Randomized Optimization Assignment

Vivek Nagarajan
CS 7641- Assignment 2
Vnagarajan34@gatech.edu

Abstract

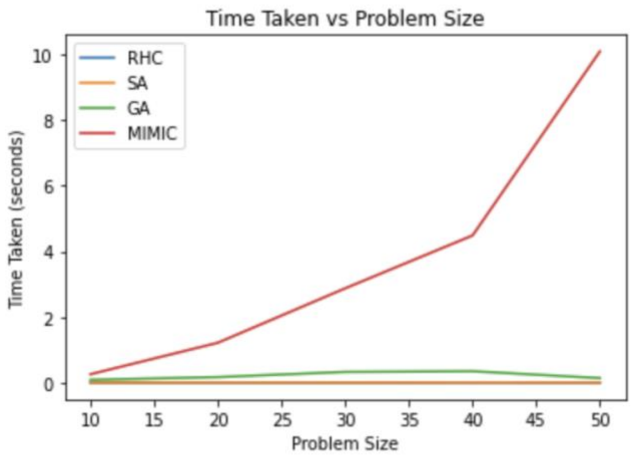
The purpose of this assignment is to compare the performances of four different randomized optimization algorithms on 3 separate problems to judge their advantages and disadvantages by observing their behavior on the various problem cases. The 4 algorithms are Random Hill Climb (RHC), Simulated Annealing (SA), Genetic Algorithm (GA) and Mutual Information Maximizing Input Clustering (MIMIC). The 3 problems chosen are Flip-Flop, Travelling Salesperson and FourPeaks.

Problem 1- Flip-Flop:

Introduction to the Flip-flop Problem

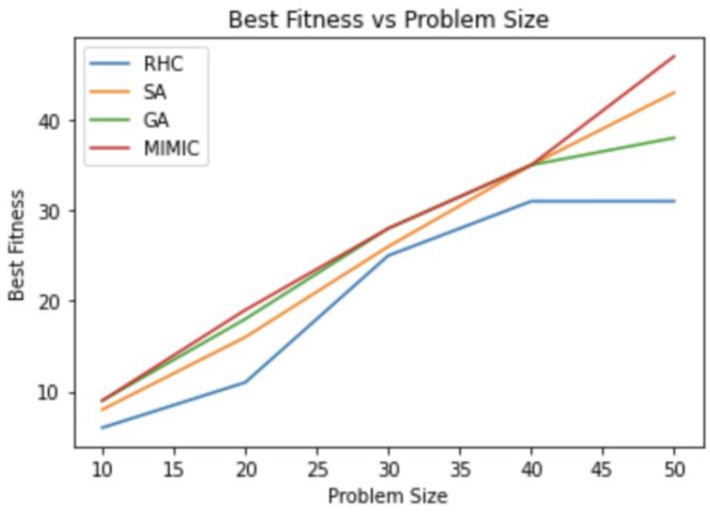
The Flipflop problem is a classic example of a combinatorial optimization problem seen in computer science. The primary objective is to maximize the number of adjacent pairs of bits within a binary string that are different. That is, given a binary string of length *N*, the goal is to maximize the number of 'flips' or transitions from 0 to 1 or from 1 to 0. This problem is faced in various real-world scenarios such as signal processing and error correction.

