An Analysis of Optimization Strategies in Machine Learning

Manavi Shukla[1]

[1]*gtid:903952938, mshukla40@gatech.edu*

Abstract

This report compares the performance of various optimization algorithms for neural network training and solving bitstring optimization problems. We first identify the most suitable problems by testing several bitstring challenges, selecting the Flip Flop problem for Simulated Annealing (SA) and the Four Peaks problem for Genetic Algorithm (GA). Each algorithm is tuned across multiple hyperparameters to analyze their impact on performance, considering metrics like fitness progression, convergence rate, and computational efficiency. We also compare these algorithms as alternatives to backpropagation for training neural networks on a previously examined dataset. The report concludes by identifying the most effective algorithm and suggesting potential improvements.

1. Introduction

This report explores the application of various optimization algorithms to neural network training and bitstring optimization problems. Bitstring problems are chosen for their discrete nature and ability to highlight the strengths and weaknesses of different optimization strategies. To ensure meaningful comparisons, we test several bitstring challenges and select the Flip Flop problem for Simulated Annealing (SA) and the Four Peaks problem for Genetic Algorithm (GA) based on their complexity and suitability for analysis.

We use the mlrose library, which provides implementations of various optimization algorithms, to facilitate our experimentation. The selected algorithms are tuned across multiple hyperparameters to examine their impact on performance, considering metrics such as fitness progression, convergence, computational efficiency and scalability.

Additionally, we compare these optimization strategies as alternatives to backpropagation for training neural networks on a previously examined dataset. This involves using the optimization algorithms to update neural network weights, analyzing their effectiveness in this context.

The report concludes by identifying the most effective algorithm for each problem and suggesting potential improvements for future applications. This comprehensive analysis provides insights into the relative strengths of different optimization techniques in both discrete and continuous domains.

2. Optimization Strategies

Optimization algorithms play a crucial role in solving complex problems by finding the best solution from a vast search space. This section explores four popular optimization strategies: Randomized Hill Climbing (RHC), Simulated Annealing (SA) and Genetic Algorithm (GA). We also hypothesize the types of problems where each strategy is expected to excel.

2.1. Randomized Hill Climbing

It is a simple local search algorithm which starts with a random solution and iteratively moves to a neighboring solution with a higher fitness value. If no such neighboring solution exists, the algorithm terminates.[4] RHC is best suited for problems with smooth fitness landscapes and few local optima. It is efficient for problems where good solutions are clustered together, making it easy for the algorithm to find the global optimum.

2.2. Simulated Annealing

Simulated Annealing is an optimization technique inspired by the process of annealing metals. Just like slowly cooling molten metal allows its structure to become more ordered and minimizes its internal energy, Simulated Annealing helps us find the best possible solution (minimum or maximum) for a given problem. [5]

Here's how it works:

- Exploring the Landscape: At each step, the algorithm explores new possibilities within the problem space. The "distance" or randomness of these explorations is controlled by a factor that acts like temperature. Higher temperatures allow for wider exploration, while lower temperatures focus the search on promising areas.
- Escaping Local Peaks: Unlike some methods, Simulated Annealing can sometimes accept solutions that don't immediately improve the situation. This, with a controlled probability, allows the algorithm to escape from getting stuck in suboptimal solutions (local minima) and explore a broader range of possibilities.
- Gradual Refinement: Over time, this "temperature" is gradually lowered, restricting the search and focusing it on the most promising areas. This process helps the algorithm converge on the optimal solution, just like cooling metal solidifies into a more ordered structure.

2.3. Genetic Algorithm

Genetic Algorithms (GAs) are a class of computational search techniques inspired by the principles of natural selection. They are adept at finding approximate solutions to optimization problems by mimicking the process of evolution.

The process iterates through generations, with each generation potentially containing better solutions than the previous. This cycle continues until a termination criterion is met, such as reaching a maximum number of generations or finding a sufficiently good solution.

