CS7641 Machine Learning
Assignment 2 : Randomized Optimization

Seema Hanji (shanji3)

1 Random Search Algorithms

1.1 Overview

An optimization problem is a problem in which "the aim is to find the best state according to an objective function" [Russell and Norvig, 2002]. Randomized optimization algorithms start with a initial state as "best" state, then finding random state as next state , using this state to replace current state if its better than the current state. This process is repeated until no longer able to find better state than the current one.

In this experiment, following optimization problems are utilized to measure performance of various random optimization algorithms. Algorithm and problem implementations from python package *mlrose* are used. For each of the algorithms, various hyper parameters are used to tune the performance of the solutions.

1.1.1 Four Peaks

Evaluates the fitness of an n-dimensional state vector x, given parameter T. Fitness function is defined as to find max between number of 0's in the beginning and number of 1's at the end of a sequence, and return this max if it is greater than given threshold T, else return 0. Fitness functions is defined as

$$f(X,T) = max[tail(0,X), head(1,X)] + R(X,T)$$

$$R(X,T) = \begin{cases} N & if\ tail(0,X) > T\ \ and\ \ head(1,X) > T \\ 0 & otherwise \end{cases}$$

There are two global maxima for this function. They are either when there are T + 1 leading 1's followed by all 0's or when there are T + 1 trailing 0's preceded by all 1's. There are also two suboptimal local maxima that occur with a string of all 1's or all 0's. For this experiment, four peaks is designed with length=*50.*

1.1.2 N-Queens

Evaluates the fitness of an n-dimensional state vector, where each element represents the row position (between 0 and n-1, inclusive) of the column of 'queen', as the number of pairs of attacking queens. N-queens is the problem of placing N chess queens on an N x N chessboard so that no two queens attack each other. For this experiment used N=*12.*

1.1.3 Flip-Flop

Evaluates the fitness of a state vector , as the total number of pairs of consecutive elements of the vector where they are not equal to each other. A maximum fitness bit string would be one that consists entirely of alternating digits. In this experiment, FlipFlop problem was created with length of *60.*

1.2 Algorithms

1.2.1 Random Hill Climbing

This is a standard hill climbing, where next optimum state is found by traversing to neighbour which has better state. In randomized hill climbing, each time a random state is selected as initial state until local maxima is reached.

This algorithm was applied to all three of the problems described above.

Four Peaks problem performance got better as number of max iterations were increased. As shown in plot in **Appendix A.1**, around iterations 50-60, best fit for four peaks reached 50 and kept improving above that through out the test. With a max of ***71***.

N-Queens problem (with n=12) performance using this algorithm is shown in plots in.

Appendix- A.1 . Performance measure of best fitness reached 60 and kept above 60 consistently as number of iterations increased.

FlipFlop performance using RHC doesn't cross 50 and doesn't seem to improve over number of iterations.

With respect to execution times (**A.1-Table 13, 15, 17**), FlipFlop performed better with comparatively shorter execution times.

Hyper parameter tuning :

Problem	Random Restarts	Best Random Restarts
Four Peaks(length=40)	[50, 100]	100
N-Queens(n=12)	[50, 100]	50,100
FlipFlop(length=60)	[50, 100]	50,100

Table 1: RHC - Hyper parameters

Plot in **Figure 1** below compares all 3 of the problems using Random Hill Climbing. Here best fitness is measured against number of max iterations. Though Four Peaks seem to have highest fitness score, this doesn't seem to be consistent. N-Queens seem to improve over iterations and stays consistent.

This indicates "**N-Queens**" is better suited for Random Hill Climbing algorithm.

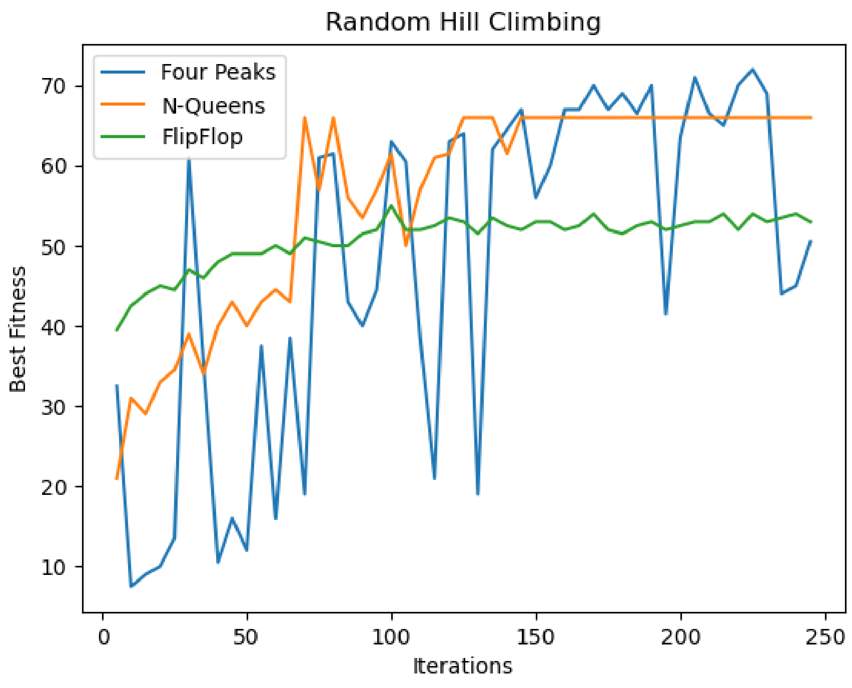


Figure 1: Random Hill Climbing

1.2.2 Simulated Annealing

Simulated Annealing is used to find the best solution for either a global minimum or maximum, without having to check every single possible solution that exists. This algorithm is basically hill climbing, but chooses random move instead of the best move. If the move improves the solution, then it is accepted. If not, then move is accepted with a probability less than 1.

