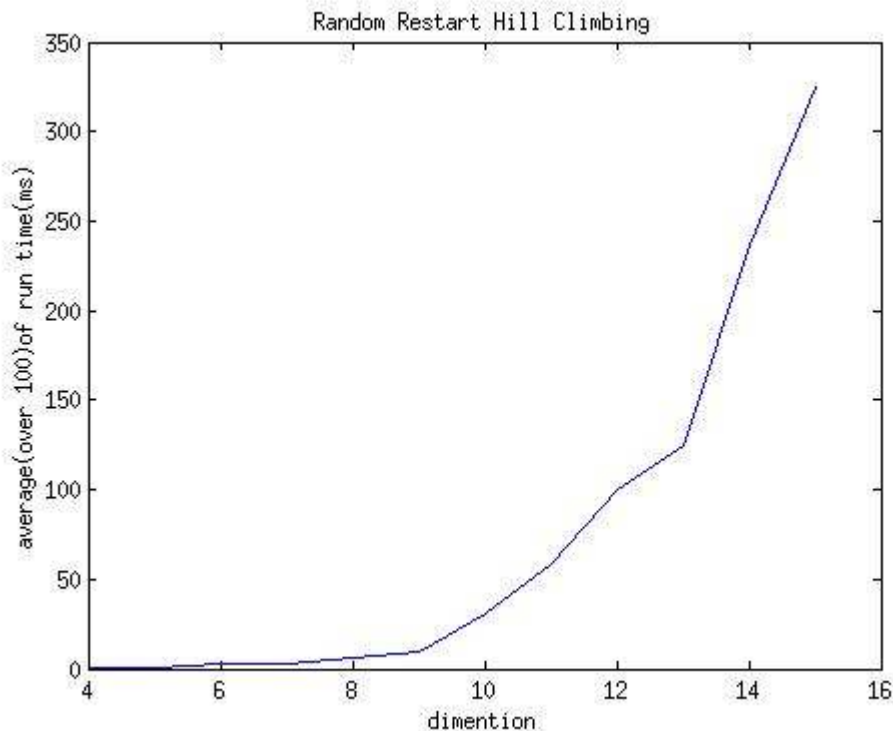
Second I will show the relation between average restarts number of RRHC against the dimension, for each dimension I run RRHC 100 times I sum the over all restarts then take average over 100, the graph that show the relation is plotted below :

average of the restart times against the dimension of the board(average over 100)



It is easily seen that the average restart times is increasing with the increase of the queens number, for four queens it's about 2.5 and it rise up to 28 for 15 queens.

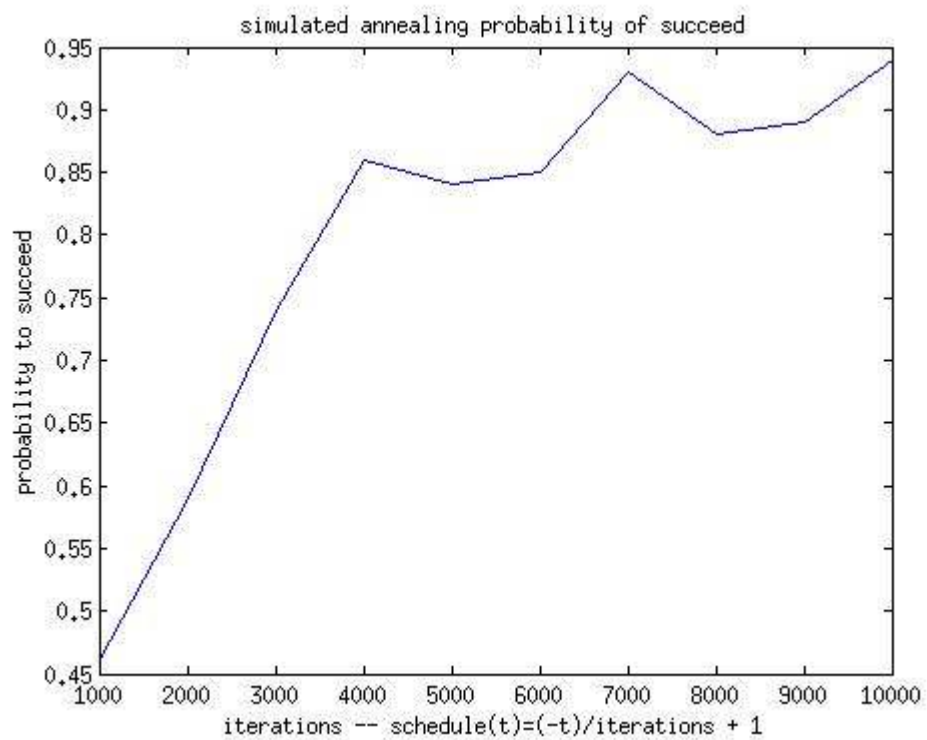
Third I examine the average run time of RRHC against the dimension of the queens board, again, for each dimension from 4 to 15, I run the algorithm 100 times and sum the over all run time of the Random Restart Hill Climbs get the average by dividing by 100 and plot the relation against the dimension, the graph is bellow:



We see that the average run time increases exponentially with the dimension of the queens board, it's near zero (ms) for dimension 4 and about 350 (ms) for dimension 15.

