Implementation and experiments of N-QUeen problems

Yoon Jin Park (s5084207),

Bachelor of Computer Science,

Griffith University (Gold Coast Campus)

17/03/2019

Contents

[Introduction 1](#_Toc4491776)

[Experiments 1](#_Toc4491777)

[Breadth first search 1](#_Toc4491778)

[Count and list of solutions for a wide range of N 1](#_Toc4491779)

[Suggestions for pruning and Time analysis for a wide range of N 5](#_Toc4491780)

[Prediction of the time to be taken for N = 30 6](#_Toc4491781)

[Hill-climbing search and Simulated annealing 8](#_Toc4491782)

[Heuristic cost function used for the two local search algorithms 8](#_Toc4491783)

[List of the solutions 8](#_Toc4491784)

[Analysis and comparisons of the two local search algorithms 11](#_Toc4491785)

[Conclusion 16](#_Toc4491786)

[Appendix A. Breadth First Search Solution example 17](#_Toc4491787)

[Appendix B. Hill climbing search Solution Example 19](#_Toc4491788)

[Appendix C. Simulated annealing Solution Example 20](#_Toc4491789)

[Reference 21](#_Toc4491790)

# Introduction

N-queen problem is a classical combinatorial problem where N number of queens are placed on a N x N board with no queens attacking one another (i.e. no queens are placed on the same row, column or diagonal line) (Martinjak and Golub,2007). The purpose of this report is to document the implementation methods and results of the experiments undertaken for three algorithms for N-queen problem. The three algorithms implemented and examined include Breadth first search, Hill-climbing search, and Simulated annealing.

# Experiments

## Breadth first search

Breadth First Search (hereafter, BFS) expands the root node, followed by expansion of all the nodes at the next level, then their child nodes, and so on (Russell and Norvig, 2011). That is, BFS expands the shallowest unexpanded node first prior to exploring their child nodes. Hence, it is justifiable to implement BFS in a First-in-first-out (FIFO) queue as the older (shallower, first-in) nodes are expanded first than the newer nodes.

### Count and list of solutions for a wide range of N

To explore a set of characteristics of BFS, the base model adopts the expansion method of “one queen at a time any place on a board other than the places that are occupied”.

Due to the limitation in memory, the main data structure for ‘frontier’ has been thought carefully and it was decided to consider the states of queens’ positions only, rather than creating a board state each time. Initially, two coordinate positions for each queen is stored in a tuple and appended to a FIFO queue. However, the experiments revealed that the larger problem is associated with the memory management. Therefore, the decision to modify a data structure into a single integer for each queen’s position (appended to a deque) is implemented as a result of the experiments. This proved to be making a minor improvement on the memory issue. However, the computation involved to check whether the state is a goal state is slightly more involved, evidenced by the increased running time.

The total number of solutions found through BFS with or without pruning for each n-queen problem are listed in Table 1 below. All BFS solutions up to n = 6 are listed in the table while some examples of the BFS solutions are included between n = 7 to 20. The experiments had to stop because of the time taken and the limited memory, especially where n is equivalent to 12. Visualisation of the example solution is attached in Appendix A.