This algorithm was applied to all three of the problems described above.

Four Peaks problem performed poorly until iterations reached 150-160, as shown in plot in **Appendix A.2**, with best fit below 10. And later spiked to 50-60.

N-Queens problem (with n=12) performance using this algorithm is shown in plots in **Appendix- A.2** . Performance measure of best fitness improved as iterations increased and stayed between 50-60 almost close to the test.

FlipFlop performance using Simulated Annealing is shown in plots **Appendix- A.2**. Iterations/fitness shows FlipFlop fitness improved quickly over iterations and stayed around 50 through the test.

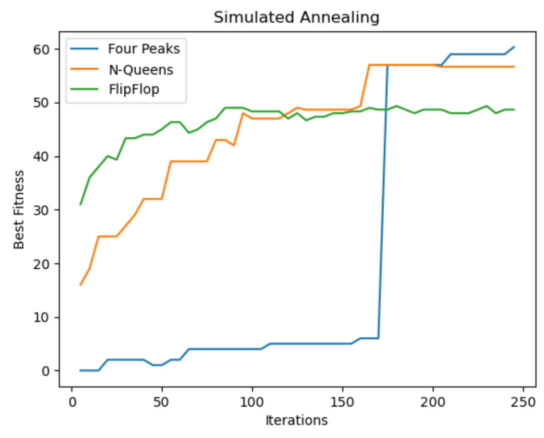
Hyper parameter tuning :

Problem	Decay Schedule	Best Decay Scedule
Four Peaks(length=50)	[GeomDecay,ExpDecay,ArithDecay]	GeomDecay
N-Queens(n=12)	[GeomDecay,ExpDecay,ArithDecay]	GeomDecay,ExpDecay,ArithDecay
FlipFlop(length=60)	[GeomDecay,ExpDecay,ArithDecay]	ExpDecay

Table 2: SA - Hyper parameters

Plot in **Table 3** below compares all 3 of the problems using Simulate Annealing. Here best fitness is measured against number of max iterations. Though Four Peaks and N-queens seem to have highest fitness score, the performance doesn't seem to be consistent. FlipFlop seem to improve early over iterations and stays consistent. Also, the comparison of execution times as shown in **Table 4** indicates FlipFlop execution time is also much better compared to other problems.

This indicates "**FlipFlop**" is better suited for Simulated Annealing algorithm.



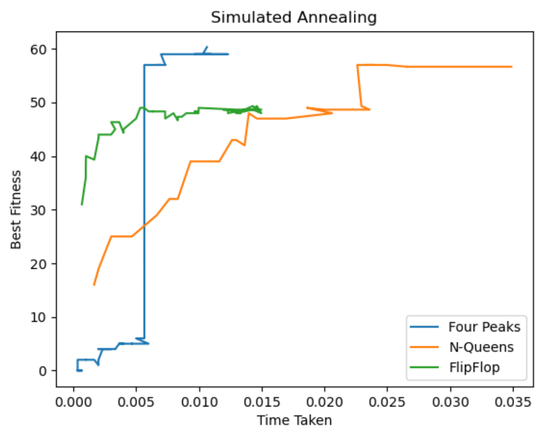


Table 3: Simulated Annealing - Iterations/Bets Fit   Table 4: Simulated Annealing - Exec Time/Best Fit

1.2.3 Genetic Algorithm

Genetic algorithm is used to solve optimization problems based on a natural selection process that mimics biological evolution. The algorithm creates a random initial population. Then creates a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population , by replacing the lower valued population. This seems to be one of the high quality algorithms.

This algorithm was applied to all three of the problems described above.

Four Peaks problem performed very well, as shown in plot in **Appendix A.3**, with best fit close to 70. Best fit scored well even with less number of iterations, reaching quickly to 65 and improving as number of iterations increased.

N-Queens problem (with n=12) performance using this algorithm is shown in plots in **Appendix- A.3** . Performance measure of best fitness stayed low and reached max to 45 as iterations increased.

FlipFlop performance using Genetic algorithm is shown in plots **Appendix- A.3**. Best fit reached max in low iterations, but it stayed at around 45.

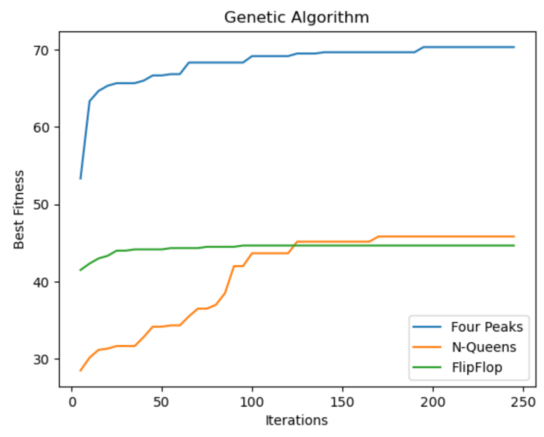
Hyper parameter tuning :

Problem	Pop Size	Mutation Prob.	Best Pop Size / Best Mutation Prob
Four Peaks(length=50)	[100,200]	[0.05,0.1,0.2]	200/0.05
N-Queens(n=12)	[100,200]	[0.05,0.1,0.2]	200/0.05
FlipFlop(length=60)	[100,200]	[0.05,0.1,0.2]	200/0.05

Table 5: GA - Hyper parameters

Plot in **Table 6** below compares all 3 of the problems using Genetic Algorithm. Here best fitness is measured against number of max iterations. Among all 3 problems, Four Peaks seem to perform well , by reaching quickly to higher best fit score(60) and continuing improving through iterations.