Genetic Algorithms offer a powerful approach to tackle complex optimization problems, particularly those where traditional methods might struggle. They leverage the power of natural selection to guide the search towards promising solutions. However, due to the inherent randomness involved, the final solution may not be the absolute optimum, but it will likely be very close. [2]

2.4. MIMIC

Mutual-Information-Maximizing-Input-clustering (MIMIC) is an advanced optimization algorithm that builds probabilistic models to guide the search. It samples solutions based on the current model, evaluates their fitness, and updates the model to focus on promising areas of the search space. MIMIC is suited for problems where understanding the structure of the search space can significantly reduce the number of evaluations needed to find the optimum. It performs well on problems with strong dependencies between variables.[1]

3. Fitness Problems

3.1. Four Peaks Problem

This problem aims to maximize the number of consecutive identical bits (either all 0s or all 1s) at the beginning and end of a binary string. It receives its name due to the resulting fitness landscape resembling four peaks: two high peaks on the edges (representing strings with all 0s or all 1s) and two lower peaks in the center (representing strings with a large cluster of 0s followed by a large cluster of 1s, or vice versa) as seen in figure 1.

The problem's requirement to find long sequences of 1s and 0s makes it suitable for demonstrating the strengths of Genetic Algorithms. GA can effectively combine and mutate solutions to explore the search space and find these long sequences. [3]

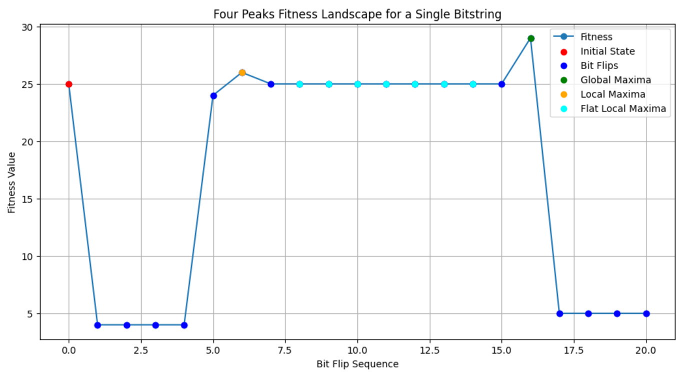

Figure 1. Example Four Peaks Problem

3.2. Flip Flop Problem

This problem seeks to maximize the number of alternations (0101...) within a binary string. The fitness function typically counts the number of these alternations. Due to the nature of this problem, strings with a single misplaced bit can have a significant impact on the fitness score, leading to a rugged and deceptive search space as seen in figure 2. It provides a contrasting scenario to Four Peaks, where the simplicity of the fitness function belies the complexity of achieving high fitness, thus showcasing the different strengths of the algorithms in handling deceptive landscapes.

The numerous local optima in the Flip Flop problem make it well-suited for Simulated Annealing. SA's ability to probabilistically accept worse solutions helps it escape these local optima and find better solution.

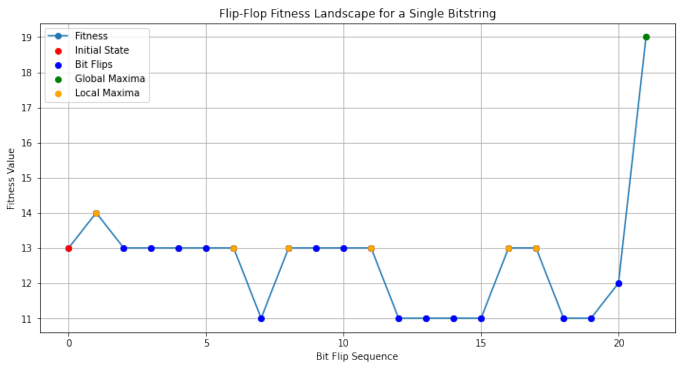

Figure 2. Example Flip Flop Problem

3.3. One Max Problem

The One Max problem is a simpler optimization problem where the goal is to maximize the number of 1s in the bitstring, refer figure 3. This problem can highlight the strengths of GA due to its efficient exploration and exploitation mechanisms. This problem was not expanded on further, as we progress with Four Peaks and Flip Flop as our target optimization problems.