Table 1. Number and list of BFS solutions for each n-queen problem

| **N queens** | **Number of solutions** | **Solutions** |
| --- | --- | --- |
| 1 | 1 | [0] |
| 2 | - | None |
| 3 | - | None |
| 4 | 2 | [1, 7, 8, 14], [2, 4, 11, 13] |
| 5 | 10 | [0, 7, 14, 16, 23], [0, 8, 11, 19, 22], [1, 8, 10, 17, 24], [1, 9, 12, 15, 23], [2, 5, 13, 16, 24], [2, 9, 11, 18, 20], [3, 5, 12, 19, 21], [3, 6, 14, 17, 20], [4, 6, 13, 15, 22], [4, 7, 10, 18, 21] |
| 6 | 4 | [1, 9, 17, 18, 26, 34], [2, 11, 13, 22, 24, 33], [3, 6, 16, 19, 29, 32], [4, 8, 12, 23, 27, 31] |
| 7 | 40 | [0, 9, 18, 27, 29, 38, 47], [0, 10, 20, 23, 33, 36, 46], [0, 11, 15, 26, 30, 41, 45], [0, 12, 17, 22, 34, 39, 44], [1, 10, 14, 27, 32, 37, 47], [1, 10, 19, 21, 30, 39, 48], [1, 11, 14, 24, 34, 37, 47], [1, 11, 16, 21, 34, 38, 47], [1, 11, 20, 24, 28, 37, 47], [1, 12, 16, 27, 31, 35, 46], [1, 13, 18, 23, 28, 40, 45], [2, 7, 19, 22, 32, 41, 45], [2, 7, 19, 24, 29, 41, 46], [2, 11, 20, 22, 31, 40, 42], [2, 12, 15, 25, 28, 38, 48], [2, 13, 15, 24, 33, 35, 46], [2, 13, 17, 21, 32, 36, 47], [3, 7, 16, 26, 29, 41, 46], [3, 7, 18, 22, 33, 37, 48], [3, 8, 20, 25, 30, 35, 47], [3, 12, 14, 23, 32, 41, 43], [3, 13, 16, 26, 29, 39, 42], [3, 13, 18, 22, 33, 35, 44], [4, 7, 17, 27, 30, 40, 43], [4, 7, 19, 24, 29, 41, 44], [4, 8, 19, 23, 34, 38, 42], [4, 9, 14, 26, 31, 36, 48], [4, 13, 15, 24, 33, 35, 44], [4, 13, 15, 26, 30, 35, 45], [5, 7, 16, 25, 34, 36, 45], [5, 8, 18, 21, 31, 41, 44], [5, 9, 14, 24, 34, 39, 43], [5, 9, 18, 27, 28, 38, 43], [5, 9, 20, 24, 28, 39, 43], [5, 10, 15, 27, 32, 37, 42], [5, 10, 20, 21, 30, 39, 43], [6, 8, 17, 26, 28, 37, 46], [6, 9, 19, 22, 32, 35, 45], [6, 10, 14, 25, 29, 40, 44], [6, 11, 16, 21, 33, 38, 43] |
| 8 | 92 | [0, 12, 23, 29, 34, 46, 49, 59], [1, 13, 23, 26, 32, 43, 54, 60], [2, 12, 22, 24, 35, 41, 55, 61], [3, 9, 22, 28, 32, 47, 53, 58], [4, 10, 16, 30, 33, 47, 53, 59], [5, 10, 20, 30, 32, 43, 49, 63], [6, 9, 21, 26, 32, 43, 55, 60], [7, 9, 19, 24, 38, 44, 50, 61] |
| 9 | 352 | [0, 11, 23, 34, 37, 48, 62, 69, 76], [1, 13, 24, 30, 36, 47, 62, 68, 79], [2, 14, 19, 35, 40, 45, 61, 66, 78], [3, 14, 20, 35, 37, 52, 58, 69, 72], [4, 10, 23, 35, 42, 48, 54, 65, 79], [5, 12, 18, 33, 44, 46, 61, 67, 74], [6, 9, 21, 34, 40, 47, 62, 68, 73], [7, 9, 21, 32, 38, 53, 60, 67, 73], [8, 11, 22, 28, 43, 45, 60, 66, 77] |
| 10 | 724 | [0, 12, 25, 37, 49, 54, 68, 71, 83, 96], [1, 19, 22, 36, 48, 53, 60, 74, 87, 95], [2, 18, 21, 39, 44, 56, 60, 73, 85, 97], [3, 17, 24, 32, 40, 59, 66, 78, 85, 91], [4, 17, 21, 36, 49, 52, 60, 78, 83, 95], [5, 13, 26, 39, 42, 58, 61, 74, 87, 90], [6, 14, 20, 37, 45, 52, 68, 71, 83, 99], [7, 15, 21, 36, 44, 50, 68, 73, 89, 92], [8, 13, 25, 37, 41, 56, 60, 72, 84, 99], [9, 14, 26, 33, 40, 57, 61, 78, 85, 92], |
| 11 | 2,680 | [0, 13, 26, 39, 52, 65, 67, 80, 93, 106, 119], [1, 14, 29, 43, 48, 64, 71, 77, 90, 105, 118], [2, 18, 32, 34, 53, 55, 71, 80, 96, 105, 114], [3, 16, 24, 41, 54, 62, 70, 78, 97, 99, 116], [4, 13, 30, 43, 51, 56, 69, 77, 94, 108, 115], [5, 19, 22, 40, 48, 57, 75, 83, 98, 102, 111], [6, 14, 22, 35, 49, 63, 76, 78, 92, 108, 117], [7, 16, 23, 42, 50, 55, 68, 81, 96, 109, 113], [8, 12, 31, 35, 49, 65, 66, 80, 94, 103, 117], [9, 18, 27, 36, 45, 65, 74, 83, 92, 101, 110], [10, 19, 28, 37, 46, 55, 75, 84, 93, 102, 111] |
