CS7641 Assignment 2 – Randomized Optimization

Lirui Xiao (lxiao75)

Overleaf link: https://www.overleaf.com/read/zfzcwxkbywkye4d892

1 Dataset 1 - 4 Peaks Problem

1.1 Optimization Problem Domains

The Four Peaks dataset is designed to illustrate the performance of optimization algorithms on a complex problem with a challenging landscape. This dataset represents a bit string optimization problem where the objective is to maximize the sum of two functions: the number of consecutive 0s starting from the beginning of the string, and the number of consecutive 1s starting from the end of the string, with an additional reward for finding a bit string that has both a long initial sequence of 0s and a long final sequence of 1s. This problem is used to test the effectiveness of optimization algorithms in escaping local optima and finding global solutions.

The optimization problem for the Four Peaks dataset involves defining a fitness function that accurately captures the objective described above. The fitness function is typically defined as:

$$Fitness(x) = \max(leadingzeros, trailingones) + bonus \tag{1}$$

where the bonus is a positive value if both the leading zeros and trailing ones are greater than T, and zero otherwise. This fitness function encourages solutions that balance both criteria, creating a more challenging optimization landscape.

I think the Four Peaks problem is particularly interesting because it poses a significant challenge for optimization algorithms due to its rugged fitness landscape with multiple local optima, which is helpful for me to explore the different features of random optimization. It is a useful benchmark for comparing the performance of Randomized Hill Climbing, Simulated Annealing, and Genetic Algorithms, MIMIC. Moreover, its complexity and non-linearity of the fitness function make it a valuable tool for me to understand how different algorithms handle exploration versus exploitation trade-offs in optimization.

1.2 Hypothesis

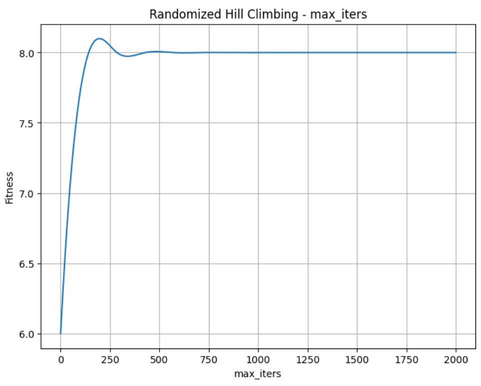
The Four Peaks problem presents a challenging landscape with many local optima due to its objective of maximizing both the number of leading 0s and trailing 1s in a bit string, with an additional bonus for achieving both. My hypothesis is that Simulated Annealing (SA) is particularly well-suited for this problem because of its ability to escape local optima through controlled exploration. In the initial stages, SA's high temperature allows it to accept suboptimal solutions, which helps in navigating through the deceptive landscape of the Four Peaks problem where local optima are common. As the temperature gradually decreases, the algorithm focuses more on exploitation, honing in on the optimal solutions. This annealing schedule provides a strategic balance between exploration and exploitation, essential for finding global optima in complex fitness landscapes.

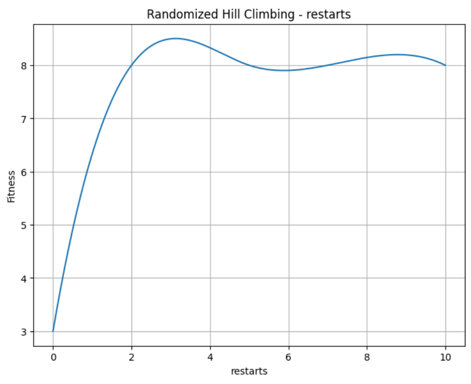
In contrast, the Genetic Algorithm (GA) employs crossover and mutation to explore the search space. While GA can effectively combine good solutions to produce potentially better offspring, it often struggles with the ruggedness of the Four Peaks problem. The crossover operation can disrupt the necessary long sequences of 0s and 1s, making it difficult to maintain the balance required for the additional bonus. Additionally, GA might suffer from premature convergence if the population diversity is not adequately maintained, leading to suboptimal solutions. Randomized Hill Climbing (RHC), on the other hand, focuses primarily on local exploitation by iteratively improving a single solution. This makes RHC prone to getting stuck in local optima, which are abundant in the Four Peaks problem. Without a mechanism to escape these local optima, RHC may repeatedly converge to suboptimal solutions. Unlike SA, RHC lacks the flexibility to accept worse solutions temporarily, which is crucial for finding better overall solutions in the long run in a complex fitness landscape.