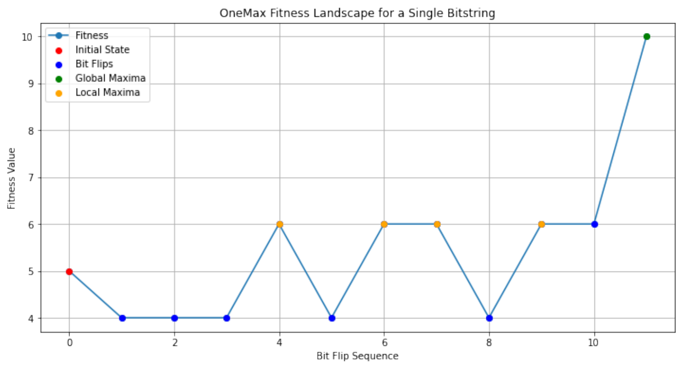

Figure 3. Example One Max Problem

4. Hypothesis

4.1. Fitness Problem

- Hypothesis 1: Simulated Annealing will work better consistently for all problem sizes than genetic algorithm on Flip Flop Problem as it allows for uphill moves (accepting worse solutions) with a certain probability, which helps escape local maxima. This is crucial because the flip flop problem has many "almost good" solutions with just a few non-alternating bits.
- Hypothesis 2: For Four Peaks Problem, A fitness function with two peaks, requiring a balance between exploration and exploitation - GA will perform better than SA
- Hypothesis 3: MIMIC will outperform both four peaks and flip flop problem.

4.2. Flip Flop Problem

4.3. NN Weight Optimization

- Hypothesis 1: Simmulated Annealing should perform better than Random Hill Climbing.
- Hypothesis 2: Genetic Algorithm should outperform other optimization strategies

5. Experiment

The three mentioned problems are evaluated using RHC, SA, GA and MIMIC. For RHC we tuned on multiple restart and random state values, to see if this affects the convergence of the problem. For SA we tune initial temperature, decay rate, cooling schedule, minimum temperature and exponential constants. For GA we tune population size and genetic algorithm and for MIMIC we also tune population size and percent of population kept at each iteration.

5.1. Four Peaks Problem

We've tuned hyperparameters for problem size of length 20 for all strategies. The initial random state can have a significant impact

Restarts	Random Value	Fitness
0	42	2
0	101	6
20	303	32
5	101	26
5	202	25

Table 1. RHC Parameter Tuning For Four Peak Evaluation

on the fitness value of RHC, especially with fewer restarts. This is because the algorithm's performance heavily depends on the starting point in the search space. As the number of restarts increases, the effect of the random state diminishes, and the algorithm consistently finds high-quality solutions regardless of the initial random state.

For SA, we found that all values of parameters gave the same fitness value (18) for the problem length of 20, and we were not able to improve upon the problem further, indicating it might not be well-suited for this particular problem.

Pop Size	Mutation Prob	Fitness
50	0.1	31
50	0.2	36
50	0.3	35
100	0.1	32

Table 2. GA Parameter Tuning For Four Peak Evaluation

In GA, a population size of 50 provides a sufficient pool of solutions for the GA to work with, allowing it to effectively explore and exploit the search space. A mutation probability of 0.2 provides enough variation to escape local optima without disrupting the good solutions too much. Too low a mutation probability limits exploration, while too high a mutation probability disrupts good solutions. When tuning for MIMIC, we see that a larger population size provides a broader pool of solutions, enhancing the MIMIC algorithm's ability to explore the search space and effectively sample and retain good solutions, improving overall fitness. A keep percentage of 0.20 at population size of 200 balances the need to retain high-quality solutions while allowing enough diversity. Too low or too high keep percentages can either lose good solutions or reduce diversity, leading to sub-optimal performance.

Pop Size	Keep Pct	Fitness
200	0.2	36
200	0.1	34
100	0.1	32
100	0.3	30
100	0.05	28
50	0.3	31
50	0.2	10

Table 3. MIMIC Parameter Tuning For Four Peak Evaluation

5.2. Flip Flop Problem

Fitness values for different RHC parameters are presented in table 4. We observe that without restarts, the algorithm's performance is highly sensitive to the initial random state, leading to significant variance in the fitness values. With more restarts, not only this sensitivity is reduced, but the algorithm performs better more consistently with lower bias as it allows the algorithm to escape local optima and explore more thoroughly.

Restarts	Random Value	Fitness
0	42	12
0	101	14
0	202	10
5	42	17
5	101	16
10	303	19

Table 4. RHC Parameter Tuning For Flip Flip Evaluation

While tuning SA parameters we found that there's no fitness value improvement for different values of Initial temperature (taken:1), decay rate (taken:0.95) and minimum temperature (taken: 0.001). Moreover it is observed that the exponential decay schedule performs best with a smaller decay constant as the exponential decay schedule with a smaller decay constant offers robustness against the variations in initial temperature and decay rate because it ensures a controlled cooling process. The gradual decrease in temperature with a smaller decay constant provides a better balance between exploration and exploitation phases, leading to improved fitness outcomes.