| 12 | Not available | [9, 15, 30, 36, 55, 71, 73, 89, 98, 116, 130, 136], [6, 22, 26, 36, 53, 68, 73, 95, 100, 115, 129, 135], [3, 23, 31, 37, 52, 69, 72, 92, 101, 110, 126, 142], [9, 19, 27, 46, 48, 62, 77, 95, 104, 109, 124, 138] |
| 13 | Not available | [10, 18, 35, 45, 55, 65, 80, 103, 112, 128, 137, 147, 157], [5, 22, 26, 42, 60, 77, 85, 95, 105, 128, 136, 153, 158], [5, 15, 35, 40, 56, 76, 78, 97, 114, 129, 137, 146, 164], [1, 16, 37, 48, 59, 69, 88, 91, 106, 122, 138, 149, 168] |
| 14 | Not available | [9, 17, 29, 49, 67, 78, 84, 102, 125, 131, 142, 166, 178, 188], [7, 18, 39, 43, 62, 83, 86, 106, 115, 126, 149, 159, 180, 192], [0, 22, 33, 54, 58, 76, 95, 101, 122, 139, 149, 155, 172, 189], [9, 18, 41, 50, 62, 71, 96, 103, 112, 136, 147, 156, 179, 185] |
| 15 | Not available | [8, 16, 37, 47, 74, 87, 94, 114, 125, 148, 161, 165, 183, 201, 220], [4, 17, 38, 48, 69, 88, 95, 119, 131, 142, 162, 171, 180, 205, 211], [9, 21, 43, 47, 68, 76, 97, 109, 120, 147, 160, 170, 194, 206, 213], [6, 26, 32, 45, 63, 85, 103, 114, 132, 139, 151, 172, 185, 209, 218] |
| 16 | Not available | [11, 29, 37, 60, 73, 83, 96, 114, 142, 145, 167, 186, 200, 214, 228, 255], [12, 19, 41, 55, 66, 93, 97, 118, 136, 159, 171, 190, 197, 218, 224, 244], [3, 29, 43, 57, 68, 95, 97, 120, 130, 158, 166, 176, 202, 213, 231, 252], [11, 17, 36, 54, 67, 90, 108, 127, 130, 151, 174, 184, 205, 208, 233, 245] |
| 17 | Not available | [3, 29, 42, 56, 81, 94, 103, 123, 151, 153, 180, 203, 210, 223, 245, 266, 286], [13, 27, 38, 66, 76, 90, 105, 119, 150, 160, 181, 193, 206, 237, 239, 264, 284], [8, 19, 39, 61, 74, 100, 114, 135, 139, 166, 177, 188, 208, 232, 252, 255, 281], [9, 20, 46, 59, 83, 90, 104, 130, 152, 159, 183, 188, 214, 221, 245, 269, 276] |
| 18 | Not available | [16, 32, 43, 59, 73, 96, 121, 126, 161, 174, 182, 209, 224, 238, 261, 273, 298, 321], [17, 24, 45, 54, 77, 105, 120, 133, 160, 175, 182, 208, 230, 237, 253, 274, 299, 314], [1, 29, 45, 57, 85, 94, 110, 141, 156, 168, 188, 215, 230, 250, 257, 280, 288, 313], [1, 21, 41, 62, 86, 101, 125, 133, 160, 166, 180, 213, 226, 236, 258, 282, 297, 319] |
| 19 | Not available | [4, 27, 48, 62, 89, 111, 132, 133, 159, 185, 191, 226, 230, 256, 272, 288, 315, 338, 354], [7, 34, 46, 68, 85, 113, 114, 135, 164, 177, 191, 223, 233, 260, 270, 302, 307, 339, 352], [13, 27, 43, 69, 77, 98, 132, 144, 156, 181, 206, 215, 230, 262, 273, 299, 304, 340, 351], [14, 24, 46, 57, 89, 101, 130, 145, 162, 175, 207, 210, 239, 254, 268, 303, 313, 338, 345] |
| 20 | Not available | [16, 23, 55, 67, 85, 110, 138, 154, 166, 184, 219, 231, 240, 272, 297, 308, 322, 349, 361, 393], [3, 25, 58, 61, 92, 117, 131, 156, 167, 189, 200, 222, 254, 279, 295, 304, 328, 350, 373, 386], [10, 28, 53, 65, 81, 114, 131, 159, 176, 183, 206, 220, 258, 272, 295, 307, 322, 344, 369, 397], [10, 26, 41, 79, 96, 113, 127, 144, 175, 192, 214, 237, 240, 263, 285, 302, 338, 349, 371, 388] |