This indicates "**Four Peaks**" is better suited for Genetic algorithm.



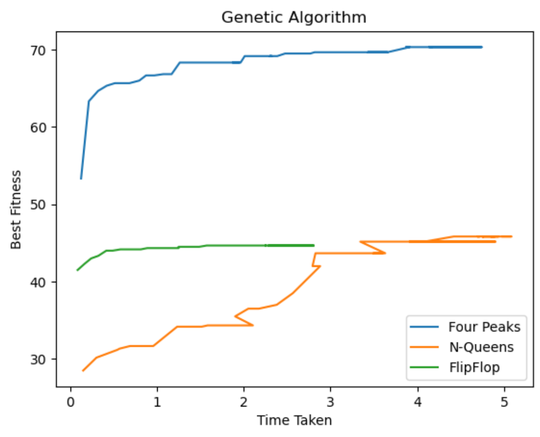


Table 6: Genetic Algorithm - Iterations/Bets Fit   Table 7: Genetic Algorithm - Exec Time/Best Fit

1.2.4 MIMIC Algorithm

MIMIC(Mutual-Information-Maximizing Input Clustering) algorithm uses information of cost function from one iteration to later iterations. MIMIC first randomly samples from those regions of the input space most likely to contain the optima for cost function, and then uses an effective density estimator which can be used to capture input space.

This algorithm was applied to all three of the problems described above.

Four Peaks problem performed well, as shown in plot in **Appendix A.4**, with best fit close to 66. Best fit scored well even with less number of iterations, reaching quickly to 65 and staying at that through the iterations

N-Queens problem (with n=12) performance using this algorithm is shown in plots in **Appendix- A.4** . Performance measure of best fitness stayed at 43 and there was no improvement even with tuning hyper params.

FlipFlop performance using MIMIC algorithm is shown in plots **Appendix- A.4**. Best fit reached max in low iterations, but it stayed at around 50.

Hyper parameter tuning :

Problem	Pop Size	Keep Pct.	Best Pop Size / Best Keep Pct
Four Peaks(length=50)	[100,200]	[0.1,0.15]	200/0.15
N-Queens(n=12)	[100,200]	[0.1,0.15]	[100,200]/[0.1,0.15],
FlipFlop(length=60)	[100,200]	[0.1,0.15]	200/0.15

Table 8: MIMIC - Hyper parameters

Plot in **Table 6** below compares all 3 of the problems using MIMIC Algorithm. Here best fitness is measured against number of max iterations. Among all 3 problems, Four Peaks seem to perform well , by reaching quickly to higher best fit score(65) and staying at that through iterations.

This indicates "**Four Peaks**" is better suited for MIMIC algorithm.

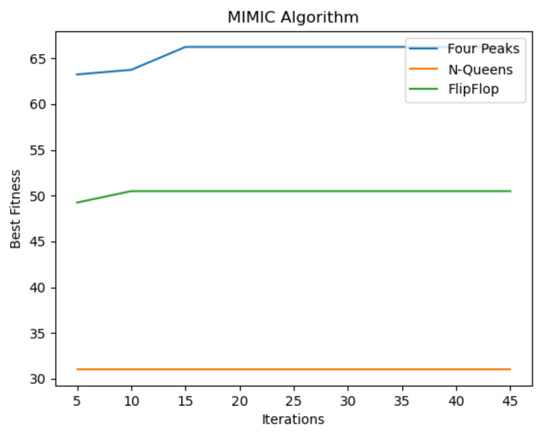


Table 9: MIMIC Algorithm - Iterations/Best Fit

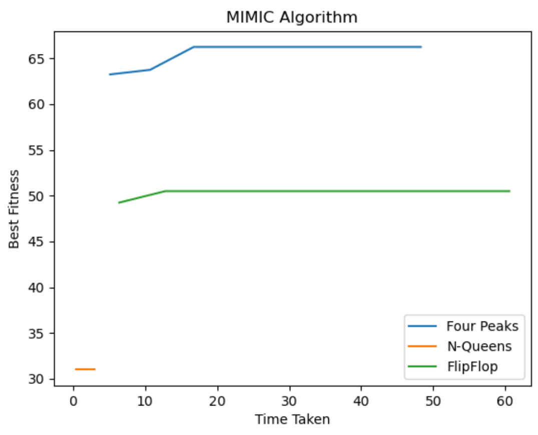


Table 10: MIMIC Algorithm - Exec Time/Best Fit

2 Neural Network

Algorithms *"Random Hill Climbing", "Simulated Annealing" and "Genetic"* are applied to neural network to find optimal weights. From assignment-1 , data set "Credit Card Payment Default"(default of credit card clients.xls) is used to provide classification solution. This dataset is split into train and test, 30% is used as test data.

First hyper parameter for each of the algorithms is fine tuned to get optimal result and then final parameters are applied get final results. The out put of the final results include train and test accuracy, fitness curve which specifies loss with optimal value as 0.

Here are some of the hyper parameters used for fine tuning.

Hyper parameter tuning :

Algorithm	Params	Best Params
Random Hill Climbing	restarts=[0,50,100]	restarts=0
Simulated Annealing	schedule=[GeomDecay,ExpDecay,ArithDecay]	schedule=GeomDecay
Genetic	mutation$_prob = [0.01, 0.1]$	mutation$_prob = 0.1$

Table 11: Neural Network - Algorithms Hyper parameters

Hyper params for Neural network :
activation='relu'
layers=[15]

Following plot displays performance of various algorithms , in comparison to Gradient Descent. And it indicates that gradient descent performs better, with improve in loss and reaching optimal value.