Cooling Schedule	Exp Const	Fitness
ExpDecay	0.001	19
ExpDecay	0.100	18
ArithDecay	0.001	17

Table 5. SA Parameter Tuning For Flip Flip Evaluation

Pop Size	Mutation Probability	Fitness
50	0.1	18
50	0.2	19
50	0.3	18
100	0.1	18
200	0.1	18

Table 6. GA Parameter Tuning for Flip Flop

In Genetic algorithm, the best fitness value (19) is achieved with a population size of `50` and a mutation probability of `0.2` because these parameters effectively balance exploration and exploitation. The population size ensures sufficient diversity to explore the search space, while the mutation probability introduces new alternating bit pairs without disrupting the existing good solutions. This balance helps the genetic algorithm to navigate the numerous local optima and find solutions with a high number of alternating pairs.

Pop Size	Keep Pct	Fitness
50	0.05	18
50	0.1	17
50	0.5	19
100	0.1	18
100	0.2	17

Table 7. MIMIC Parameter Tuning for Flip Flop

When tuning MIMIC it is found that a higher keep percentage (`0.5`) results in the highest fitness value (19) for a population size of `50`. Lower keep percentages (`0.05` and `0.1`) lead to lower fitness values, indicating that retaining a larger portion of the population may help maintain good solutions and guide the search more effectively. Larger population sizes with higher keep percentages do not lead to better fitness values, suggesting that increasing the population size alone is not sufficient to improve performance.

5.3. Neural Network Weight Optimization Problem

Different Hyperparamters tuned for each optimization algorithm used in the back-propagation algorithm to get the best parameters. Similarly, MLP with adam solver that was used for prediction previously is also re-tuned and re-trained for comparison. Most promising fitness values for each algorithm is presented in table 1, 2 and 3 respectively.

Hidden Nodes	Restarts	learning Rate	Fitness
(5,)	5	0.05	0.67
(10, 5)	5	0.1	0.62
(10, 5)	10	0.1	0.73
(10, 10)	10	0.1	0.69

Table 8. Hyperparameter Tuning for RHC

For RHC, the highest fitness achieved was 0.73 with the (10, 5) configuration, 10 restarts, and a learning rate of 0.1. Simpler models (e.g., (5,) with 5 restarts) are likely to have higher bias because they are not complex enough to capture all the patterns in the data. This is evident from the fitness score of 0.67, which is decent but not the best. The improvement in fitness from 0.62 to 0.73 when increasing restarts from 5 to 10 for the (10, 5) model suggests that the model was initially overfitting (high variance), and increasing restarts helped in finding a more general solution (reduced variance).

Hidden Nodes	Schedule	learning Rate	Fitness
(5)	GeomDecay	0.1	0.46
(10, 5)	GeomDecay	0.01	0.5
(10, 5)	ExpDecay	0.05	0.5
(10, 10)	ExpDecay	0.1	0.42

Table 9. Hyperparameter Tuning for SA

While tuning SA, the two-layer models with appropriate learning rates (0.01 and 0.05) achieve higher fitness (0.5), indicating reduced bias as the model complexity increases. The consistent performance of the two-layer (10, 5) models with different schedules but appropriate learning rates (0.5 fitness) suggests a balanced approach, reducing variance while maintaining complexity. Additionally, cross-validation could help in better understanding the trade-off between bias and variance.

Hidden Nodes	Population Size	Mutation Prob	Fitness
(10, 10)	50	0.2	0.66
(10, 10)	100	0.1	0.71
(10, 10)	200	0.1	0.70

Table 10. Hyperparameter Tuning for GA

The configurations with a population size of 100 and 200 and a mutation probability of 0.1 demonstrate low bias. The GA can adequately explore the solution space and combine partial solutions effectively, leading to high fitness.

The low mutation probability helps in fine-tuning the solutions rather than making drastic changes, reducing bias while the higher mutation rate introduces more randomness, which can lead to fluctuations in fitness and less stable convergence due to higher variance.

6. Results

6.1. Optimization Problem

Both four peak problem and flip problem were then evaluated with all optimization strategies to understand which one performs best.