### Suggestions for pruning and Time analysis for a wide range of N

Two methods to alleviate the issues associated with time and memory have been implemented: 1) row separation; and 2) column check pruning. Row separation places a queen on each row so that queens do not have a conflict at the row level. Column check is to check whether queens are in conflict at the column level. After checking whether queens are safe in column-wise, if any queen is placed in a not safe position, the set of these child nodes are not expanded further.

Without pruning, the state space considered is . However, placing a queen on each row reduces the state space to if the solution exists at the bottom level. For example, for the 4-queen problem, there are 1,820 states without pruning, which decreases to 321 after the row separation. This enables time reduction and BFS running with the limited computational resources. The comparison of running time between three different BFS models is listed in Table 2.

Table 2. Comparison of running time between base model and models with pruning

|  |  |  |  |
| --- | --- | --- | --- |
| **N queens** | **Base model** | **+Row separation** | **+Row separation & Column check** |
| 1 | 0.000401 | 0.000000 | 0.000000 |
| 2 | - | - | - |
| 3 | - | - | - |
| 4 | 14.808859 | 0.000000 | 0.000000 |
| 5 | - | 0.009023 | 0.003004 |
| 6 | - | 0.129173 | 0.026025 |
| 7 | - | 2.714385 | 0.232299 |
| 8 | - | 59.446083 | 2.455049 |
| 9 | - | - | 28.986122 |
| 10 | - | - | 356.476245 |
| 11 | - | - | 4574.794219 |

The base model without pruning takes a long time after N is larger than and equal to 5. With the row separation, N could go up to 8 only, due to a memory error raised. However, placing a queen after the column check with the row separation enables N up to 11. This is because the model with the pruning method does not expand a certain collection of nodes.

### Prediction of the time to be taken for N = 30

Based on the model with the row separation and column check, the time to be taken for n = 30 is predicted with the linear and polynomial regression and .

The scatter graph of the pruned model is as follows. The graph displays that it appears to be more closed to the exponential model.

Figure 1. Scatter plot for N between 1 to 11

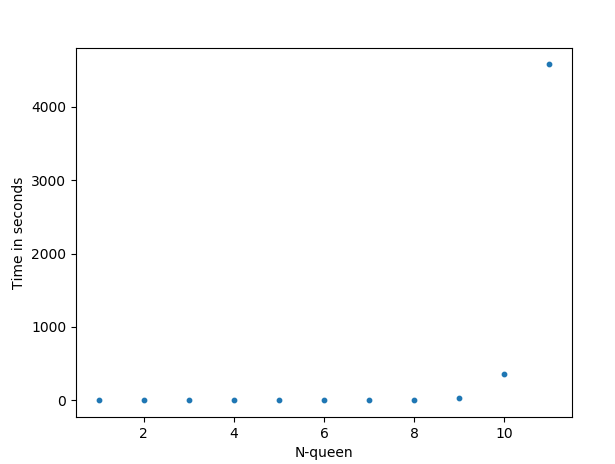
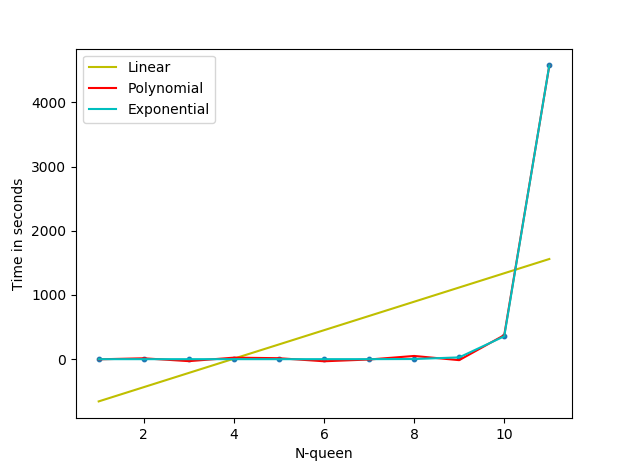


Figure 2. Comparison between the three different models



The linear regression line (yellow line) does not fit well, evidenced by a loss function of the root mean squared error (RMSE), 1104.069713, and the root squared (R2, normalised version of MSE), 0.287435. Using the linear regression, the time to be taken for solving n = 30 is estimated as 5773.06 seconds (1 hour 36 minutes 22 seconds), which is totally underestimated.