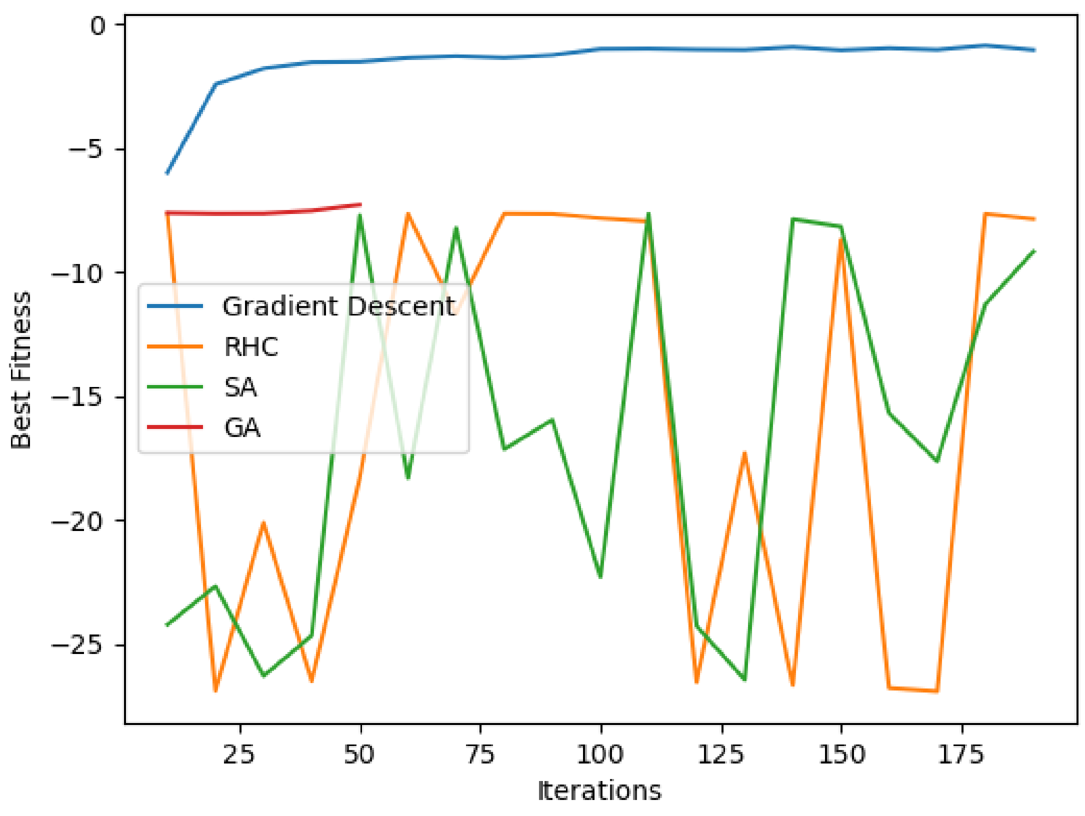


Figure 2: Optimization Algorithms using Neural Network

RHC seemed to start off with minimum error and as max iterations increase, seemed to move further away from the optimal point. As iterations increase, it seems to improve but never recovers from its minimum error point. Simulated Annealing started with its worse point and tries to recover and improve, but can never come closer to optimal. Genetic algorithm shows some consistency between iterations 75-100 and holds up at it's minimum loss position.

Following plot compares test accuracy between these algorithms , used in Neural Network. In comparison, Genetic Algorithm performs better, as it stays at constant accurancy through the experiment.

