Repot optimization

by Advanced Writers

Submission date: 02-Oct-2023 07:47AM (UTC-0600)
Submission ID: 2183281175
File name: Report.docx (1.16M)
Word count: 2212
Character count: 13012

Randomized Optimization

Abstract

In this research study, we investigate the effectiveness of various optimization techniques in training a neural network on the Breast Cancer Wisconsin dataset. Specifically, we compare the performance of Gradient Descent, Genetic Algorithm, Randomized Hill Climbing, and Simulated Annealing as weight update methods. Our analysis centers on assessing testing accuracies and understanding the underlying factors influencing these outcomes.

We compare four optimization algorithms (Random Hill Climb, Simulated Annealing, Genetic Algorithm, and MIMIC) across different problem sizes, considering fitness scores and execution times. In the Continuous Peaks Problem, Simulated Annealing and Genetic Algorithm excel across various sizes, while Random Hill Climb is optimal for smaller problems. MIMIC consistently consumes the most time. For the Flip Flop Problem, Simulated Annealing maintains stability and high scores, outperforming other algorithms as problem sizes increase. However, MIMIC is slower. In the Traveling Salesman Problem, Random Hill Climb maintains stability and high scores, even for larger problems. Simulated Annealing and Genetic Algorithm consistently perform well across sizes, but MIMIC lags due to longer execution times and less efficient solutions.

Part 1: Neural Network Optimization

1. Dataset

a) Description: I select the Breast Cancer Wisconsin (Diagnostic) data from the last homework. This data set is publically available on internet. This dataset contain **30** features of **569** of instances with two classes malignant and benign.

b) **Dataset Overview:**

1. Instances: 569
2. Attributes: 30
3. Class labels: 2 (**M**: Malignant, **B**: Benign)

2. Tools:

(1) For this assignment, I utilized the sys and mlrose Python package to implement randomized optimization algorithms. Additionally, I incorporated the Scikit-learn machine learning library, along with data structures from Pandas and NumPy.

(2) The source code I developed is compatible with Google Colaboratory on Microsoft Windows.

3. Experiment Overview and Methodology

I. The dataset is binary classification problem, I separated it into 80% training set and 20% testing set.

II. The dataset consists of 30 different features, which may include attributes like mean radius, texture, perimeter, area, and various other cell-related measurements.

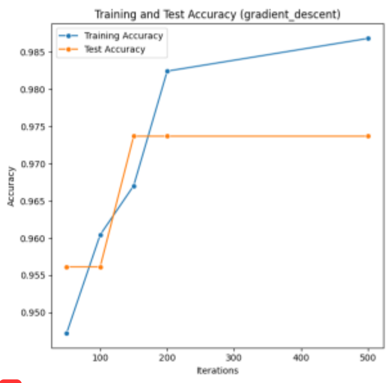
III. Prior to training, I use to preprocess the training data to make prediction more accurate. It because of null values and irrelative feature. It can be done with the help of standard scalar, which involves applying techniques such as Z-score normalization, etc.

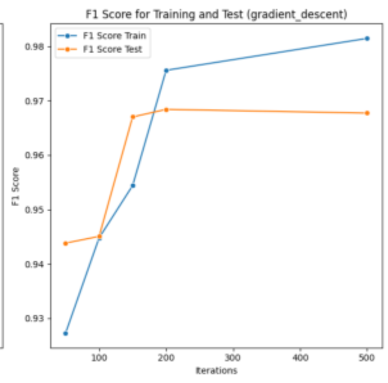
IV. I optimize each algorithm by running them with different numbers of iterations, including 50, 100, 150, 200, and 500, to determine their performance under various conditions. This allows me to assess their efficiency and effectiveness across a range of iteration values.

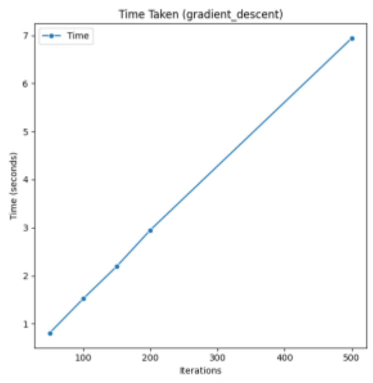
V. In the following graphs, I will demonstrate the accuracy, F1 score and execution time over iterations. The orange line is for testing and the blue line represent training.

4. Gradient Descent (Baseline)

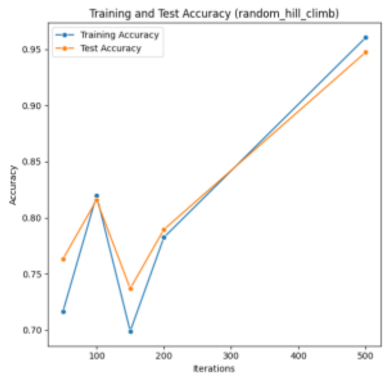
1. To facilitate a more comprehensive comparison, I reintroduced the standard gradient descent neural network as the baseline, employing weight updates through backpropagation, with a fixed learning rate set at 0.0001.
2. We can observe that the accuracy stabilizes after approximately 100 iterations with the testing data, but it exhibits a slower rate of increase beyond 160 iterations. Similarly, the F1 score follows a similar trend as accuracy, plateauing after 200 iterations on the testing data. However, it's worth noting that the computation time also rises proportionally with the number of iterations.