Time Taken vs Problem Size



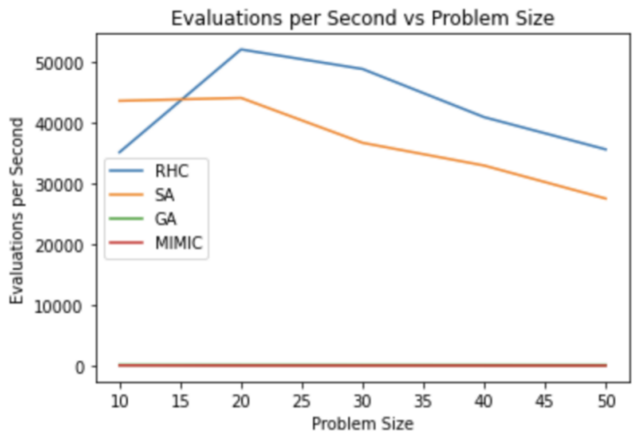
RHC and SA are quicker perhaps due to their iterative nature and fewer parameters to adjust. GA and MIMIC, on the other hand, with their population-based search and complex probabilistic modeling, take more time, especially as the problem size increases. MIMIC, specifically grows exponentially as the Problem size increases. This could be due to the time taken to find out the Distribution that tries to accurately describe the point-space that exceeds the given threshold. This space could become very complex and have lot of gaps as the polynomial grows as higher polynomials have multiple crests and troughs making the function more complex and difficult to calculate.

Best Fitness vs Problem Size



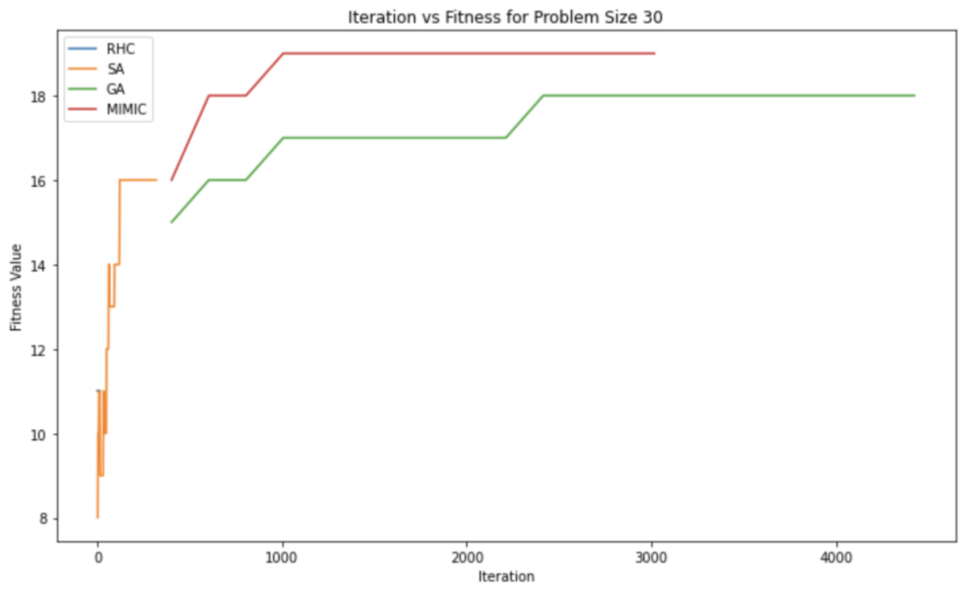
As the problem size grows, the ability to find the optimal or near-optimal solution becomes more challenging. GA and MIMIC achieved higher fitness values due to their population-based approach, which allows exploration of multiple solutions simultaneously. RHC, being a local search algorithm, and SA, which accepts worse solutions under certain conditions, did not fair that good but certainly close given the complexity of the problem was not that high. RHC follows a greedy approach and can be too focused on a certain point tending to "Overfit". SA on the other hand, surpasses GA when it comes to higher problem size as it allows for cooling (rearrangement) of values when an optimal temperature is selected.

Evaluations per Second vs Problem Size



Evaluations per second indicate the efficiency of the algorithm concerning time and computational effort. RHC and SA tend to have higher evaluations per second as they are less computationally intensive on a per-evaluation basis. The efficiency of GA and MIMIC is generally lower due to the more complex operations involved in each evaluation.

Fitness Value vs Iterations



Since RHC is prone to getting stuck at local optima, it terminated quickly. SA on the other hand was able to explore in addition to exploiting and showed gradual increase before eventually terminating around value 16. GA and MIMIC are more capable to search the sample space more thoroughly and simultaneously. This helped them reach much higher Fitness values over multiple iterations without getting stuck at local optima.