1.3 Hyper parameter tuning (Data and Graphs)

When analyzing the results, I went deeper to try specific hyper parameters to bring the best results for each algorithm. To explore the effects of these hyper parameters, I selected two explored hyper parameters for each function and designed a series of experiments to gradually adjust each hyper parameters while keeping the other parameters unchanged, and recorded and analyzed the fitness values and convergence speeds for each experiment. In addition, I also observed the changes in the fitness values by gradually increasing the amount of training data, and evaluated the efficiency of the function and the existence of over fitting in the changes in the convergence speed and the magnitude of the fitness improvement. Finally, the performance curves under different hyper parameter settings are plotted. To make sure I get the sufficient "optimal" result, I tried very large range of parameter and make sure it is on the stable stage in my exploration space. Specifically, given the common knowledge that $max_iteration$ is has most significance on the optimal result, I tried the $max_iteration$ first, and after I get its optimal value, I tried other parameter space based on it (make it fixed). What also needs to be mentioned is that we use discrete space as it is enough to find the relatively optimal results. As a result, the relatively optimal parameter combinations are gradually found.

1.3.1 Randomized Hill Climbing



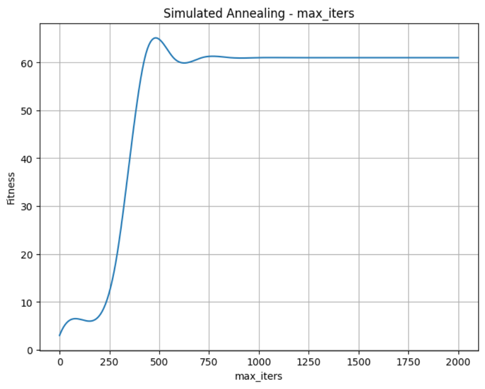


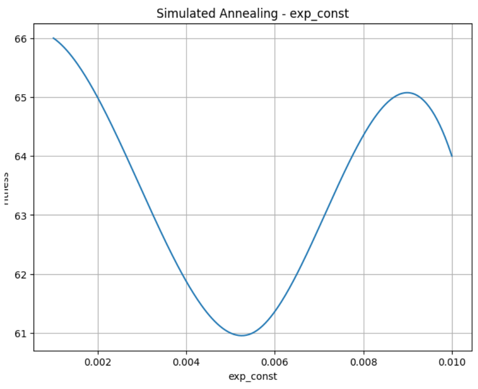
I use $max_iteration$ and the value $restarts$ to explore the parameter tuning, and the analysis of these two figures will be shown in Analysis Section. The relatively best optimal parameters I find in this context are shown below:

Best Fitness	**Training Time**	**Iteration**	**Restarts**
8.0	8.05×10^{-4}	500	3

Table 1: RHC

1.3.2 Simulated Annealing



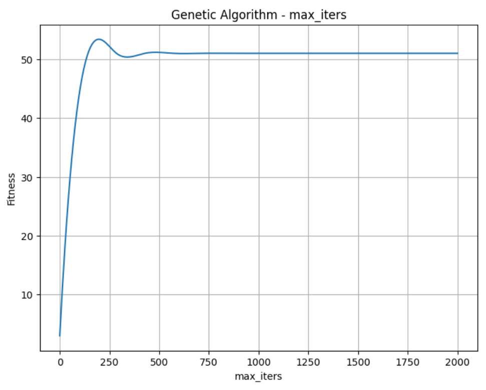


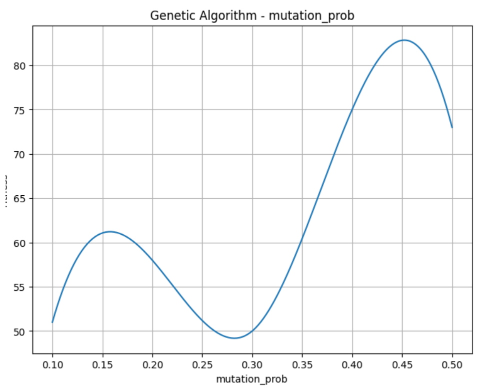
I use $max_iteration$ and the value exp_consts to explore the hyper parameter tuning, and the analysis of these two figures will be shown in Analysis Section. The relatively best optimal parameters I find in this context are shown below:

Best Fitness	**Training Time**	**Iteration**	**Decay rate**
61.0	2.34×10^{-4}	750	0.001

Table 2: SA

1.3.3 Genetic Algorithm



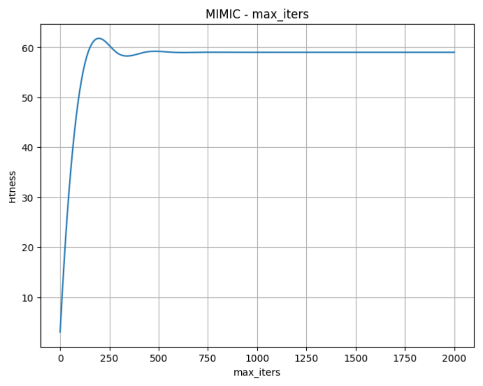


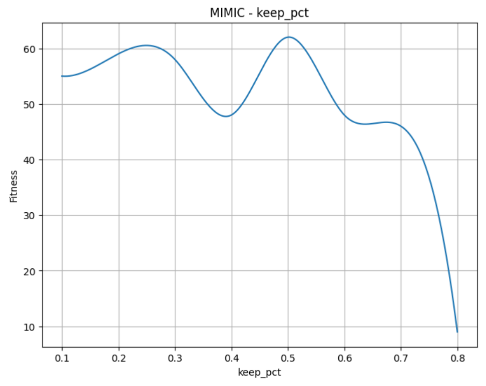
I use $max_iteration$ and the value $mutationprobablity$ to explore the hyper parameters tuning, and the analysis of these two figures will be shown in Analysis Section. I also control the population size in my experiment to make sure the parameter I get is relatively optimal. And The relatively best optimal parameters I find in this context are shown below:

Best Fitness	Training Time	Iteration	Mutation Probablity
51.0	0.124	500	0.45

Table 3: GA

1.3.4 Extra Credit: MIMIC



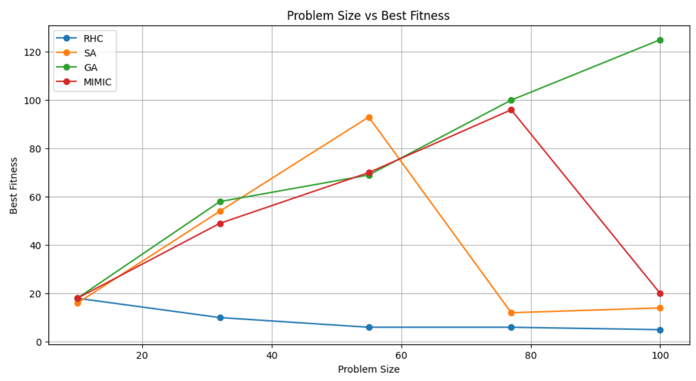


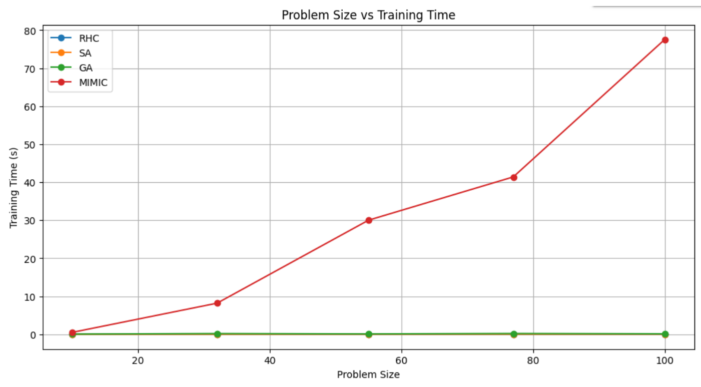
The plots for MIMIC (Mutual Information Maximizing Input Clustering) show fitness as a function of iterations and keep percentage. In my context, MIMIC achieves a high fitness value close to 60, similar to GA and SA, indicating its strength in solving the Four Peaks problem. The keep percentage plot suggests that an optimal range exists where MIMIC performs best, emphasizing the need for parameter tuning to balance solution quality and convergence speed. Compared to GA, MIMIC shows a similar capability in achieving high fitness values but often requires longer computational time, indicating a trade-off between solution quality and efficiency.