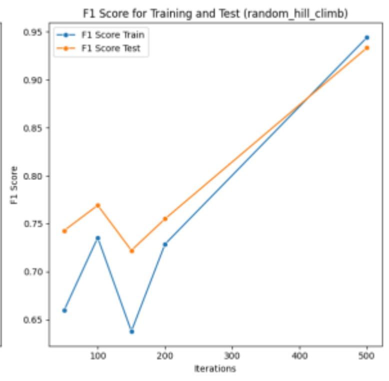


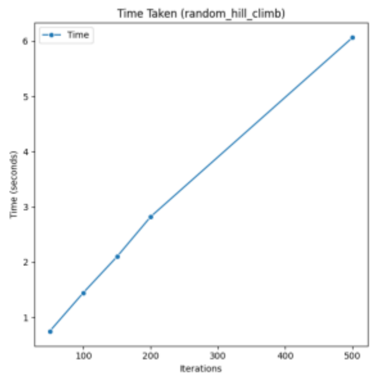




5. Randomized Hill Climbing (RHC)







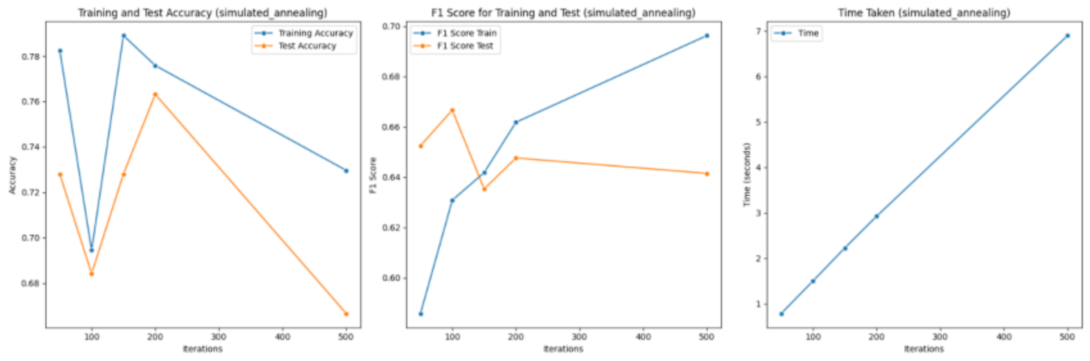
1. In graphs, we can observe that the accuracy converges until 100 iterations, but afterward, it decreases. After 200 iterations, it starts converging again, which is lower than the gradient descent approach. However, in the case of testing, the accuracy starts converging to a higher value than during training. This is because of local optima rather than the global optima.
2. Regarding the time taken, both algorithms almost take the same amount of time, as illustrated in the figures.
3. **Simulated Annealing (SA)**

i. I applied simulated annealing with varying numbers of iterations and observed significant fluctuations in the accuracy of both the training and testing data.

ii. The accuracy exhibits a fluctuating pattern as the number of iterations increases. This behavior could be attributed to the temperature factor introducing randomness, causing the algorithm to explore suboptimal states initially before converging. Consequently, it requires more iterations to reach convergence.

iii. In my problem, the accuracy of simulated annealing (SA) does not improve as the number of iterations increases. Instead, it requires more time for training and a higher number of iterations to achieve convergence.

iv. In terms of time, it remains nearly consistent across all cases, with time increasing as the number of iterations rises.

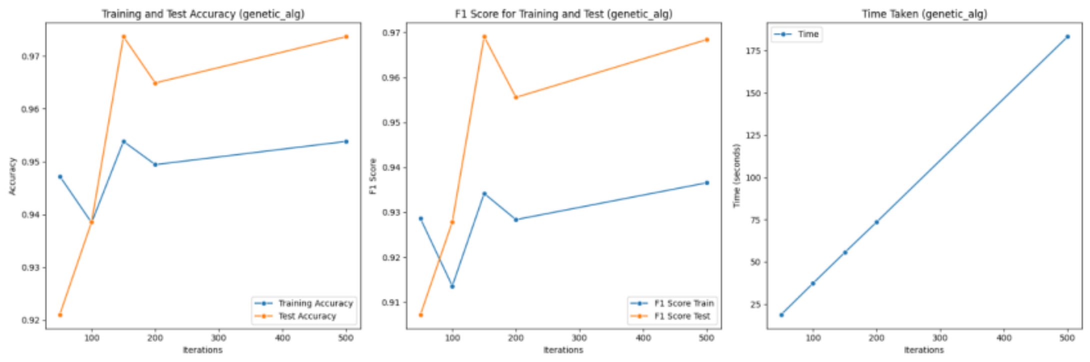


4. Genetic Algorithm

a) The Genetic algorithm achieves consistently high accuracy on both the training and test datasets, with accuracy scores ranging from approximately 0.938 to 0.954 as the number of iterations increases. This indicates the algorithm's effectiveness in learning and generalizing from the data.