6.1.1. Four Peak Problem

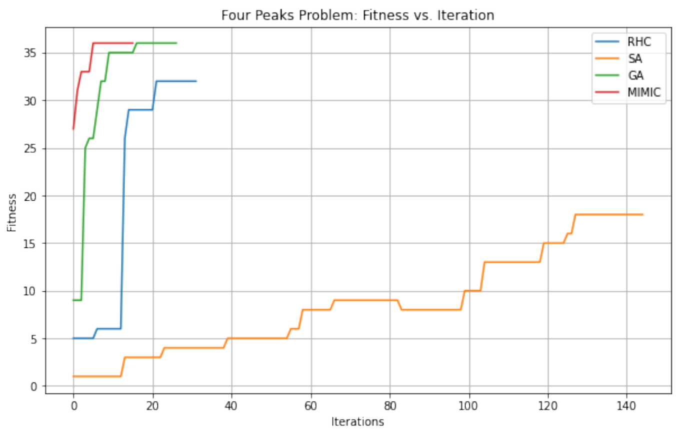

Figure 4. Four Peak Problem: Fitness vs Iteration

For four peak, all algorithms exhibited some degree of convergence, reaching a fitness plateau within their allocated iterations. However, the level of convergence varied. RHC's rapid convergence suggests getting stuck in a local optimum, while the others likely haven't fully converged within the given iterations. RHC has a high bias due to its greedy nature, and low variance at a higher restart time as then it does not depend on starting point.

GA significantly outperforms SA in terms of fitness for the Four Peaks problem, as GA is able to effectively balance exploration and exploitation through its population-based approach and genetic operators (crossover and mutation).

Strategy	Fitness	Iteration	Fevals	Time
RHC	32	32	350	0.005
SA	18	145	230	0.004
GA	36	27	1434	0.072
MIMIC	36	16	3420	1.872

Table 11. Four Peak: Optimization Strategy evaluation

RHC and SA were the fastest in terms of wall clock time and iterations. GA required more evaluations and time due to its population-based approach. MIMIC was the slowest, likely due to its complex internal mechanisms.

GA is preferable over MIMIC,as although MIMIC's iterations are lower than GA to the same fitness score, the time taken and the number of functions evaluations are quite high for MIMIC as it generates a large number of samples in each iteration.

Interestingly, for problem size as large as 200, SA outperformed GA for the four peaks problem. We observed that SA can more effectively explore the vast search space by gradually cooling and fine-tuning the solution as the problem size grows. Because the search space grows exponentially with problem size GA's population-based approach might struggle to maintain diversity and adequately explore the larger search space without a corresponding increase in population size.

Although MIMIC converged to optima in lesser iterations for lower problem size, the performance drop of MIMIC for larger problem sizes can be attributed to the hyperparameters being tuned for smaller problems. These hyperparameters may not scale effectively, leading to inefficient exploration and sampling for larger problems. In contrast, GA's operations showed more scalability and lesser sensitivity to problem size as compared to MIMIC.

6.1.2. Flip Flop Problem

RHC: Achieves a fitness of 17, which indicates it often gets trapped in local optima. **SA, GA, MIMIC**: All achieve a fitness of 19, indicating they are more effective at finding near-optimal solutions compared to RHC.

Strategy	Fitness	Iteration	Fevals	Time
RHC	17	35	73	0.010
SA	19	285	4240	0.029
GA	19	17	919	0.0945
MIMIC	19	16	767	1.950

Table 12. Flip Flop: Optimization Strategy evaluation

For the Flip-Flop problem at a size of 20, SA required more function evaluations (4240) but much lesser time compared to GA. While SA's iterations are higher, the probabilistic nature of its search can lead to faster evaluation. This indicates that SA can be more efficient in terms of the total time needed to reach near-optimal solutions, which is a significant advantage when evaluation cost is high.

GA is robust due to its population-based approach, which allows it to explore multiple areas of the solution space simultaneously.

MIMIC is robust but highly sensitive to the number of samples and the method used to update the probabilistic model. It can be computationally expensive.

