CS 534 – Artificial Intelligence

Assignment -1

Group 5:

Mandeep Singh ***Nikunj Parmar***

Prasham Patel ***Purna Patel***

Suketu Parekh

1. Heavy Queens

For the problem, a grid of side *n* was to be solved in the same fashion as a normal n-queens problem with the variation that each queen has a weight which increases the cost of its movement by a factor of the square of the weight.

Two approaches- A star and Greedy search were employed for solving the problem. A heuristic was thought of for the same. The heuristic can be understood as follows:

All of the attacking queens for a particular queen are identified. The number of attacking queens for that queen is found. Then that queen is 'removed' from the board and the same is continued for other queens column-wise.

Mathematically, the heuristic can be expressed as:

$$h = \sum_{k=1}^{n} {w_k}^2 * p$$

where, *n* is the total number of columns in the grid, w_k is the weight of the queen in k^{th} column and *p* is the number of attacks on the queen in k^{th} column.

The selected heuristic is not admissible. The same can be seen with the following example:

Consider a state expressed as the following grid:

		3	
9	**5**		
			4

The estimated cost from the given state to a solution will be given by the heuristic function as:

$$h = 9^2 * 1 = 81$$

But it is apparent that the given state can be solved by moving the queen in the 2nd column down by 2 places. The cost of the same will be $5^2*2 = 50$ which is less than (81) the value given by our heuristic function. Hence the selected heuristic over-estimates the cost and is therefore an **inadmissible** heuristic.

		3	
9	5		
			4
	5		

However, the proposed heuristic is useful as it intuitively gives a sense of how far a given state is from a solution. The form of the heuristic cost function is also on the same lines as the actual cost function.

➢ Analyses:

1. Effective branching factor for the algorithms were calculated as:

$$b = n^{1/d}$$

where, n is the total number of nodes explored and d in the depth till which the search algorithm goes.
The effective branching factor for Greedy and A-star algorithm can be seen in the following table:

Grid Size	Branching Factor	
	A-star	Greedy
6	2.1445661558327243	1.0509452364110587
7	3.1111307422375853	1.185330771982479

2. The solution implemented using the Greedy Search algorithm was able to solve a board of size 33 under 60 seconds. However, the solution consistently failed to solve a board of size 34 under 1 minute time.
However, for A star algorithm, the solution was converged to under 1 second for a board of size 8 and within 4-5 moves. But the same code was unable to perform the required task for a 9-by-9 board in 1 minute.

2. Heavy (9N/8) Queens

For the problem, the same n-queens problem was supposed to be solved with the change that for every 8 cells in the grid, there will be 9 queens; thus, essentially making it a minimum collision problem as there will be no obvious 'solution' to the problem.

One of the approaches employed for solving the problem was the Hill Climbing search. The algorithm was implemented making the use of Simulated Annealing and Random Restarts. Sideways moves were not used as it was thought that there may be many configurations where the calculated cost will be equal. Due to this reasoning, only Simulated Annealing and Random Restarts were used. For Random Restarts, it was made sure that the configuration of the starting grid remains the same.

For the 2nd approach, Genetic Algorithm was implemented. Culling, Elitism and Mutation were all deployed for the solution code.

- Analyses:

1. Hill Climbing: For the implementation of Hill Climbing algorithm, Simulated Annealing and Random Restarts were used. For Simulated Annealing, a cut-off probability value of 0.95 was choses after attempts with multiple values in the code; meaning that the algorithm will start with a Random Restart if the probability value is lower than 0.95. The same can be understood in brief in figure 1. The initial 'temperature' was set to be a constant multiple of the size of the grid hence accounting for that variability.