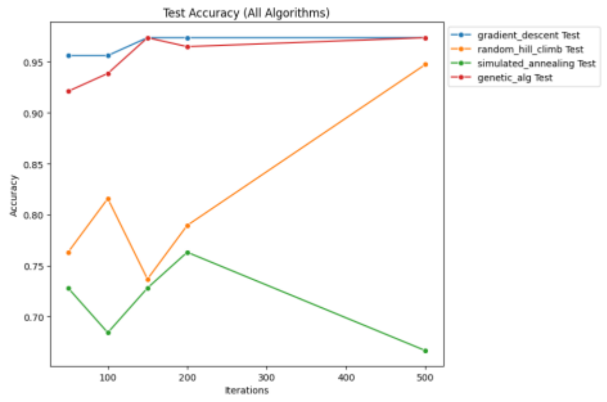
b) As the number of iterations for the Genetic algorithm increases, the computational time also escalates significantly, with execution times ranging from around 18.84 units for 50 iterations to 183.30 units for 500 iterations. This highlights the trade-off between achieving high accuracy and the computational resources required for longer training times.

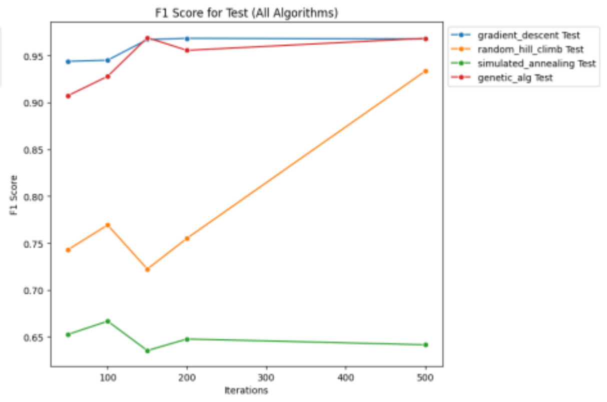
c) In term of time its increases as iterations increase.