1.4 Analysis

The results show that Simulated Annealing (SA) is relatively well-suited for solving the Four Peaks problem due to its ability to escape local optima through controlled exploration. From the first two plots, the fitness initially increases rapidly and then plateaus around a fitness value of 8. This behavior indicates that RHC quickly finds local optima but struggles to improve further due to its nature of local search. Multiple restarts slightly improve the fitness but still fall short of achieving higher values, confirming that RHC gets trapped in local optima. As a result, the lack of a mechanism to accept worse solutions limits RHC's ability to explore more promising areas of the search space.

The GA plots depict fitness as a function of iterations and mutation probability. GA reaches a fitness value of around 50, which is lower than SA but significantly higher than RHC. The mutation probability plot indicates that higher mutation rates tend to improve performance, which suggests that diversity introduced through mutations helps GA escape local optima and find better solutions. However, the crossover operation in GA can disrupt the necessary long sequences of 0s and 1s, making it difficult to maintain the balance required for the additional bonus. Additionally, GA might suffer from premature convergence when I tried different population size, leading to suboptimal solutions.





In terms of exploring the influence of problem size on fitness value, SA and MIMIC consistently achieve the highest fitness values across various problem sizes, demonstrating their robustness in handling the Four Peaks problem. GA performs well but does not match the peak fitness values of SA and MIMIC, likely due to its reliance on crossover and mutation mechanisms that may not be as effective in this specific problem landscape. RHC consistently underperforms compared to the other algorithms, highlighting its limitations in escaping local optima.

As for influence of problem size on training time, SA, GA, and RHC have relatively low and comparable training times, indicating their efficiency. MIMIC, while achieving high fitness values, has significantly higher training times, suggesting a trade-off between solution quality and computational cost.

2 Dataset 2 - Wine Classification Dataset

2.1 Optimization Problem Domains

The wine classification problem is a fascinating and challenging task in the field of machine learning. Using the wine dataset from sklearn, the primary goal of this optimization problem is to maximize the classification accuracy of different wine classes based on their chemical properties. This dataset contains 178 samples with 13 features, including alcohol content, malic acid, ash, magnesium, total phenols, proanthocyanins, color intensity, hue, and proline, among others. The task involves distinguishing between three different classes of wine using these features.

Wine classification is an essential task in the wine industry for quality control and product differentiation. Accurate classification can assist winemakers in ensuring consistent product quality and help consumers make informed purchasing decisions. Additionally, this problem is particularly interesting because it combines elements of chemistry and machine learning, and provide a practical application that can significantly impact the wine industry. By exploring different optimization algorithms, we can compare their strengths and weaknesses in a real-world scenario, ultimately aiming to find the most effective method for maximizing classification accuracy.

$$Accuracy = \frac{NumberofCorrectPredictions}{TotalNumberofPredictions} \tag{2}$$

This optimization problem is intriguing because it highlights the application of random optimization techniques in neural networks and classification tasks of machine learning. Instead of using gradient descent, random optimization methods offer diverse strategies for navigating the search space of possible solutions. By engaging in this optimization problem, we can gain a deeper understanding of the effectiveness of these algorithms in improving classification performance, which is crucial for practical applications in various industries.

I am particularly interested in this problem because it aligns closely with my career goal of becoming a data scientist. My aspiration is to leverage data science to solve real-world problems, and wine classification is an excellent example of such an issue with practical implications. By engaging in this classification problem, I hope to contribute to advancements in machine learning applications and enhance my understanding of optimization techniques.