With a sufficiently high degree (for example, 7), the polynomial regression line (red line) gets close to the scattered plot. This provides RSME of 26.274152 and the R2 of 0.999596. The polynomial regression predicts the time to be taken for solving n = 30 as 4.80727456E+08 seconds.

Since the scatter plot appears to be exponential, the best fit would be an exponential form (cyan line) where the RSME of 0.343625 and the R2 score of 1.000000. The exponential formula predicts that the time to be taken for solving n equivalent to 30 as 5.228104831036341E+24 seconds. I believe this time will be more accurate than the other two predictions due to its exponential nature shown during the experiments, though there is a possibility that this exponential line may represent an overfitting pattern.

## Hill-climbing search and Simulated annealing

Two local search algorithms are adopted, namely Hill-climbing search (HC) and Simulated annealing (SA). HC is a local search algorithm using heuristics to guide the search. Only the best neighbour is selected for searching for a solution. SA is also a local search but combines HC and random walk to take the best of both worlds. By using randomness, SA aims to escape from a local optimum (Martinjak and Golub, 2007), which HC tends to get stuck in.

### Heuristic cost function used for the two local search algorithms

The heuristic cost function used for both search algorithm is the number of queens in conflict. One queen can be in conflict with other queens whilst at the same time each queen can be in conflict with the other queen multiple times. This implies that the cost function can have the number of conflicts greater than the number of queens, and the objective of the local search is to find a local or global minimum with the number of conflicts lesser than the initial state. The global optimal solution will have zero number of queens in conflict.

### List of the solutions

The local minimum or the solution found is visualised to verify whether it is correct. The list of solutions found is shown in Table 4 below. Where N=1,000, it takes a long time, hence, the experiment is not complete. Examples of the visualisation is displayed in Appendix B for HC and Appendix C for SA.

Table 4. List of solutions found by HC and SA

|  |  |  |
| --- | --- | --- |
| **N queens** | **HC Solution** | **SM Solution** |
| 1 | [0] | [0] |
| 2 | None | None |
| 3 | None | None |
| 4 | [2, 4, 11, 13] | [2, 4, 11, 13] |
| 5 | [1, 9, 12, 15, 23] | [2, 5, 13, 16, 24] |
| 6 | [1, 9, 17, 18, 26, 34] | [2, 11, 13, 22, 24, 33] |
| 7 | [4, 13, 15, 24, 33, 35, 44] | [5, 9, 20, 24, 28, 39, 43] |
| 8 | [1, 11, 21, 31, 34, 40, 54, 60] | [5, 10, 16, 31, 35, 41, 54, 60] |
| 9 | [1, 15, 22, 27, 44, 48, 59, 70, 74] | [6, 12, 19, 35, 41, 47, 58, 70, 72] |
| 10 | - | [4, 11, 29, 36, 43, 50, 62, 78, 85, 97] |
| 11 | - | [6, 21, 25, 33, 48, 63, 67, 86, 95, 104, 112] |
| 12 | - | [3, 17, 31, 46, 48, 62, 76, 85, 104, 119, 129, 138] |
| 13 | - | [6, 25, 36, 43, 52, 70, 86, 92, 115, 119, 137, 152, 159] |
| 14 | - | [7, 19, 28, 55, 66, 76, 87, 99, 123, 135, 142, 158, 176, 194] |
| 15 | - | [11, 23, 35, 47, 74, 84, 97, 117, 120, 138, 156, 178, 190, 199, 211] |
| 16 | - | [11, 25, 35, 53, 72, 95, 108, 112, 141, 148, 166, 190, 194, 218, 225, 247] |
| 17 | - | [4, 25, 34, 56, 83, 98, 111, 125, 150, 155, 181, 203, 216, 222, 241, 265, 279] |
| 18 | - | [4, 32, 41, 63, 85, 105, 108, 129, 150, 172, 181, 209, 233, 246, 268, 278, 290, 313] |
| 19 | - | [9, 24, 52, 58, 87, 108, 122, 151, 152, 174, 206, 211, 243, 257, 273, 289, 316, 340, 348] |
| 20 | - | [17, 30, 54, 71, 82, 115, 126, 141, 173, 180, 209, 224, 259, 276, 298, 303, 332, 347, 365, 388] |
| 30 | - | [25, 31, 86, 95, 127, 171, 192, 212, 246, 280, 320, 346, 368, 394, 449, 469, 504, 538, 543, 588, 627, 644, 669, 705, 720, 761, 803, 827, 862, 883] |
| 50 | - | [31, 72, 128, 154, 239, 264, 321, 391, 412, 465, 518, 586, 629, 699, 737, 761, 809, 866, 900, 976, 1033, 1051, 1117, 1184, 1208, 1260, 1342, 1395, 1447, 1473, 1535, 1555, 1644, 1675, 1720, 1752, 1846, 1853, 1940, 1957, 2027, 2069, 2132, 2163, 2206, 2274, 2343, 2388, 2448, 2480] |
| 100 | - | - |