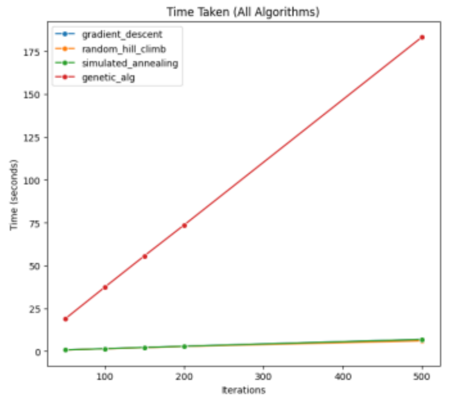


5. Summary

I. The following graph combines the results of the above algorithms:



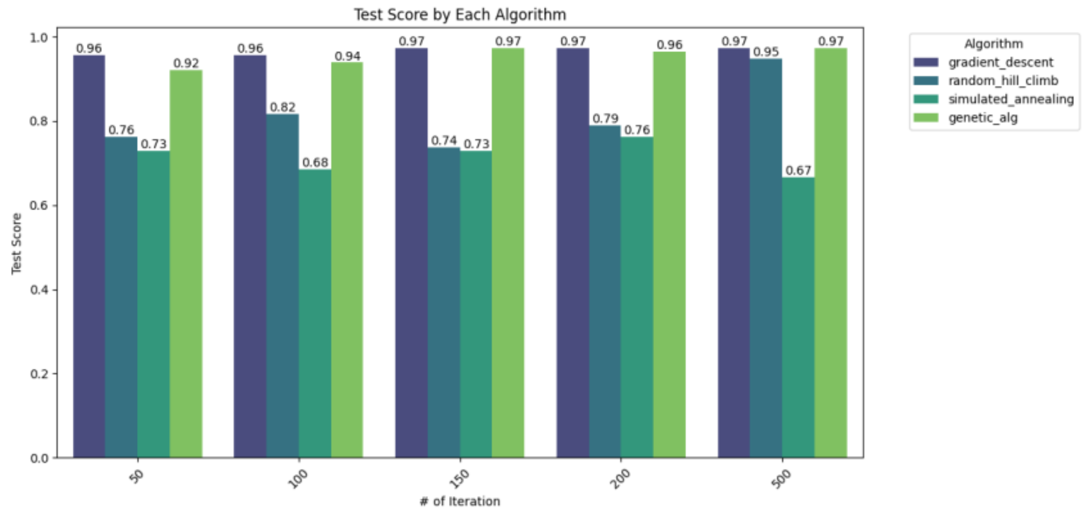


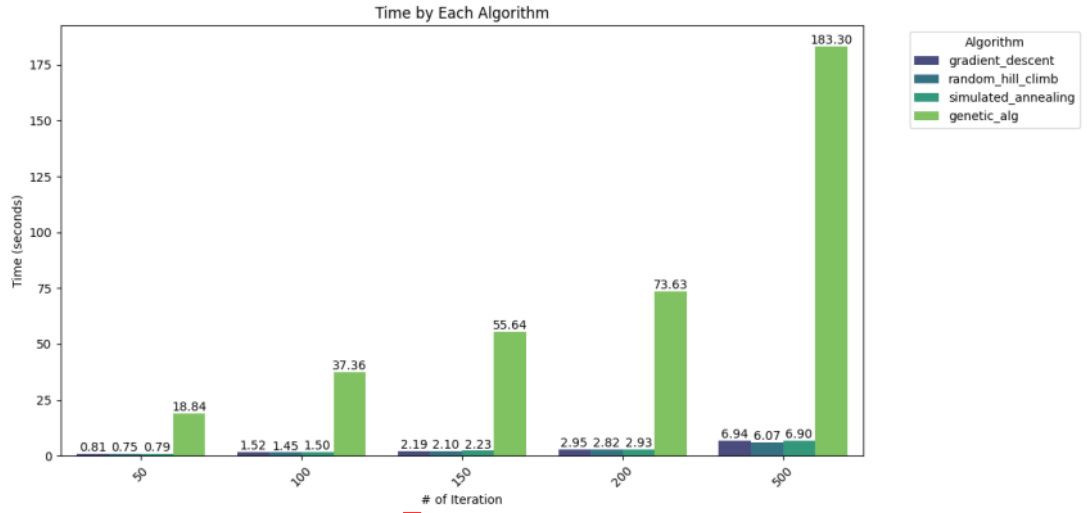


II. Firstly Simulated Annealing achieved good accuracy but late Gradient Descent consistently improves accuracy with more iterations, while Random hill climb exhibit fluctuating performance. Genetic Algorithms maintains strong accuracy, especially with increased iterations.

III. In case of time trade of as iterations increase, so does computation time for all algorithms. Genetic algorithm requires the most time. Gradient Descent is efficient but Genetic Algorithm excels in accuracy at the expense of longer training.

IV. Decision should align with specific needs. For efficiency, Gradient Descent may suffice. For robustness and high accuracy, Genetic Algorithm is strong. Random Hill Climb" and Simulated Annealing need careful tuning for effective convergence, considering the problem's nature and resources available.





V. The table represents a summary of algorithms:

Algorithm	**Best Testing Accuracy**	**Iterations and Time at Best Accuracy**	**Average Iterations per second**	**Overall Ranking**
Gradient Descent	0.97	6.94	0.01388	1 **(Best)**
Genetic Algorithm	0.97	55.64	0.37093	2
Random Hill Climb	0.95	6.07	0.01214	3
Simulated Annealing	0.76	2.93	0.01460	4

Part 2: Optimization Problems

1. Experiment Overview and Methodology

I. I have selected three problems: the Continuous Peaks Problem, the Flip Flop Problem, and the Traveling Salesman Problem. In the upcoming experiments, I will showcase the unique advantages of Random Hill Climb, Mimic, Genetic Algorithm (GA), and Simulated Annealing (SA) individually for each of these problems.

II. I critically design the problem setups for each algorithm, and I find it attractive to apply the knowledge I've gained from these experiments to real-world data.

III. We evaluate them on different problem size to check their processing in term of time.

IV. In the below given pictures, I will illustrate the fitness values and execution time over iterations for each problem