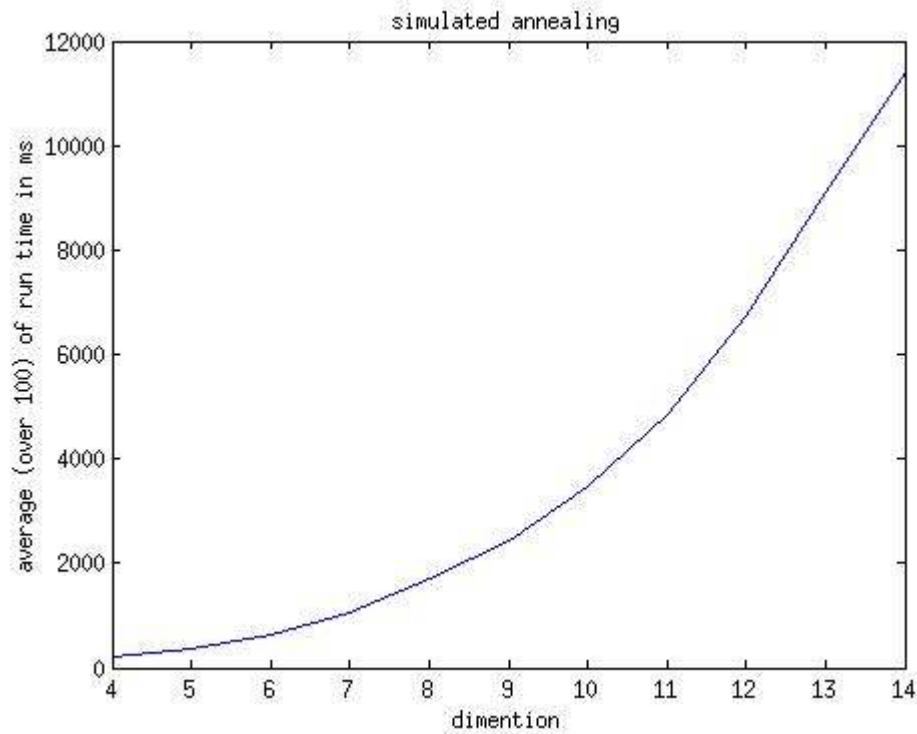
Simulated Annealing

The algorithm of simulated annealing run a fixed number of iterations, which determined by the schedule(t) function, which in our case is:
 $T(t) = (-1)/\text{iterationsNumber} + 1$, thus, first I will examine the probability of finding a solution against the iterations number, for each iterations number that runs from 1000 to 10000, I will run the algorithm 100 times, sum the successful times and get the average over 100, which is actually the probability to succeed, the graph I got from the experiment is given bellow:



It is possible to see as we expect and know that the probability reaches 1 for iterations number that is getting closer to 10 000.

Second and last I will explore the average run time of simulated annealing against the dimension of the board, I expect it to increase with the increase of the board dimension, the graph I got is given bellow :