### Analysis and comparisons of the two local search algorithms

The main difference between HC and SA seems to lie on the pursuit of perfectionism. HC does not allow any single move deviating from the objective of decreasing the heuristic cost. This search will always take the best neighbour. However, SA accepts move against the objective with certain probability. To find the global optimal solution, SA takes into consideration of the fact that it sometimes needs to make an allowance to take a worse option temporarily.

With this difference, the success rate and the time of finding a solution between HC (modified with a random restart) and SA also varies. Table 5 displays the rate and running time to find a solution for the two search algorithms. 3 trials are made for each n-queen problem up to 100-queen problem.

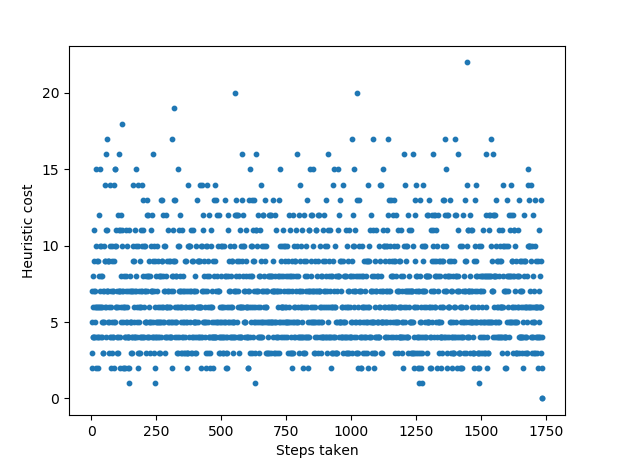
Table 5. Success rate and running time for finding a solution – HC and SA (from N = 12, HC’s results are mostly the local optimum only, therefore these results are omitted from the table)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N queens** | **HC Running time** | **HC Success rate** | **SA Running time** | **SA Success rate** |
| 1 | 0.000401 | 100.00% | 0.000000 | 100.00% |
| 2 | N/A | N/A | N/A | N/A |
| 3 | N/A | N/A | N/A | N/A |
| 4 | 0.000000 | 100.00% | 0.011014 | 100.00% |
| 5 | 0.000998 | 100.00% | 0.009351 | 100.00% |
| 6 | 0.044069 | 100.00% | 0.199590 | 100.00% |
| 7 | 0.029036 | 100.00% | 0.460240 | 100.00% |
| 8 | 0.061077 | 100.00% | 8.228929 | 100.00% |
| 9 | 1.520908 | 100.00% | 12.492247 | 100.00% |
| 10 | 38.430970 | 100.00% | 15.484255 | 100.00% |
| 11 | 121.706899 | 33.33% | 18.222058 | 100.00% |
| 12 | - | - | 20.492234 | 100.00% |
| 13 | - | - | 22.042835 | 100.00% |
| 14 | - | - | 24.141792 | 100.00% |
| 15 | - | - | 26.624548 | 100.00% |
| 16 | - | - | 28.531592 | 100.00% |
| 17 | - | - | 30.266089 | 100.00% |
| 18 | - | - | 32.101048 | 100.00% |
| 19 | - | - | 38.295442 | 100.00% |
| 20 | - | - | 37.789154 | 100.00% |
| 30 | - | - | 70.122811 | 100.00% |
| 50 | - | - | 132.293357 | 100.00% |
| 100 | - | - | - | - |

HC is fast searching for a solution, but it struggles to search for the global optimal solution after N = 12, where HC frequently returns a local optimal solution. However, SA finds a solution stably up to 50 within a relatively short time frame. Where N = 100, it takes too long for both HC and SA searching for the global optimal solution within the time frame/temperature range.