2. Problem 1: Continuous Peaks Problem

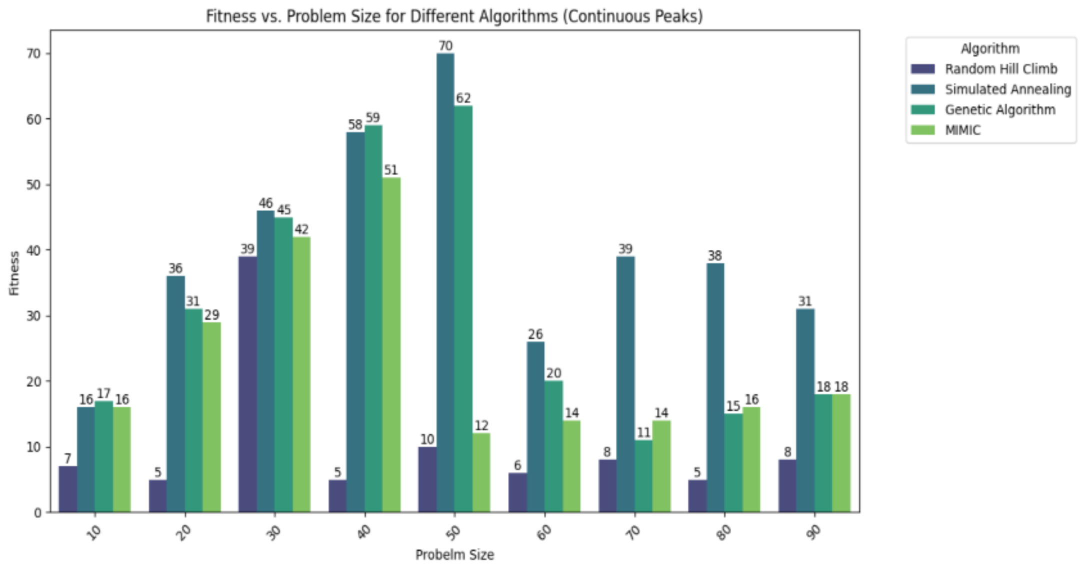
I. Problem Definition: The Continuous Peaks Problem involves finding the maximum number of continuous peaks in a given dataset. A continuous peak is a sequence of consecutive values that are greater than their neighbors. The goal is to identify the longest continuous peak within the dataset.

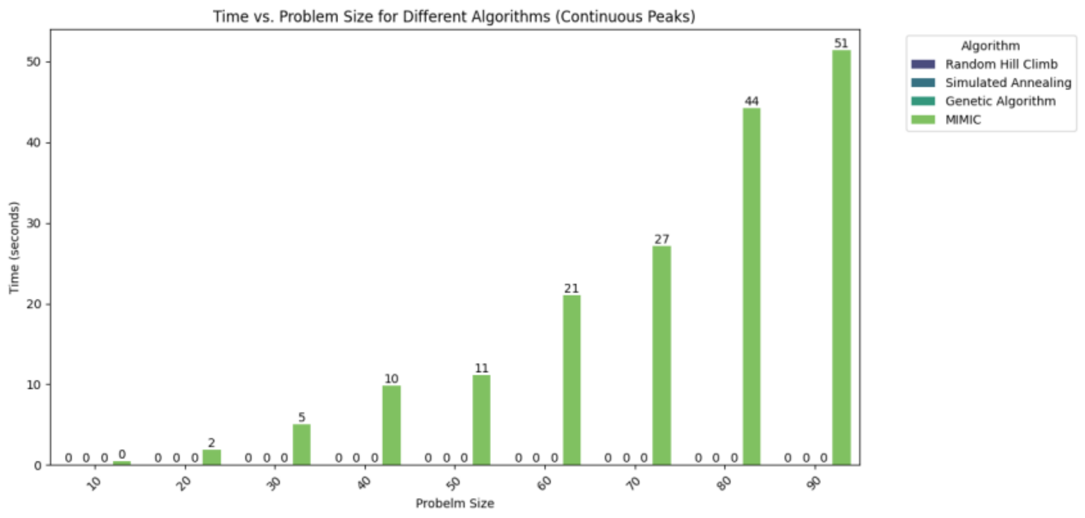
II. Input: The input for the Continuous Peaks Problem consists of a one-dimensional array of data points. Each data point represents a value, and the array represents a dataset.

III. Four different optimization algorithms (Random Hill Climb, Simulated Annealing, Genetic Algorithm, and MIMIC) to solve problems of varying sizes.

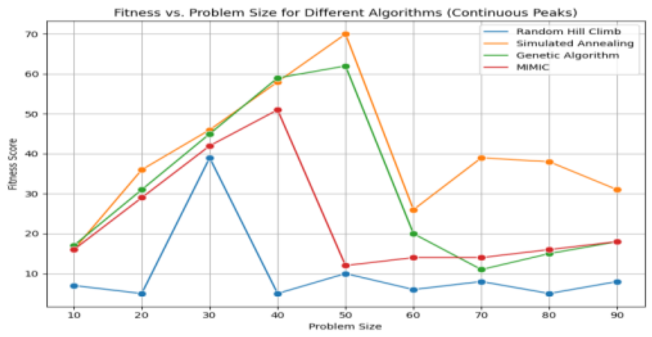
IV. As the problem size increases from 10 to 90, the time required by each algorithm generally increases, demonstrating the computational complexity of these algorithms for larger problem instances. Notably, MIMIC consistently requires the most time, while Random Hill Climb is the fastest for all problem sizes.

V. In term of fitness on different size problem simulated annealing achieve 70 on 50 problem size but it rapidly start decreasing as problem size increase its because of search space complexity and convergence over problem.





VI. We also evaluate four different optimization algorithms for this problem and relate it with fitness score as shown in figure. In this case there are change in fitness score with change in problem size but simulated annealing outperform other.



VII. Simulated Annealing and Genetic Algorithm demonstrate strong performance across various problem sizes when tackling the Continuous Peaks problem. Random Hill Climb excels in finding viable solutions for smaller problem sizes but exhibits diminishing effectiveness as the problem size expands. On the other hand, MIMIC consistently ranks as the slowest and least efficient algorithm among the four in addressing this particular problem.