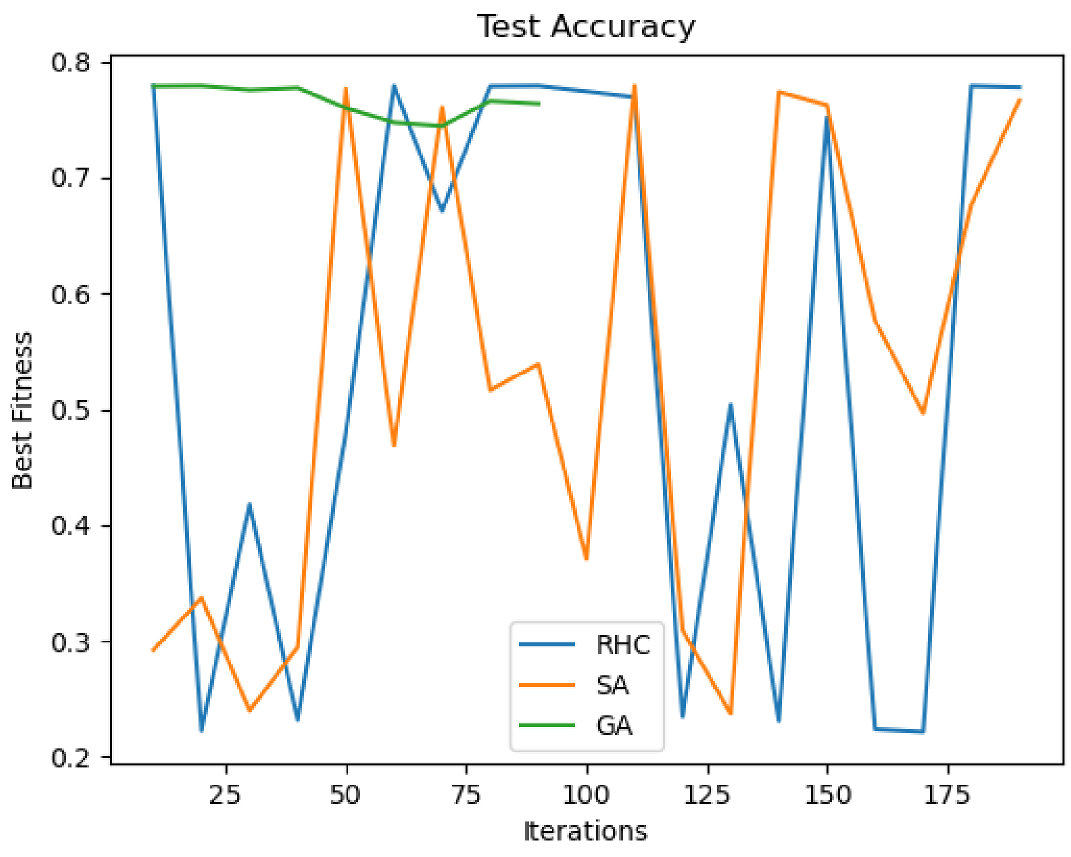


Figure 3: Optimization Algorithms using Neural Network

Other than the default Gradient Descent, Genetic algorithm performs well when used in Neural network.

Due to the nature of this algorithm, this performs well in complex problems.

3 Conclusion

With all the analysis of various optimization algorithms, we can observe that Random Hill climbing and Simulated Annealing behave similarly on set of problems. Due to the basic nature of hill climbing, both perform well in simple problems, but do not perform well in complex scenarios. They have a tendency to be stuck at local maxima if the problem is complex.

On the other hand, Genetic algorithm performs better in complex scenarios as well. Due to the nature of evolution and choosing random states as initial state, this algorithm can cover wide input space.