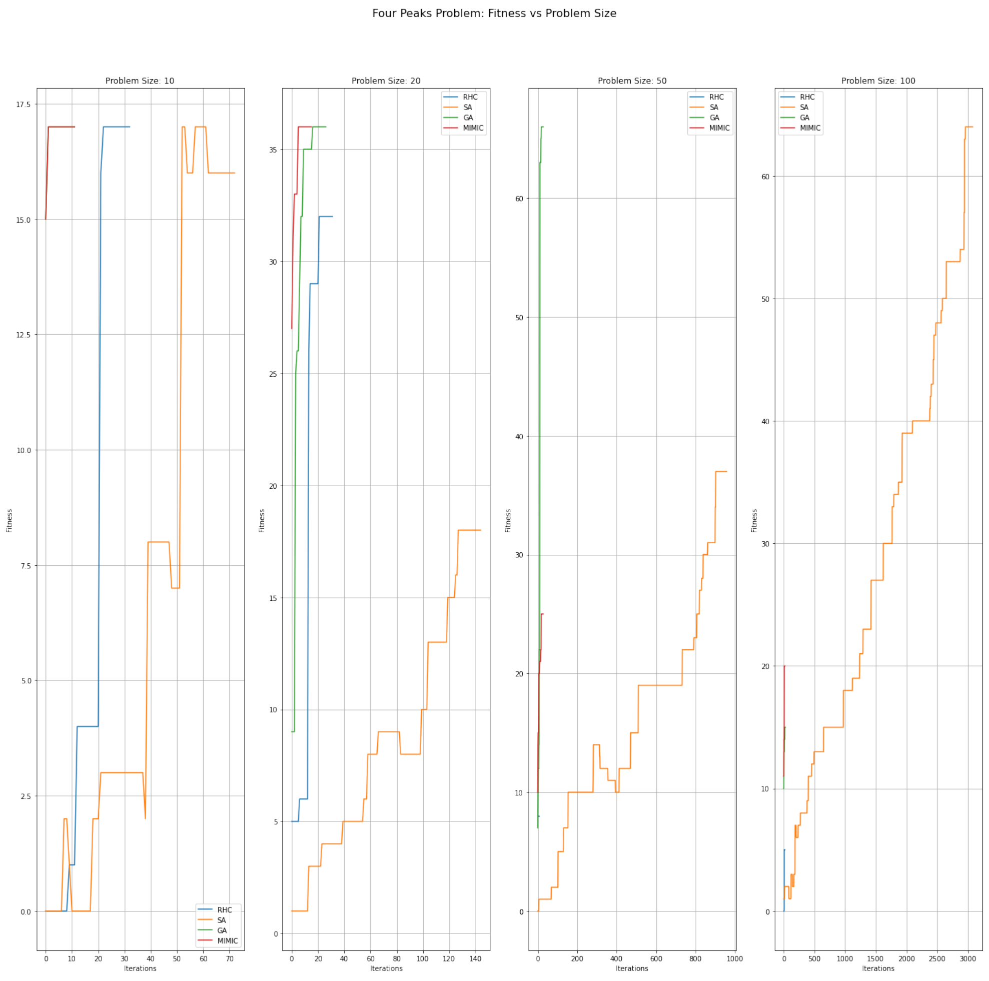

Figure 5. Four Peak Problem: Fitness vs Problem Size

SA has lower bias compared to RHC due to its ability to explore more of the solution space, but this comes with higher variance as the results can vary significantly between run, whereas GA has moderate bias and variance. The population-based search reduces the risk of getting stuck in local optima. **MIMIC**, similar to GA in terms of bias and variance, but its model-based approach leads to high variance as the model is not well-calibrated for all problem sizes.

When testing fitness for problem sizes of 20, 50, 100, and 200 as shown in figure 6:

- **SA** reaches the optimal solution for problem sizes 20 and even exceeds GA and MIMIC for problem sizes 50, indicating its effectiveness in these scenarios.
- For a problem size of 100 and 200, **GA** outperforms **SA** and **MIMIC**, likely due to SA's increasing computational cost and difficulty in maintaining an effective cooling schedule over a larger search space

6.2. Machine Learning Weight Optimization

The final metrics comparison on the tuned hyperparameters is shown in table 13. Both RHC and GA methods achieve moderate performance but are less efficient compared to the MLP-adam solver. They take significantly longer to converge and do not reach as high accuracy or F1-Score. GA, in particular, has a very high wall clock time.

Optimization	Accuracy	F1-Score	Time	Loss
MLP-adam solver	0.871	0.871	1.6	0.28
RHC	0.726	0.729	85.7	0.55
SA	0.50	0.33	8.29	1.14
GA	0.73	0.71	536	0.49

Table 13. Neural Network: Optimization Strategy Performance

SA performs the worst among all optimization methods, indicating that it is not suitable for this specific neural network and dataset combination. It has low accuracy, low F1-Score, and high loss, suggesting that it struggles to find good solutions likely due to its probabilistic nature and difficulty in maintaining effective exploration and exploitation balance for the neural network training.