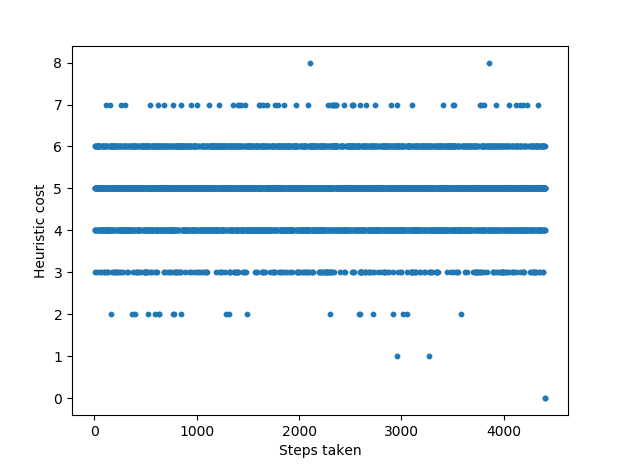
The number of steps to find a solution in HC is not consistent with the number of queens for the problem. It takes 9,089 steps to find a solution for N = 7 while it takes 6,206 steps for N = 8. It seems that whether initial queens have the neighbour with a global optimal solution influences the number of steps taken. HC with N = 10, if it works, goes down towards the minimum heuristic cost fast, as shown in Figure 3 below.

Figure 3. Heuristic cost and steps taken for HC where N = 10



SA tends to provide the optimal solution with a greater effectiveness than HC, but sometimes it could take a greater number of steps to find the solution. In Figure 4, where N = 10, SA takes more steps to find a solution than HC in some cases.

Figure 4. Heuristic cost and steps taken for SA where N = 10



Additional experiments are undertaken. First, as an experiment, HC is modified to incorporate a ‘random restart’ technique. As seen previously, HC often fails to find the global optimal solution because the search move only goes towards one direction, that is, the conformance to the objective. This move would be OK if that direction’s hill contains the global optimal solution, however, it will always fail if the direction does not have the global optimal solution. Therefore, a ‘random restart’ aims to reset the initial point so that it will increase the chance of starting from the right region.

10 runs for 4, 5, 8, and 10 queens are organised to compare results between with and without a random restart. The chance of HC finding a solution without random restart is quite low, average 75%, higher than 14% addressed in the Week 4 lecture note, due to a slightly modified algorithm (i.e. checking whether heuristic cost equals to zero; otherwise, keep looping). From the table below, it is noted that the chance of finding a solution goes up approximately 33.00% by randomly restarting the state after certain number of iterations for the 8-queen problem.

Table 6. Success rate of HC without and with a random restart

|  |  |  |
| --- | --- | --- |
| **N queens** | **Without random restart  Success rate (%)** | **With random restart Success rate (%)** |
| 4 | 100.00% | 100.00% |
| 5 | 100.00% | 100.00% |
| 8 | 66.66% | 100.00% |
| 10 | 33.33% | 100.00% |
|  | **75.00%** | **100.00%** |

Next, SA with a random restart is implemented. Similar to HC, if there is a bad start, SA may take a longer time to converge to the global maximum. Hence, a random restart may alleviate the issues with a bad start. The result revealed that this was not the case. As SA tried to converge through its own cycle, if the random start of the queens is triggered every ten-hanging status, it actually makes the success rate worse. Especially this becomes a case where N gets larger (e.g. N = 10 in Table 7).

Table 7. Success rate of SA without and with a random restart

|  |  |  |
| --- | --- | --- |
| **N queens** | **Without random restart  Success rate (%)** | **With random restart Success rate (%)** |
| 4 | 100.00% | 100.00% |
| 5 | 100.00% | 100.00% |
| 8 | 100.00% | 100.00% |
| 10 | 100.00% | 60.00% |
|  | **100.00%** | **90.00%** |

Thirdly, SA is experimented with a wide range of parameters to find the better temperature-related settings. In particular, the influence of four parameters (starting temperature, temperature decay rate, the number of repeats, and new temperature) are investigated further. According to the recommended SA principle in the Week 4 lecture, it is desirable to make the probability of an acceptance rate as a decreasing exponential form where a worse move gets accepted at the beginning to a greater extent and the probability gets decreased gradually. With five trials, the result of experiments revealed that maintaining this decreasing exponential form is crucial since deviation from this form generated a pre-mature termination prior to finding a solution. In addition, the number of the repeat k has to be at a certain level, otherwise the success rate of SA decreases dramatically.