   $$T_{init} = n * c$$

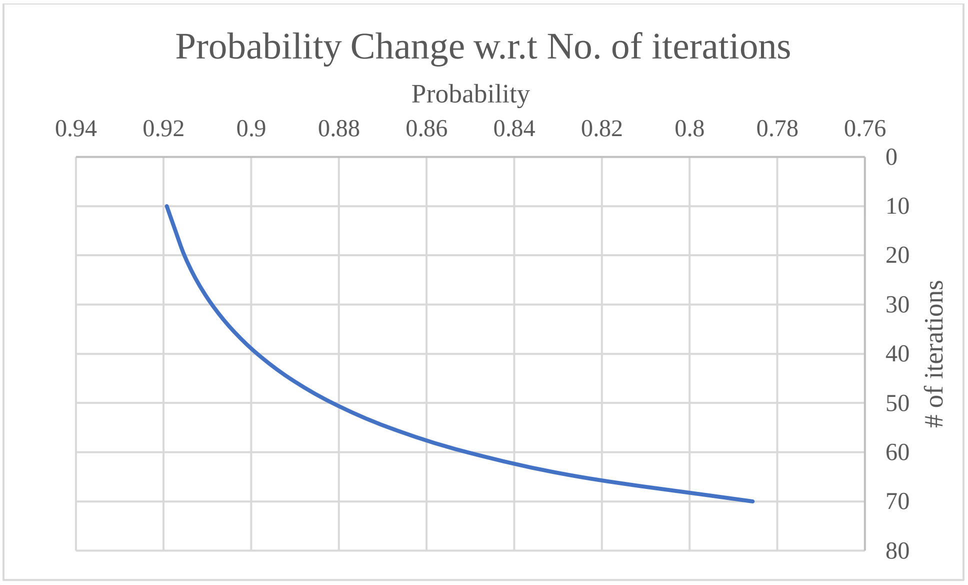
   where, T_{init} is the initial temperature, n is the grid size and c is a constant which is kept as 100 in our case. Temperature decreases with the function:

   $$T_{n+1} = T_n / \log_{1000}(counter+1000)$$

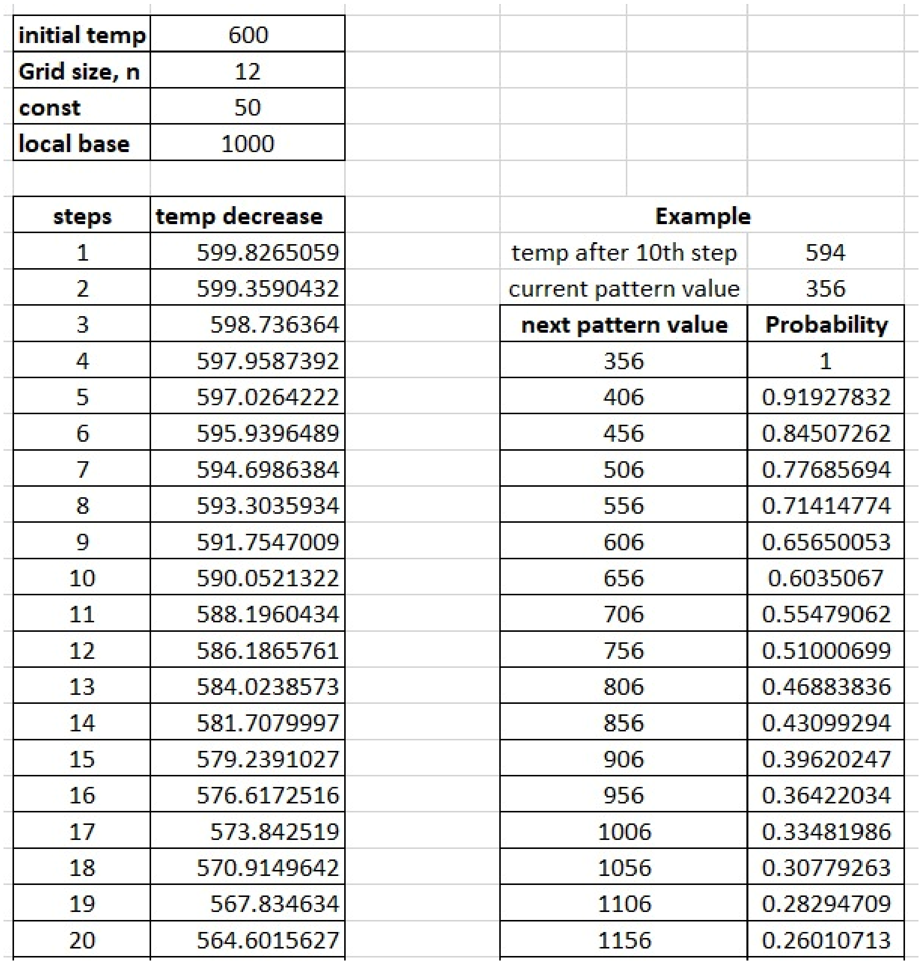
   Probability function is given by:

   $$Probability = e^{((parent\ cost - current\ cost)/temperature)}$$

   Hence, for a given parent cost and current cost :