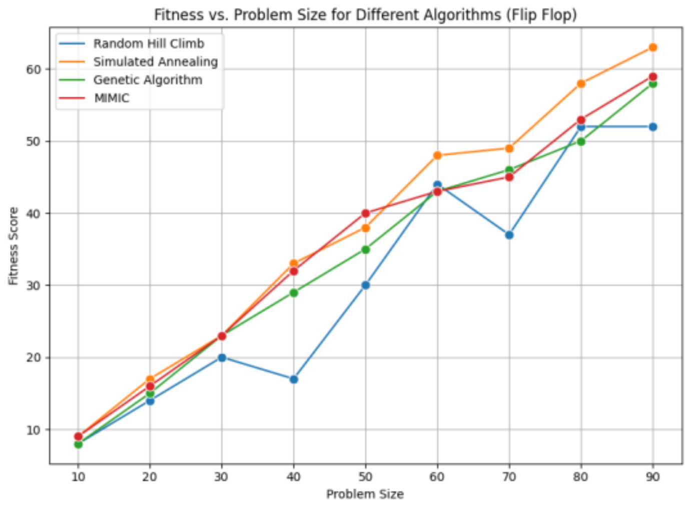
Problem 2: Flip Flop Problem

I. **Flip Flop Problem Definition:** The Flip Flop Problem is characterized by the task of rearranging a binary string (a sequence of 0s and 1s) to maximize the number of alternating bits (changing from 0 to 1 or vice versa). The goal is to find the optimal arrangement that yields the highest count of alternating bits.

II. **Input:** For the Flip Flop Problem, the input consists of a binary string, represented as a one-dimensional array of 0s and 1s. Each element in the array signifies a binary value, and the array as a whole represents the initial state of the problem. The objective is to discover the binary string configuration that results in the maximum count of alternating bits.

III. Four distinct optimization algorithms (Random Hill Climb, Simulated Annealing, Genetic Algorithm, and MIMIC) applied to address problems of diverse scales.

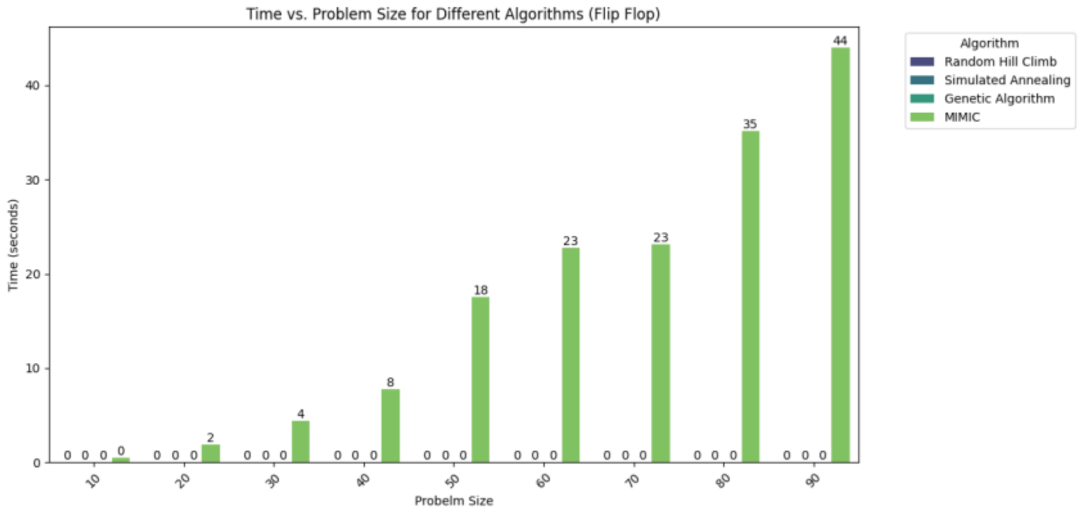
IV. The Simulated Annealing demonstrates remarkable stability, rapid convergence, and strong performance, particularly at the initial stages of problem-solving. Furthermore, it outperforms MIMIC with significantly good fitness score as size of problem increase.