A Plots

A.1 RHC

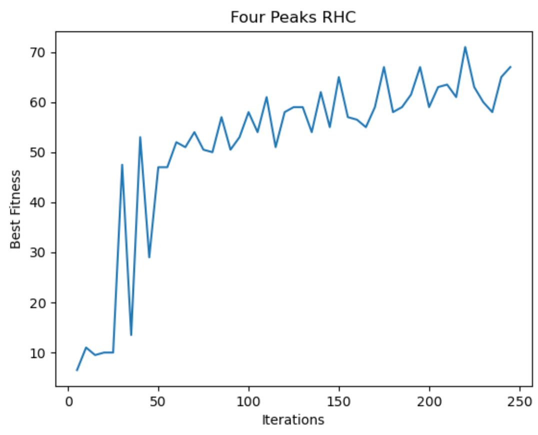


Table 12: Four Peaks - Iterations/Best Fit

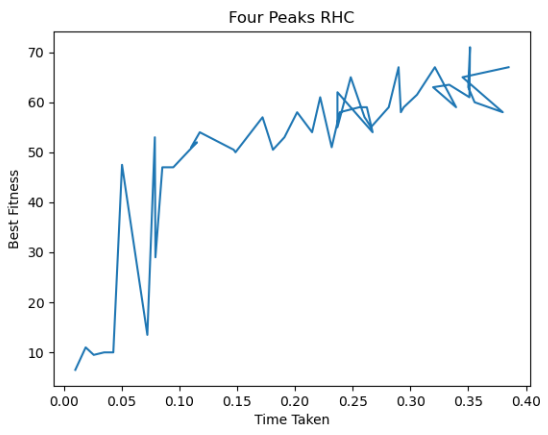


Table 13: Four Peaks - Exec Time/Best Fit

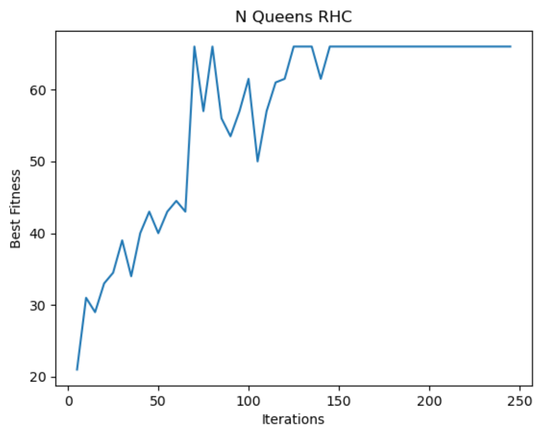


Table 14: N(12) Queens - Iterations/Best Fit

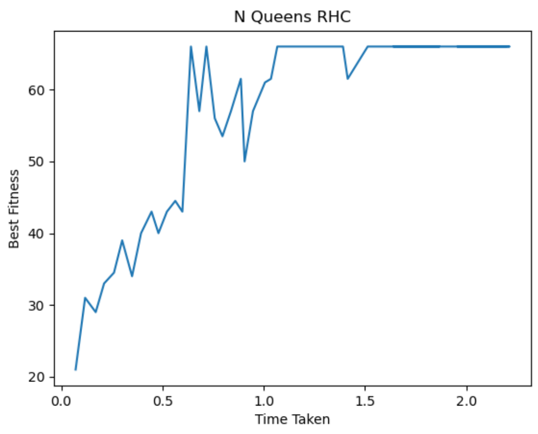


Table 15: N(12) Queens - Exec Time/Best Fit

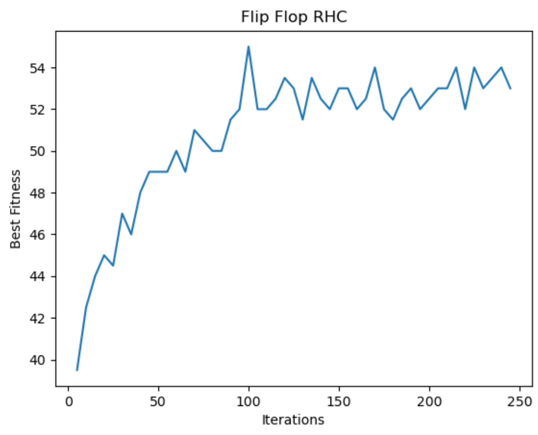


Table 16: FlipFlop - Iterations/Best Fit

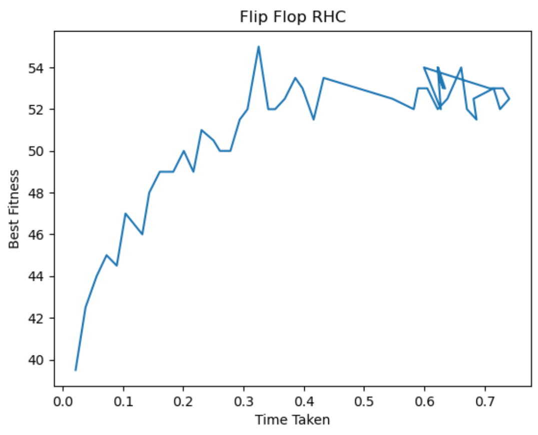


Table 17: FlipFlop - Exec Time/Best Fit

A.2 Simulated Annealing

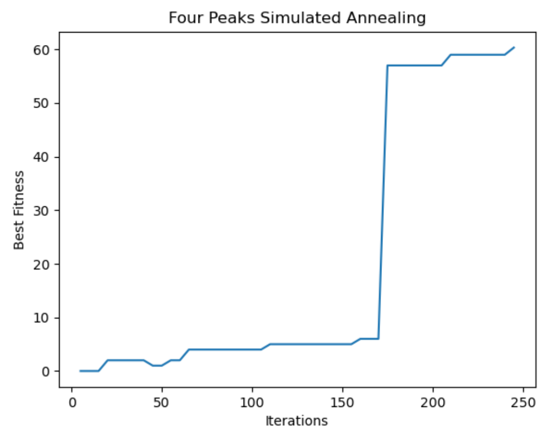


Table 18: Four Peaks - Iterations/Best Fit

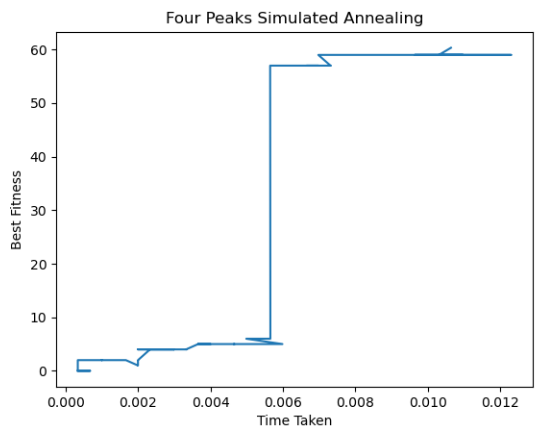


Table 19: Four Peaks - Exec Time/Best Fit

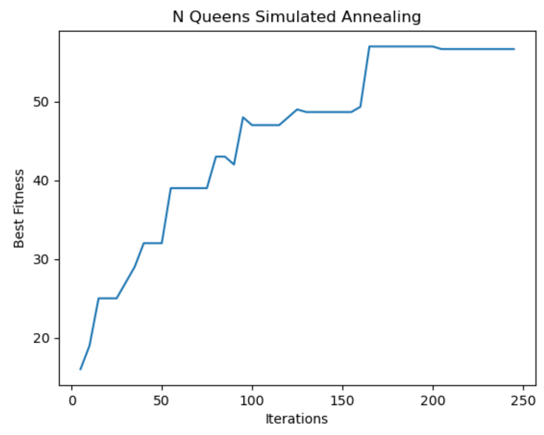