Genetic Algorithm: For the code, all of Culling, Mutation and Elitism were used. The Culling was set to 30% meaning that the least fit 30% will be removed from generating children. The top 10% of the population were marked as Elites for Elitism and the Mutation chance was set to 20%.



initial temp	600
Grid size, n	12
const	50
local base	1000

steps	**temp decrease**
1	599.8265059
2	599.3590432
3	598.736364
4	597.9587392
5	597.0264222
6	595.9396489
7	594.6986384
8	593.3035934
9	591.7547009
10	590.0521322
11	588.1960434
12	586.1865761
13	584.0238573
14	581.7079997
15	579.2391027
16	576.6172516
17	573.842519
18	570.9149642
19	567.834634
20	564.6015627

Example	
temp after 10th step	594
current pattern value	356
next pattern value	**Probability**
356	1
406	0.91927832
456	0.84507262
506	0.77685694
556	0.71414774
606	0.65650053
656	0.6035067
706	0.55479062
756	0.51000699
806	0.46883836
856	0.43099294
906	0.39620247
956	0.36422034
1006	0.33481986
1056	0.30779263
1106	0.28294709
1156	0.26010713

Fig.1: Instance Calculation for Simulated Annealing

2. Both algorithms- Hill Climbing and Genetic Algorithm were run for a fixed time of 2 minute and the best node (configuration) in terms of minimum cost at every 10 second interval was recorded and plotted here. The starting configuration for both algorithms was same.