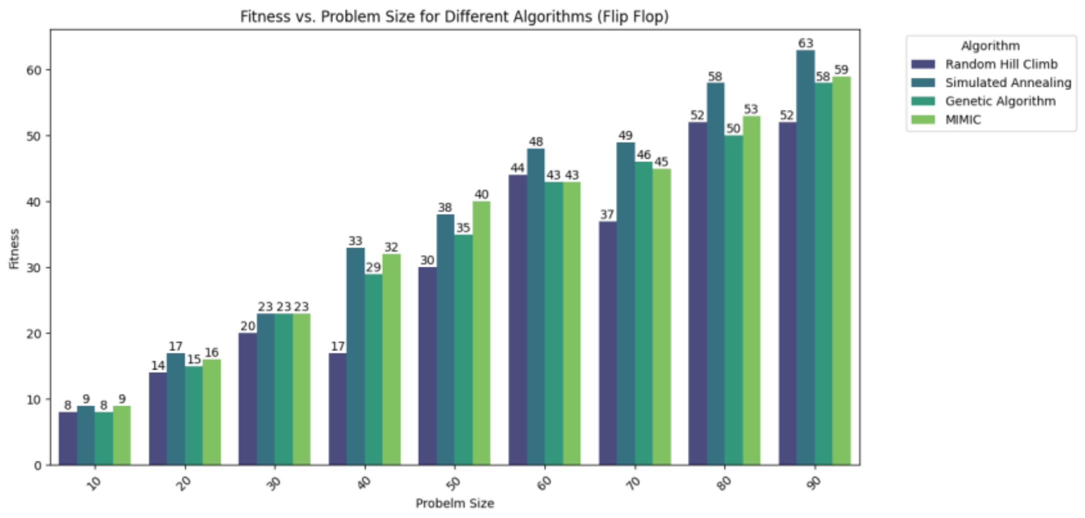


V. In terms of time, as depicted in the figure, the execution time for all algorithms remains consistent at below 1 second as the problem size increases. However, MIMIC stands out with significantly longer execution times. This divergence can be attributed to the fact that MIMIC explores a more extensive solution space and employs additional computational resources, resulting in longer processing times.

VI. We also evaluate four different optimization algorithms for this problem and relate it with fitness score as shown in figure. In this case there significance positive change in fitness score with change in problem size but simulated annealing outperform other.

VII. Simulated Annealing and Genetic Algorithm excel on the Flip Flop problem across various sizes, effectively exploring the search space, with Simulated Annealing benefiting from its probabilistic approach and Genetic Algorithm's efficient use of crossover and mutation operations. Meanwhile, Random Hill Climb performs reasonably well for smaller problems but may struggle with larger ones, while MIMIC generally performs poorly due to its time-intensive nature.





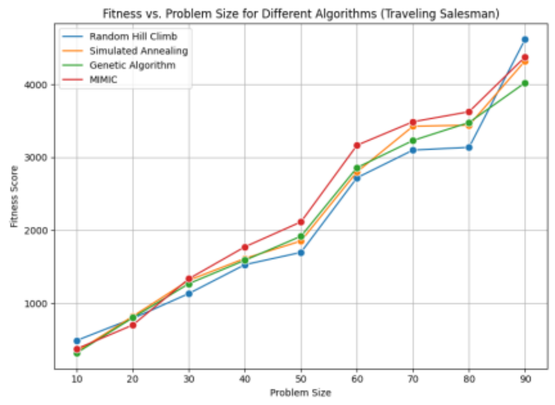
Problem 3: Traveling Salesman Problem (TSP)

I. **Traveling Salesman Problem (TSP) Definition:** The Traveling Salesman Problem involves determining the shortest possible route that visits a set of cities and returns to the starting city. The objective is to find an optimal tour that minimizes the total distance traveled while visiting each city exactly once.

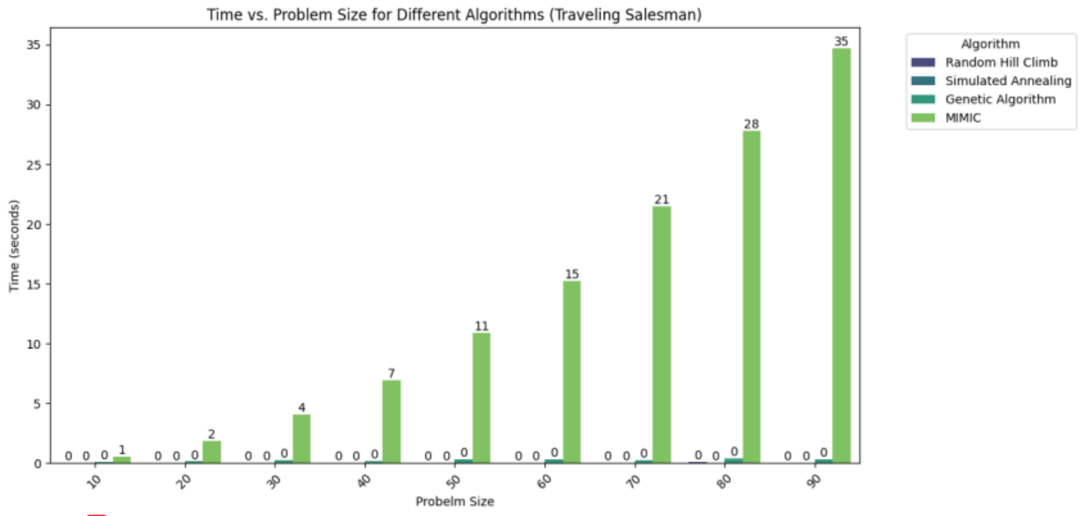
II. **Input:** For the Traveling Salesman Problem, the input comprises a list of cities and their respective coordinates in a two-dimensional space. Each city is represented as a point with x and y coordinates, allowing for the calculation of distances between cities. The goal is to discover the most efficient route that connects all cities and returns to the starting point, thereby minimizing the total distance traveled.