Table 20: N(12) Queens - Iterations/Best Fit

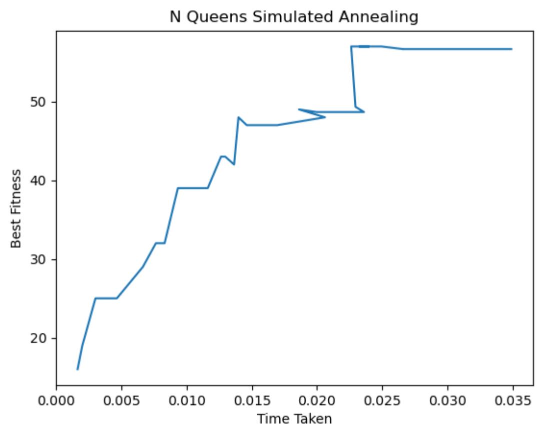


Table 21: N(12) Queens - Exec Time/Best Fit

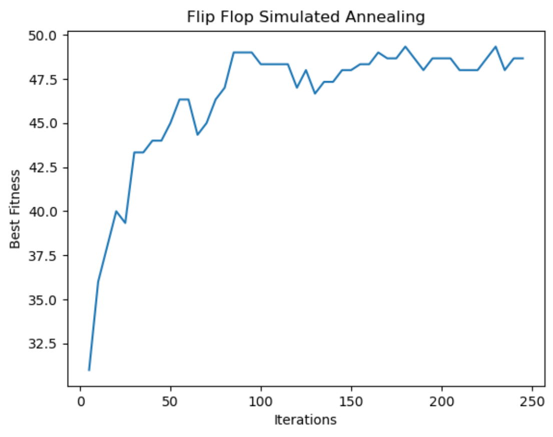


Table 22: FlipFlop - Iterations/Best Fit

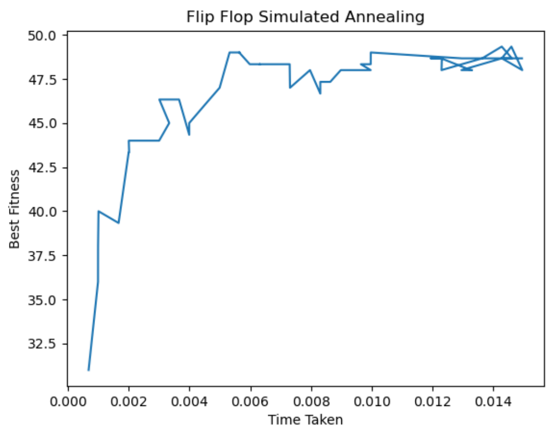


Table 23: FlipFlop - Exec Time/Best Fit

A.3 Genetic Algorithm

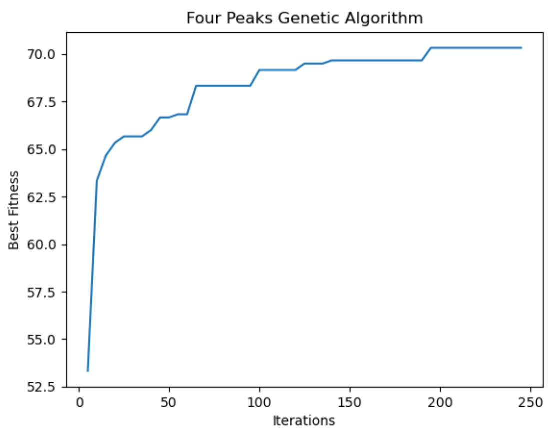


Table 24: Four Peaks - Iterations/Best Fit

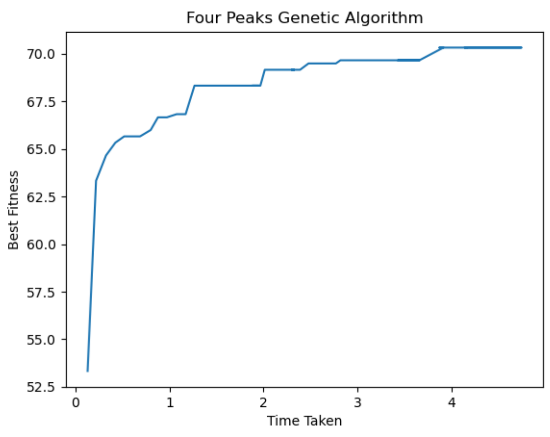


Table 25: Four Peaks - Exec Time/Best Fit

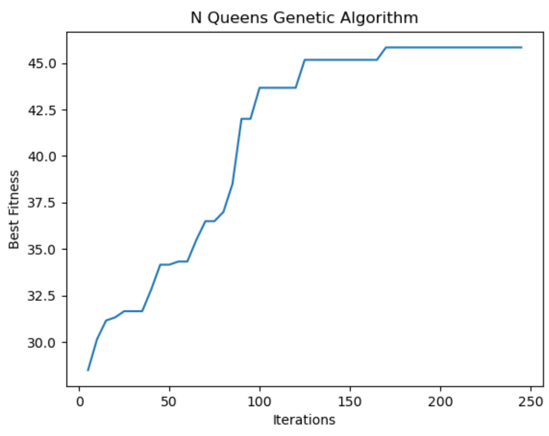


Table 26: N(12) Queens - Iterations/Best Fit

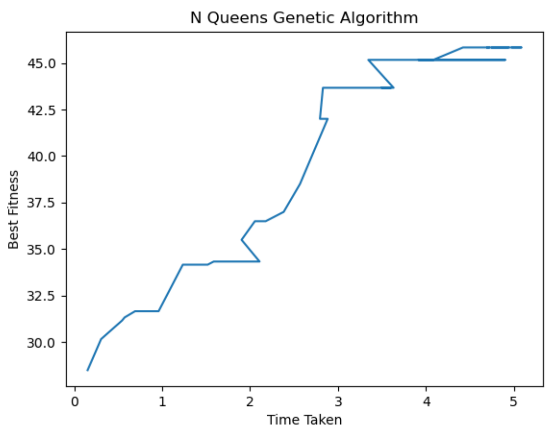


Table 27: N(12) Queens - Exec Time/Best Fit

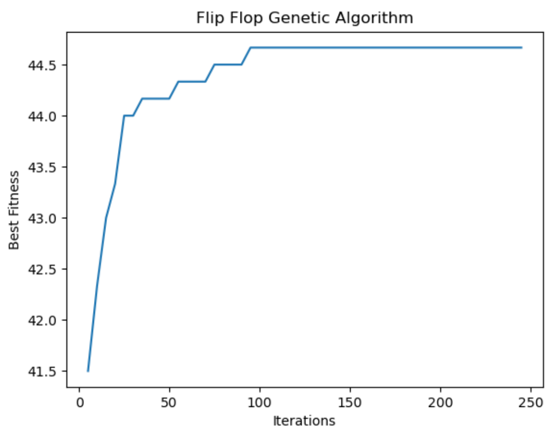


Table 28: FlipFlop - Iterations/Best Fit