From figure 8, it is observed that RHC gets stuck in a local optimum, and repeatedly generates new solutions that are not better than the current solution. Once the hyperparameters (e.g., step size, number of iterations) are set, the extent of the search space exploration is fixed. If the neighborhood of the current solution does not contain better solutions, the loss will not change.

We see constant bad performance for SA even with different hyperparameters, this poses a question - Is our binary classification problem challenges the way SA optimizes weights?

Gradient-based methods like the Adam solver can optimize more efficiently on both shallow and deep networks because they use gradient information to guide the search. SA, which relies on probabilistic acceptance and doesn't use gradient information, might need a more complex network to find regions in the weight space where significant improvements can occur.

To test this hypothesis, we increase the complexity by taking (10, 5, 10) as our network configuration for SA (after exploring other combinations). Additionally, we also adjusted the initial temperature to 5 to ensure adequate exploration of the weight space in deeper networks.

We observed that the **accuracy has improved to 0.59** and loss as

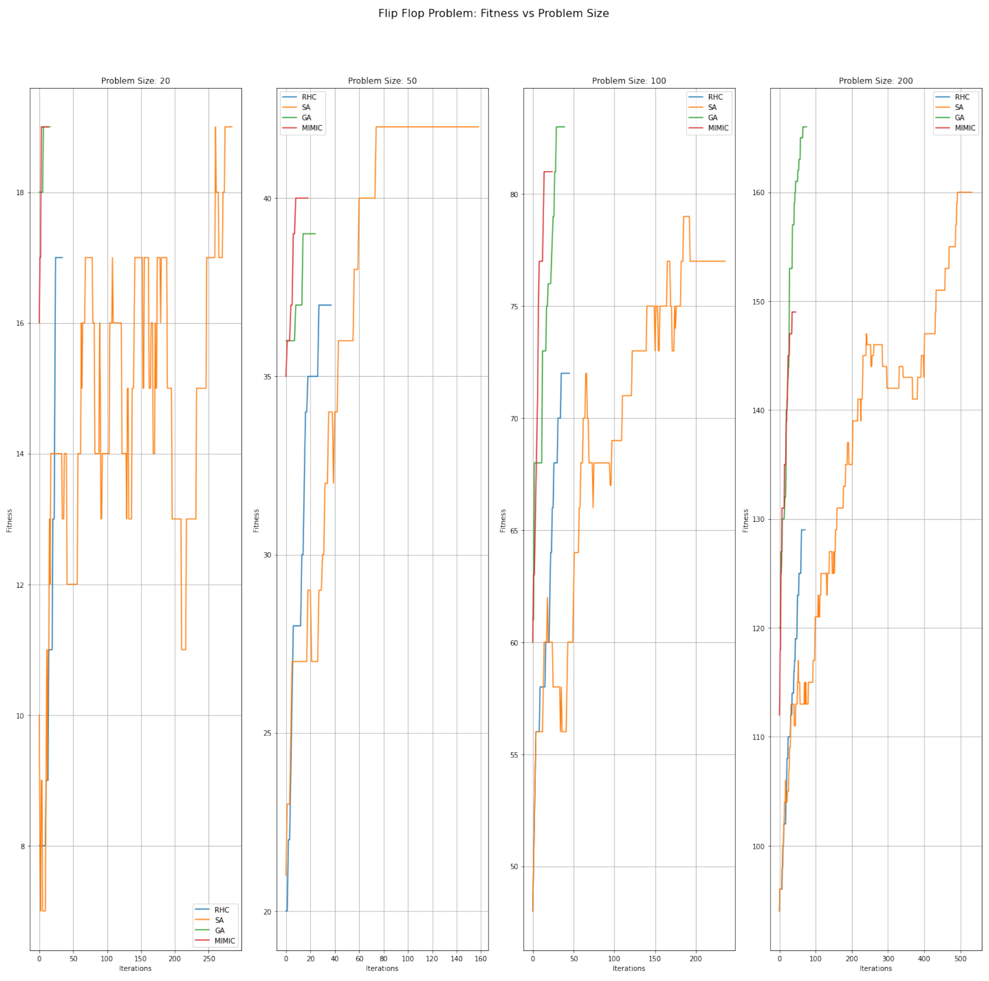

Figure 6. Flip Flop: Fitness vs Problem Size

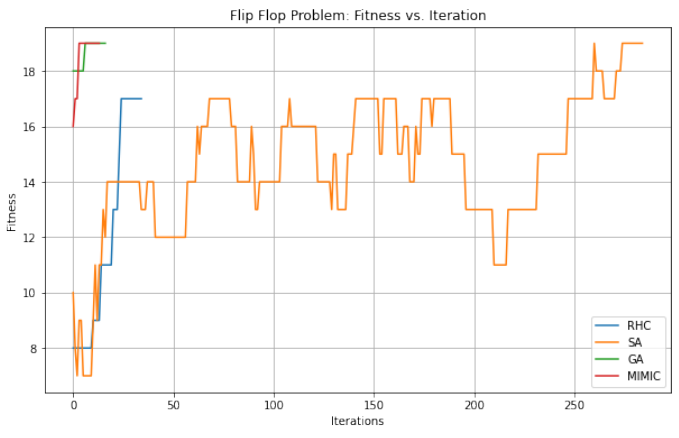

Figure 7. Flip Flop: fitness vs iteration

decreased to 1.08. Though it still does not surpass RHC performance on our dataset disproving our claimed hypothesis.

7. Conclusion

The observed performance differences between SA and GA across different problem sizes for the Flip Flop and Four Peaks problems highlight the nuanced strengths and weaknesses of each algorithm. While SA excels in smaller and moderately sized problems for the Flip Flop problem, it struggles with larger problem sizes due to the increasing computational cost and difficulty in maintaining an effective cooling schedule. Conversely, GA's population-based approach

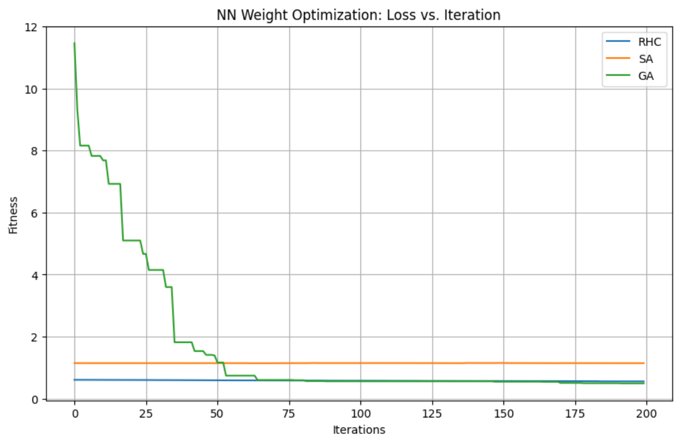

Figure 8. Loss vs Iteration for all Optimizers

and genetic operators make it more effective in larger search spaces, leading to better performance in the Flip Flop problem for sizes 100 and 200.