Observations

The relative performance of these algorithms can be attributed to their heuristic mechanisms. RHC is simple and fast but lacks robustness. SA introduces randomness to get over local optima but can be slow to converge. GA's crossover and mutation provide a balance between exploration and exploitation, making it effective for a range of problem sizes. MIMIC's complexity allows for a comprehensive search of the solution space at the cost of time and computational resources. Therefore, computational resources not being a constraint, it is better to opt for population-based optimization.

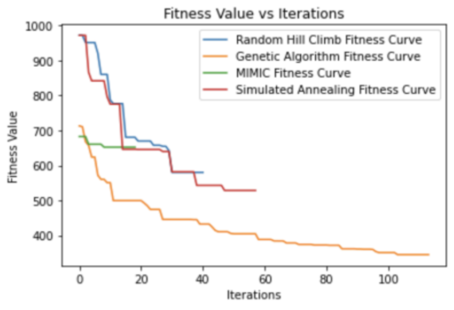
Other problems to potentially reinforce these findings:

Problem 2: Travelling Salesperson

The travelling salesperson is a famous Operations Research problem with its usage spilling into Computer science and networking in addition logistics. The problem is to help a salesperson visit N number of cities and return in the shortest path possible. The input for the cities could either be co-ordinates or distances between consecutive cities. The salesman can visit each city only once, and then, after visiting all the cities on the list, he must come back to the city he started from.

Fitness Value vs Iterations

```
Random Hill Climb: Best Fitness: 580.2928388642224, Time Taken: 0.010578870773315435, Function Evaluations: 41, Eva
luations/Second: 3875.649951544928
Genetic Algorithm: Best Fitness: 345.2808560945127, Time Taken: 2.2901082038879395s, Function Evaluations: 114, Eva
luations/Second: 49.77930728620642
MIMIC: Best Fitness: 652.3743113569111, Time Taken: 3.645542860031128s, Function Evaluations: 19, Evaluations/Secon
d: 5.211843812978176
Simulated Annealing: Best Fitness: 528.641155641346, Time Taken: 0.002698898315429687 5s, Function Evaluations: 58,
Evaluations/Second: 21490.250176678444
```



Here, the objective function is to minimize distance, so the function value decreases with iterations
Random Hill Climb had the worst performance in terms of best fitness, indicating that it did not find as good a route as the other algorithms. However, it was very fast, with the highest number of evaluations per second. Random Hill Climb is a local search algorithm and greedy by nature. Its high number of evaluations per second suggests it's quickly searching through solutions, but its tendency to get stuck in local optima likely explains the lower best fitness value.
The **Genetic Algorithm,** produced a better fitness value than Random Hill Climb but took more time. This algorithm employs a broader search strategy, exploring a larger solution space by combining different solutions ('parents') to create new ones ('offspring'). This method allows for more exploration, which can lead to better solutions but at the cost of additional computational time. (lower evaluations per second)
MIMIC, had the second-best fitness value, showing that it can effectively explore the search space and avoid local optima. However, it had the fewest function evaluations and the lowest evaluations per second, which suggests it's a more computationally intensive method. It spends more time building and sampling from its probability model, which slows down the number of evaluations it can perform. However, the accuracy per evaluation is higher than the other algorithms.
Simulated Annealing performed best in terms of finding the route with the lowest fitness value (indicating the shortest path). It uses a temperature parameter to balance exploration and exploitation: high temperatures allow the algorithm to explore widely and escape local optima, while lower temperatures help it fine-tune and exploit the best solutions found. This method has led to the best overall solution, finding a balance between searching new areas and refining good solutions. The initial value was also quite low for SA. Could this perhaps indicate a specific scenario where the given data seemed favorable to SA? Further experimentation may be needed.
Broadly, the analysis of TSP served to reiterate the results of FlipFlop.