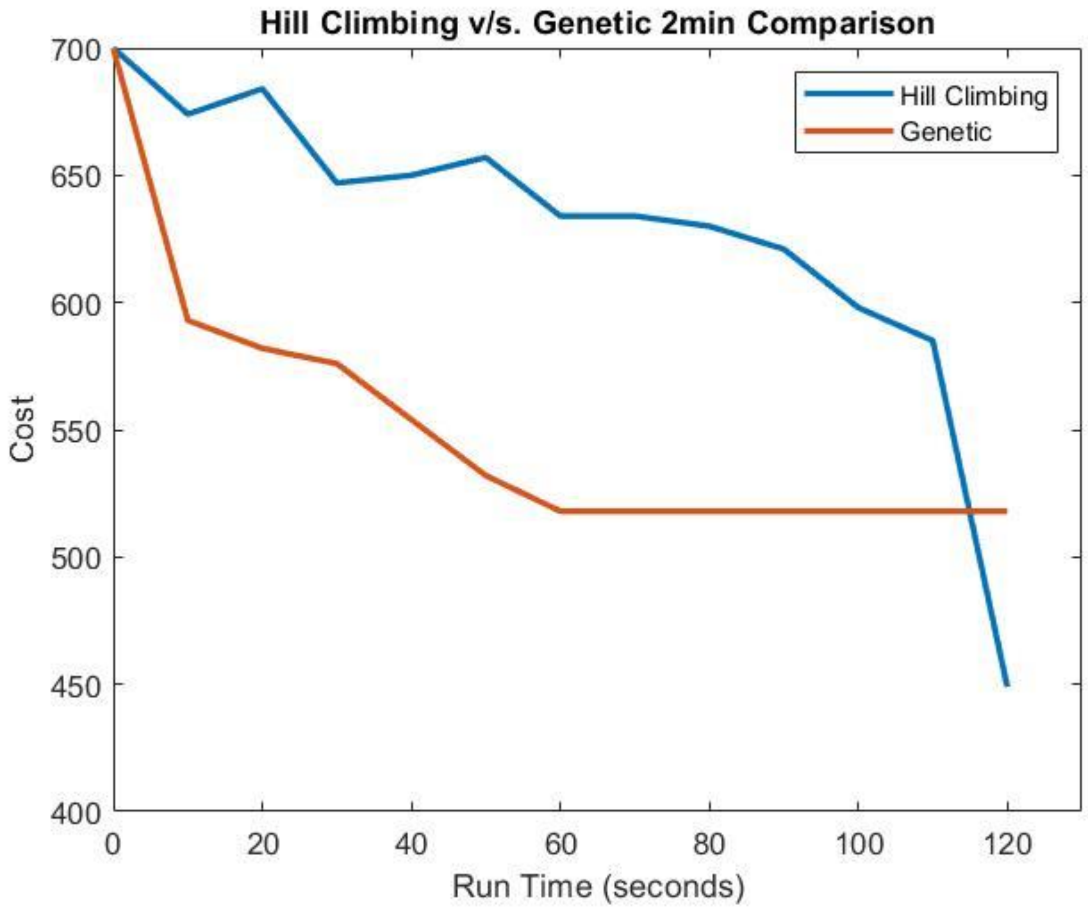


Fig.2: Comparison of Cost between Hill Climbing and Genetic Algorithm

3. For measuring the size of the board for which algorithms can be used to solve in under 20 seconds, the codes were run for **40 seconds** with different grid sizes and the size for which the minimum cost at the end of 40 seconds was less than the same the end of 20 seconds by ~15% were found and the maximum size of the board was reported.
   For the Hill Climbing algorithm, the size of the board came out to be 18-by-18. The cost did not decrease significantly after 20 seconds mark. The same was difficult to check for the Genetic algorithm as the algorithm was observed to settle at a constant cost value in a few iterations regardless of gird size input to the code. But empirically, it can be concluded that the algorithm can easily handle a grid size of 18 under 20 seconds.

4. The performance of the algorithms were measured using the minimum cost of the in-process nodes during the stipulated time intervals. The result of the same can be seen in the following figures.

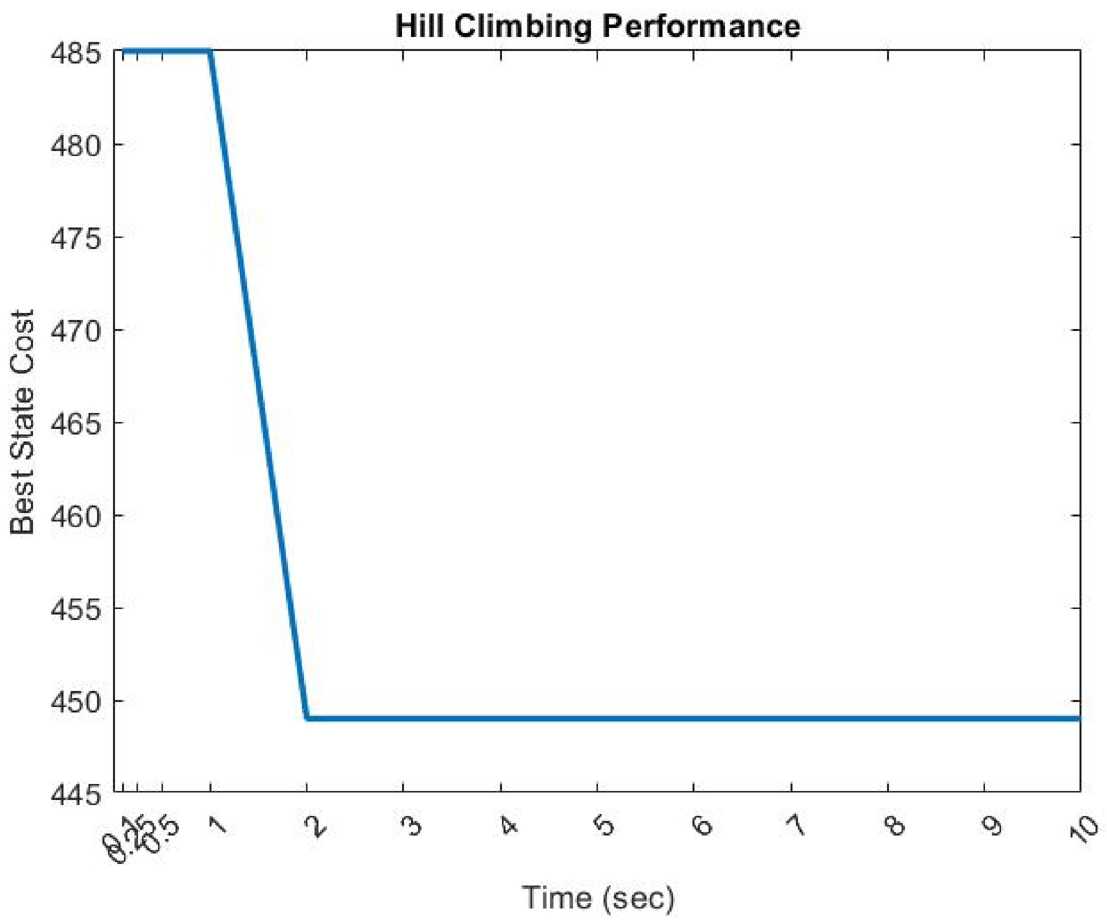


Fig.3: Hill Climbing Performance Measure

It can be concluded that Genetic algorithm performs really good in short period of time (say 0.25 seconds) and the contrary is true for longer periods (say 20 seconds) where Hill Climbing proves to be more effective.