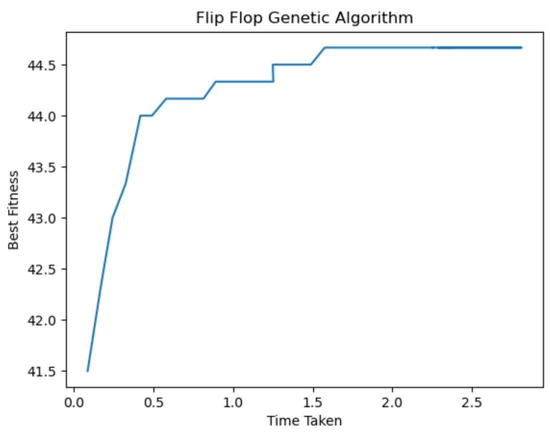


Table 29: FlipFlop - Exec Time/Best Fit

A.4 MIMIC Algorithm

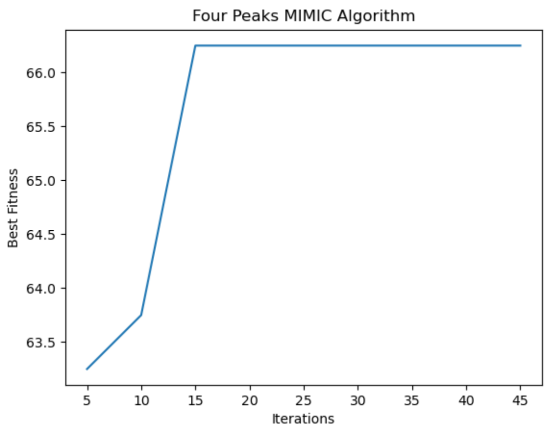


Table 30: Four Peaks - Iterations/Best Fit

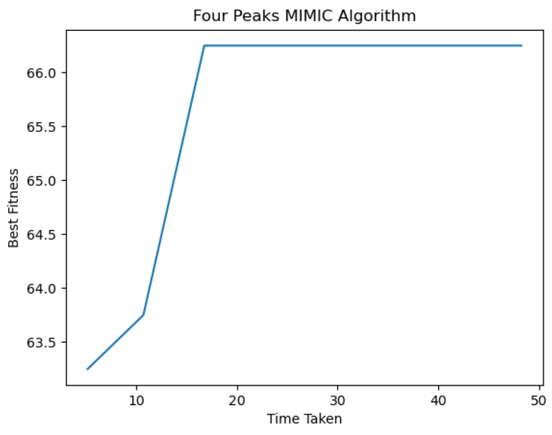


Table 31: Four Peaks - Exec Time/Best Fit

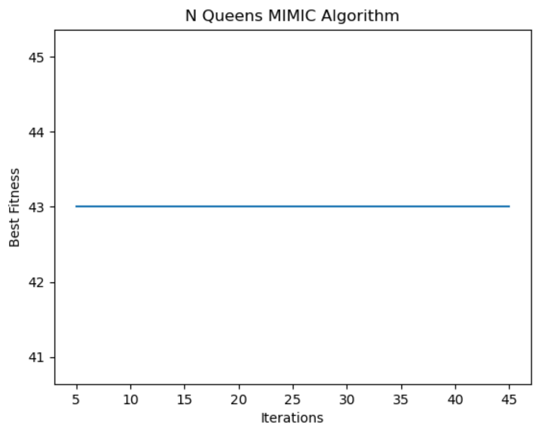


Table 32: N(12) Queens - Iterations/Best Fit

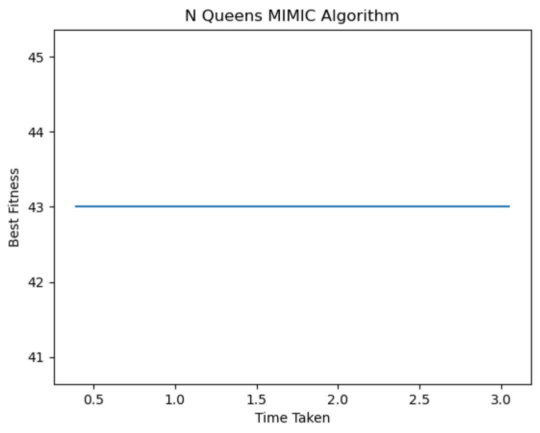


Table 33: N(12) Queens - Exec Time/Best Fit

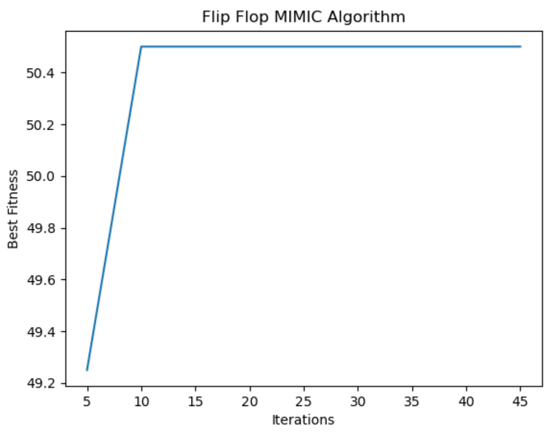


Table 34: FlipFlop - Iterations/Best Fit

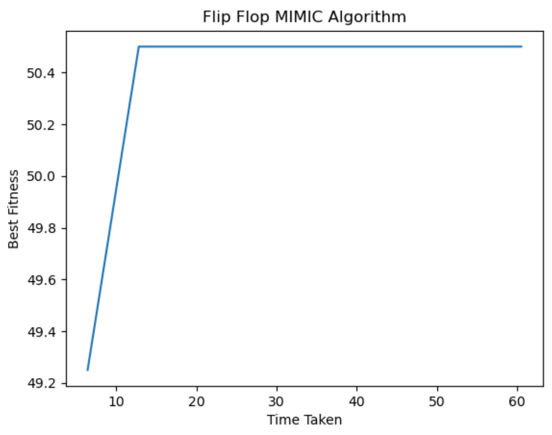


Table 35: FlipFlop - Exec Time/Best Fit

References

[Russell and Norvig, 2002] Russell, S. and Norvig, P. (2002). Artificial intelligence: a modern approach.