Problem 3: FourPeaks

The Four Peaks Problem a very simple problem and is good for testing algorithms. The fitness consists of counting the number of 0s at the start, and the number of 1s at the end and returning the maximum, if both the number of 0s and the number of 1s are above some threshold value T then the fitness function gets a bonus of 100 added to it. This is where the name ” four peaks“ comes from: there are two small peaks where there are lots of 0s, or lots of 1s, and then there are two larger peaks, where the bonus is included. It is designed to be challenging for optimization techniques due to its deceptive fitness landscape, which can easily trap algorithms in local optima.

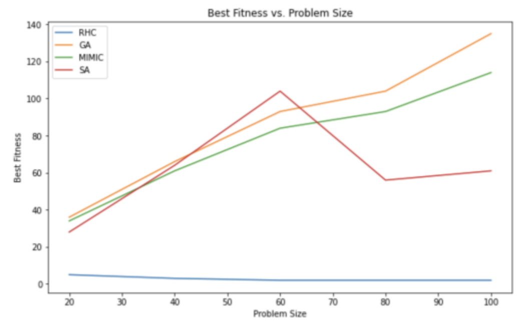
The problem is defined over binary strings of length �*n*. The objective is to maximize a fitness function that is designed to have two competing peaks. The fitness function is usually defined in terms of two components: Tail and Head.

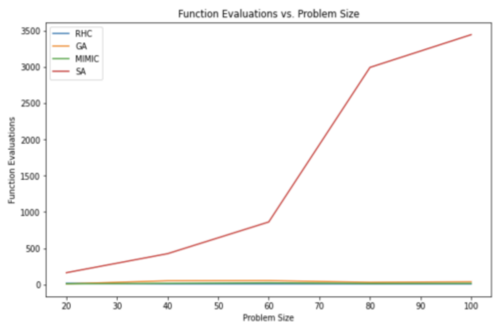
Tail: This is the count of trailing zeros in the string. For example, in the string 0001110,=1*tail*=1 because there's only one zero following all the ones at the end of the string.

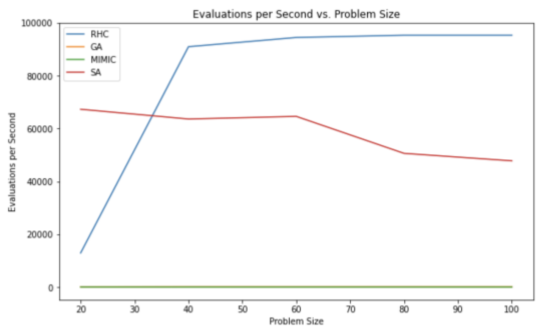
Head: This is the count of leading ones in the string. For example, in the string 1110011, *h*=3*head*=3 because there are three ones at the beginning of the string.

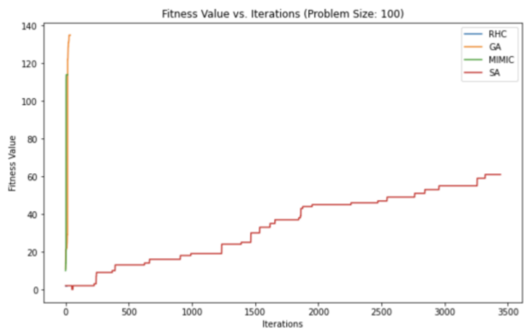
The fitness function is designed to reward strings that have a lot of leading ones or a lot of trailing zeros, but not both.