For the Four Peaks problem, the initial hypothesis is flipped for larger problem sizes, where SA outperforms GA. SA's probabilistic acceptance of worse solutions and gradual cooling schedule allow it to navigate the larger search space with many local optima more effectively. GA, on the other hand, faces challenges in maintaining long sequences required for the bonus in the Four Peaks problem as the problem size increases.

Overall, the performance of SA and GA is highly dependent on the specific characteristics of the problem and the problem size, emphasizing the importance of adaptive hyperparameter tuning and a deep understanding of the problem landscape.

In our weight optimization problem, the MLP-adam solver's performance indicates that gradient-based methods are well-suited for this problem. In contrast, SA struggled due to their local search nature, and GA's performance was limited by its computational cost. Based on Accuracy, Loss and Wall Clock Time, **RHC** performed the best for our dataset.

While we were able to slightly improve the performance of SA with slower cooling schedules and higher initial temperatures, further experimenting with different neural network architectures (e.g., varying hidden layers and nodes) could provide insights into the optimal configuration for this dataset. Cross-validation would provide a more robust evaluation of the algorithms by ensuring that the performance is consistent across different subsets of the data.

References

[1] P. V. Jeremy S Charles L, *Mimic: Finding optima by estimating probability densities*, 1997.

[2] F. M. Shoaib Khanmohammadi Onder Kizilkan, *Multiobjective optimization of a geothermal power plant*, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128210376000111.

[3] *Evolutionary learning*. [Online]. Available: http://vda.univie.ac.at/Teaching/ML/14s/LectureNotes/12_Evolutionary%20Learning.pdf.

[4] *Introduction to hill climbing | artificial intelligence*. [Online]. Available: https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/.

[5] *What is simulated annealing?* [Online]. Available: https://www.mathworks.com/help/gads/what-is-simulated-annealing.html.