Table 8. SA’s success rate and running time for finding a solution with different settings

| **Starting temper-ature (temp\_max)** | **Temperature decay (alpha)** | **Repeat (k)** | **New temperature** | **Success rate**  **(n = 8)** | **Time**  **(n = 8)** |
| --- | --- | --- | --- | --- | --- |
| 30,000 | 0.999997 | n\_queen\*n\_queen\*10,000,000,000 | temperature\*temp\_decay\_alpha^(time\_elapsed) | 100% | 11.211368 |
| 3,000,000 | log(temp\_max/temp\_min) | n\_queen\*n\_queen\*10,000,000,000 | temperature \* e^(temp\_decay\_alpha\*current\_repeat/total\_repeat) | 100% | 9.346054 |
| 3,000,000 | log(temp\_max/temp\_min) | n\_queen\*n\_queen\*100,000,000 | temperature \* e^(temp\_decay\_alpha\*current\_repeat/total\_repeat) | 20% | 2.626261 |

The fourth experiment entails HC and SA with two different neighbour selection strategies. To inject more randomness into HC that already has a random restart, random neighbour selection (i.e. purely random selection) and best random neighbour selection are used. HC with the greater randomness in a neighbour selection makes an improvement to the success rate, as noted in Table 9 below. However, it takes a longer time to find a solution because HC still does not take any inferior neighbour.

SA adopts best neighbour selection and best random neighbour selection for the comparison. SA with the best random selection neighbour selection strategies does not seem to work better. It is noted that the success rate drops after introducing the search of ‘best random’ neighbours and it takes a longer time to reach to the global optimal solution. However, the result of the best neighbour selection (not best random) is interesting. When this neighbour selection method finds the solution, it is fast – (0.020041 vs 15.515134), but SA slightly loses its effectiveness by gaining its efficiency.

Table 9. Success rate and running time for finding a solution with different neighbour selection strategies

|  |  |  |  |
| --- | --- | --- | --- |
| **Starting temperature (temp\_max)** | **Best neighbour (n = 10)** | **Random neighbour (n = 10)** | **Best random neighbour (n = 10)** |
| HC | Success rate: 33.33% Time: 121.706899 | Success rate: 66.66% Time: 233.021336 | Success rate: 66.66% Time: 247.141965 |
| SA | Success rate: 66.66% Time: 0.020036 | Success rate: 100% Time: 15.515134 | Success rate: 33.33% Time: 87.073728 |

## Conclusion

In summary, the three different algorithms have different strengths. BFS provides a complete set of available solutions for n-queen problems, but it is difficult to run this algorithm for larger N queens due to the time constraint and the limited memory. HC is fast and works great with a small to medium N-queen problem, but it frequently provides local minimum solutions as N gets larger. Overall, SA seems to be the best algorithm if the goal is to find ‘a’ solution, since it is fast and returns the global optimal solutions most of the time. However, it should be noted that SA does not provide a complete set of solutions.

# Appendix A. Breadth First Search Solution example

|  |  |
| --- | --- |
| N = 1 | N = 2 |
|  | None |
| N = 3 | N = 4 |
| None |  |
| N = 5 | N = 6 |
|  |  |
| N = 7 | N = 8 |
|  |  |

# Appendix B. Hill climbing search Solution Example

|  |  |
| --- | --- |
| N = 1 | N = 2 |
|  | None |
| N = 3 | N = 4 |
| None |  |
| N = 5 | N = 6 |
|  |  |
| N = 7 | N = 8 |
|  |  |

# Appendix C. Simulated annealing Solution Example

|  |  |
| --- | --- |
| N = 1 | N = 2 |
|  | None |
| N = 3 | N = 4 |
| None |  |
| N = 5 | N = 6 |
|  |  |
| N = 7 | N = 8 |
|  |  |

# Reference

Russell, S. J and Norvig, P., Artificial Intelligence: A Modern Approach 3rd Edition, 2011, Pearson.

Martinjak, I. and Golub, M., 2007, Comparison of Heuristic Algorithms for the N-Queen Problem, *International Conference on Information Technology Interfaces*.