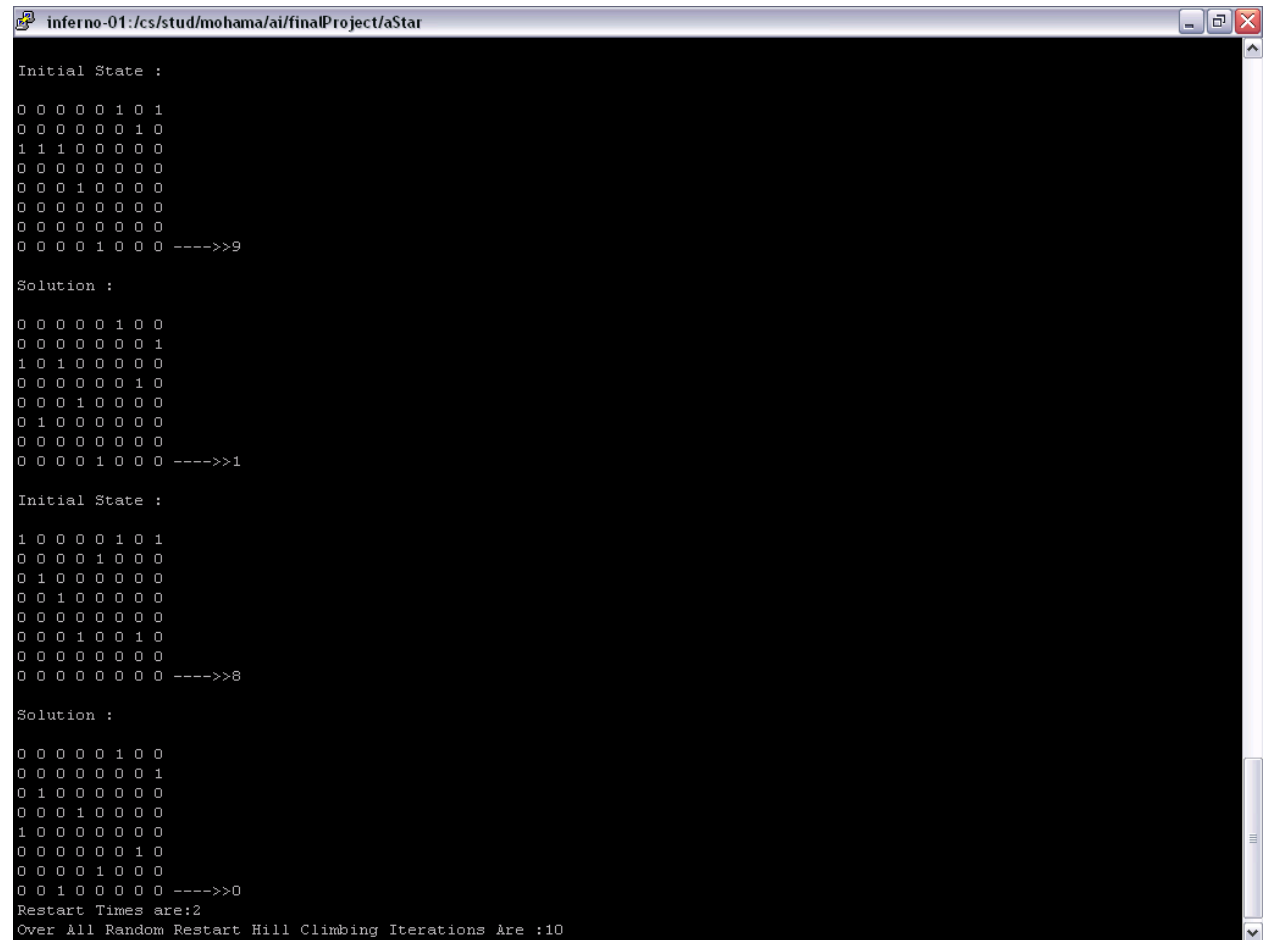


The graph shows clearly that the average run time of simulated annealing is increasing with the increase of the board dimension.

Screen Shots

Bellow I will provide some screen shots of the different algorithms for dimension 8:

For the command : # java NQueens 8 -rrhc



```
inferno-01:/cs/stud/mohama/ai/finalProject/aStar

Initial State :

0 0 0 0 0 1 0 1
0 0 0 0 0 0 1 0
1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 ---->>9

Solution :

0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
1 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0
0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 ---->>1


Initial State :

1 0 0 0 0 1 0 1
0 0 0 0 1 0 0 0
0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 ---->>8

Solution :

0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
1 0 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
Restart Times are:2
Over All Random Restart Hill Climbing Iterations Are :10
```

for simulated annealing we got as an example :
java NQueens 8 -sa



```
inferno-01:/cs/stud/mohama/ai/finalProject/aStar
<14|0>mohama@inferno-01:~/ai/finalProject/aStar> java NQueens 8 -sa

Initial State :

0 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 1
0 1 0 0 0 0 0 0 ---->5

Solution :

0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 1 0 0 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
Over All Iterations Are :10000
<15|0>mohama@inferno-01:~/ai/finalProject/aStar> █
```

For A* I got the following initial and goal states and relevant data:
java NQueensStar 8 -as

```
inferno-01:/cs/stud/mohama/ai/finalProject/aStar
<32|0>mohama@inferno-01:~/ai/finalProject/aStar> java NQueensStar 8 -as
Initial State :

0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1
1 0 0 0 0 0 1 0 ---->>7

Solution :

0 0 0 0 1 0 0 0
0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 1
0 0 1 0 0 0 0 0 ---->>0
Number Of Expanded Nodes Is:1456
Goal Node Depth Is:6
<33|0>mohama@inferno-01:~/ai/finalProject/aStar>
```

And the last one is for BFS:
java NQueensStar 8 -bfs

```
inferno-01:/cs/stud/mohama/ai/finalProject/aStar
<38|0>mohama@inferno-01:~/ai/finalProject/aStar> java NQueensStar 8 -bfs
Initial State :

0 0 0 0 0 0 0 0
0 1 1 0 0 1 0 0
0 0 0 0 0 0 0 1
1 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0 ---->>9

Solution :

0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 0 0 1 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 0
0 0 0 0 0 1 0 0 ---->>0
Number Of Expanded Nodes Is:1120
Goal Node Depth Is:18
<39|0>mohama@inferno-01:~/ai/finalProject/aStar> █
```