Performance:







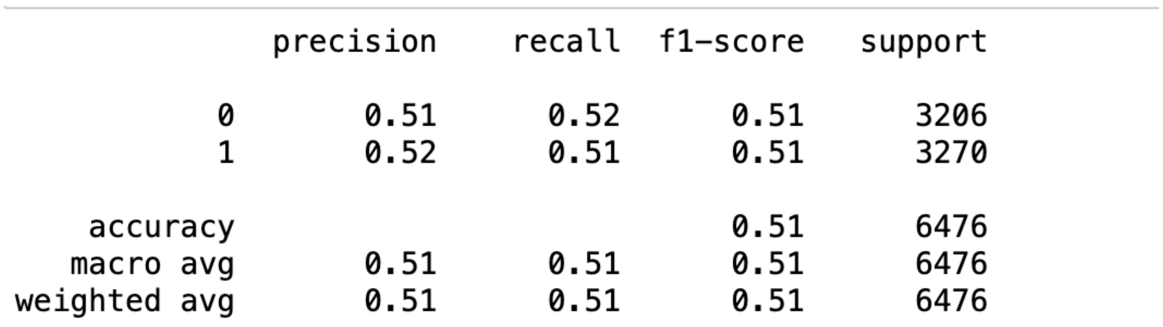


RHC has the least Best Fitness across all problem sizes consistent with other problems. It is interesting to note GA and MIMIC increase in best fitness values as problem size increases. This reinforces the fact that they perform better for more complex problems. **Also, the scoring criteria assigns points for more values so naturally larger string lengths will have more opportunity for earning points provided the algorithm can keep up**.

SA behaves peculiarly as the best fitness drops suddenly beyond a certain problem size. It can also be seen that SA increases in Function evaluations beyond a certain size as well as decreases in Evaluations per second suggesting poorer performance when higher problem sizes are reached. However, the last plot shows a very high number of iterations before termination of SA. This could mean that although it takes a very high time for computation, it does not reach a local maximum for a while. Then again, the fitness score drops off beyond a certain size possibly suggesting overfitting of SA.
The rest of the curves serve to re-enforce the trend analyzed in problem 1.

Neural Network Optimization using RHC, GA and SA:

I have used the same dataset I used in Assignment 1 of credit card transactions used to determine fraudulent transactions. I used the f1 score to calculate the performance metrics.



```
              precision    recall  f1-score   support

           0       0.51      0.52      0.51      3206
           1       0.52      0.51      0.51      3270

    accuracy                           0.51      6476
   macro avg       0.51      0.51      0.51      6476
weighted avg       0.51      0.51      0.51      6476
```

The scores were not high as I did not tune the hyperparameters properly for a lack of time, but this assignment gives me the opportunity to tune my parameters using the algorithms RHC, GA and SA to hopefully optimize the function of my Neural Network.

Random Hill Climb:

```
Accuracy of Neural Network with Random Hill Climbing: 0.51125
                                                                  In [55]:

Precision: 0.52513763059498
Recall: 0.51125
F1 Score: 0.4076309948691158
```

Random Hill climb did not improve the scores by much. Probably because of its propensity to get stuck in a local optimum. The values are like the ones obtained through grid search possibly indicating a lack of enough iterations.

Simulated Annealing:

```
Accuracy of Neural Network with Simulated Annealing: 0.51125
                                                                  In [54]:

Precision: 0.52513763059498
Recall: 0.51125
F1 Score: 0.4076309948691158
```

The SA algorithm could not hurdle over the Local maximum either. The SA has a property that allows itself to come out of the local optimum, but the temperature parameter was not aggressive enough probably to enable the algorithm to find the next peak.

Genetic Algorithm:

```
Accuracy of Neural Network with Genetic Algorithm: 0.8759722222222223
                                                                  In [59]:

Precision: 0.8760628107949149
Recall: 0.8759722222222223
F1 Score: 0.8759735644241076
```

Finally the population based approach which has the property to explore much better and approaches the problem in the least “greedy” approach possible was able to overcome the local maximum and tune the NN to a much higher value bringing the score to 0.875.

Conclusion:

This proves again that the Genetic Algorithm being a population-based algorithm is better suited for complex sample space navigation especially with large problem sizes and complex contours on the sample space.

References:

University of Vienna. Evolutionary Learning Lecture Notes. url: http://vda.univie.ac.at/Teaching/ML/14s/LectureNotes/12_Evolutionary%20Learning.pdf.