III. Four optimization algorithms (Random Hill Climb, Simulated Annealing, Genetic Algorithm, and MIMIC) employed to solve problems of varying complexity and size.

IV. Random Hill Climb showcases impressive stability, swift convergence, and robust performance, especially during the initial phases of problem-solving. Notably, it outshines the other algorithms, achieving notably high fitness scores as the problem size grows. It reaches a fitness score exceeding 4000 for problem sizes as large as N=90.

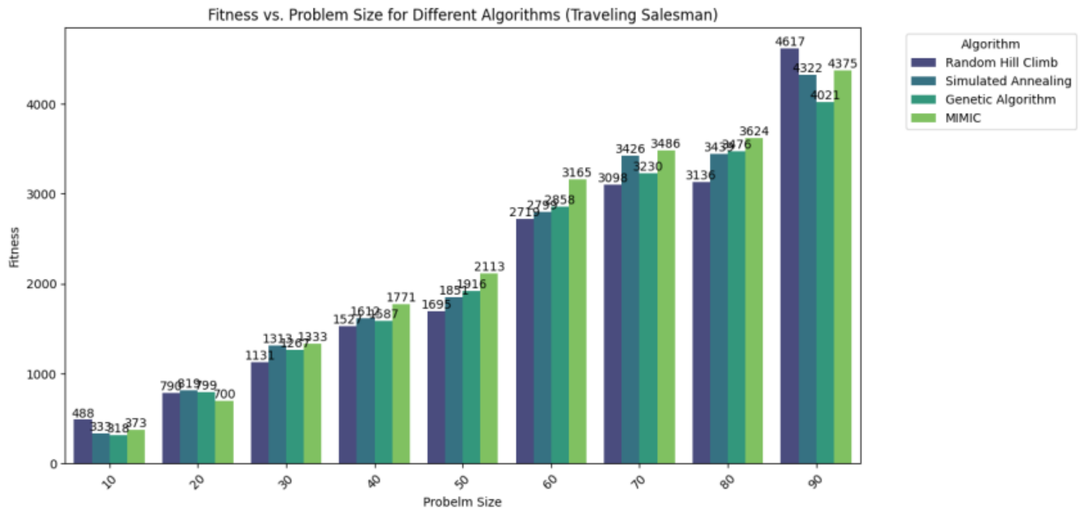


V. Regarding execution time, as illustrated in the figure, all algorithms maintain consistent performance, with execution times remaining below 1 second even as the problem size increases. However, MIMIC notably deviates from this trend, exhibiting significantly longer execution times. This distinction can be attributed to MIMIC's exploration of a more extensive solution space.



VI. All optimization algorithms (Random Hill Climb, Simulated Annealing, Genetic Algorithm, and MIMIC) for problems of varying sizes. As the problem size increases, there is a general trend of fitness scores rising, indicating improved solutions. Notably, MIMIC consistently has the lowest fitness scores, suggesting its relatively weaker performance compared to the other algorithms across different problem sizes.

0



VII. Simulated Annealing and Genetic Algorithm consistently excel in solving the Traveling Salesman Problem across various problem sizes, owing to their effective search space exploration capabilities.

Simulated Annealing benefits from its probabilistic nature, allowing it to escape local optima, while Genetic Algorithm efficiently explores the search space through crossover and mutation operations. In contrast, Random Hill Climb finds reasonable solutions but struggles with global optima, while MIMIC's performance appears less competitive due to extensive execution times and less efficient solution discovery.

References

[1] mlrose: Machine Learning, Randomized Optimization, and Search. Retrieved from mlrose.readthedocs.io (https://mlrose.readthedocs.io/en/stable/)

[2] Traveling Salesman Problem (TSP) – Wikipedia (https://en.wikipedia.org/wiki/Travelling_salesman_problem)

[3] Hill Climb Optimization. Retrieved from Geeksforgeeks (https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/)

[4] Continuous Peaks Problem - Randomized Optimization, Charles Isbell, Michael Littman, Udacity. Retrieved from Udacity (https://auth.udacity.com)

Repot optimization

ORIGINALITY REPORT

13%	1%	1%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Georgia Institute of Technology Main Campus Student Paper	10%
2	Submitted to Colorado Technical University Student Paper	1%
3	Submitted to RMIT University Student Paper	1%
4	K. Rasheed. "Automated synthesis of standard cells using genetic algorithms", Proceedings IEEE Computer Society Annual Symposium on VLSI New Paradigms for VLSI Systems Design ISVLSI 2002 ISVLSI-02, 2002 Publication	1%
5	cse.unl.edu Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		