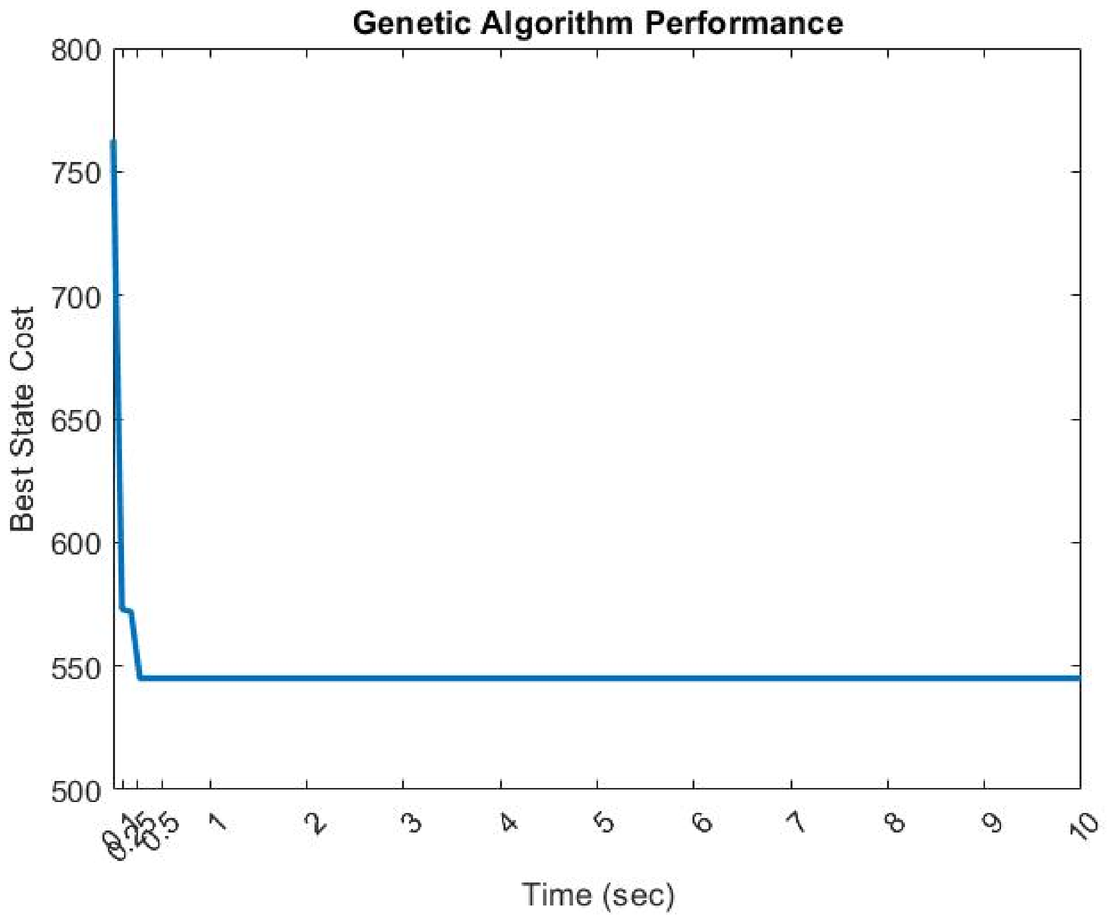


Fig.4: Genetic Algorithm Performance Measure

Reference:

1. Easy A* (star) Pathfinding, Medium [Online] - https://medium.com/@nicholas.w.swift/easy-a-star-pathfinding-7e6689c7f7b2
2. Computing Number of Conflicting Pairs in N-Queen Board in Linear Time and Space Complexity, Medium [Online] – https://towardsdatascience.com/computing-number-of-conflicting-pairs-in-a-n-queen-board-in-linear-time-and-space-complexity-e9554c0e0645
3. Simulated annealing [Online] – https://docs.google.com/spreadsheets/d/1FV31-saEQROBriFcjQ35v5SZ9dAJd10tek01dGPgZfA/edit#gid=392372808