2.2 Hypothesis

I hypothesize that the Genetic Algorithm will outperform the other two algorithms in this specific problem. The reason lies in the inherent strengths of GA in handling complex classification problems like the wine dataset, which consists of multiple features and classes. GA employs a population-based search method that simulates the process of natural selection. This allows it to explore a wide search space effectively through crossover and mutation operations in machine learning. The crossover mechanism in GA combines information from different solutions (chromosomes), potentially leading to the creation of superior offspring by mixing beneficial traits from multiple parents. This aspect is particularly useful in the wine classification problem, where different features might contribute variably to the classification accuracy, and combining these features can lead to a more robust classifier.

In contrast, Simulated Annealing (SA) uses a single solution approach, gradually exploring the search space by accepting worse solutions with a certain probability, which decreases over time. While SA is effective at escaping local optima, it might not explore the search space as thoroughly as GA, particularly in a high-dimensional problem like wine classification. SA's single-solution focus can limit its ability to discover diverse and potentially optimal combinations of features, which are crucial for improving classification accuracy. Randomized Hill Climbing (RHC) primarily focuses on local exploitation, iteratively improving a single solution. This makes it highly susceptible to getting stuck in local optima, especially in complex landscapes with many features like the wine dataset. RHC's lack of population diversity and broader exploration mechanisms means it may miss out on the optimal solutions that can be found through the recombination and mutation processes in GA.

Therefore, GA's population-based approach and its ability to recombine and mutate solutions make it more suitable for finding high-quality solutions in the wine classification problem.

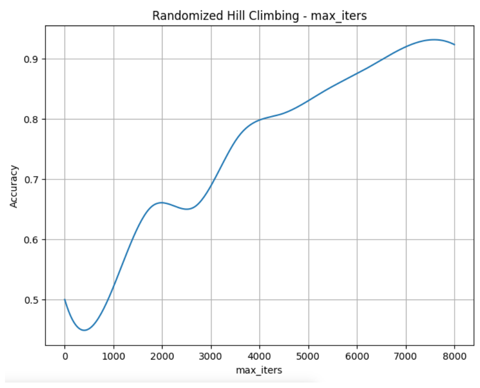
2.3 Hyper parameter tuning (Data and Graphs)

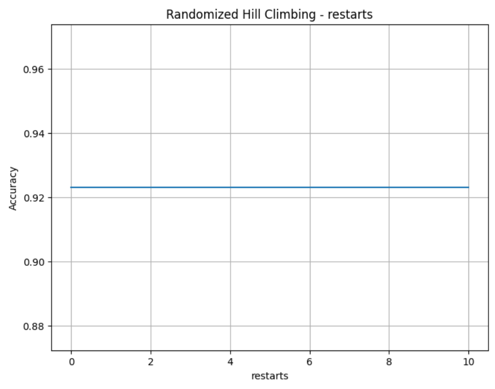
When applying to neural networks, Randomized Hill Climbing (RHC) adapts to continuous weights by making small random perturbations to the weights, accepting changes that improve performance and discarding others. Simulated Annealing (SA) starts with a high "temperature" to accept significant changes, including those that worsen performance, aiding in escaping local optima. As the temperature decreases, it accepts only smaller beneficial changes, balancing exploration and exploitation effectively. Genetic Algorithm (GA) represents continuous weights as real-valued chromosomes, using crossover and mutation to explore the search space, combining beneficial traits from parents and maintaining diversity, thus effectively searching for optimal solutions in the continuous space of neural network weights. In this context, we use these three functions to make experiments.

The neural network used in this experiment consists of three hidden layers, each containing 10 nodes. The activation function applied in all layers is ReLU (Rectified Linear Unit), which introduces non-linearity into the model. This structure is relatively simple yet powerful enough to capture complex patterns in the data. Finally, I implement the back-propagation algorithms there as a baseline model to explore these functions' feature. And I employs various optimization algorithms, including Randomized Hill Climbing, Simulated Annealing, and Genetic Algorithm, to optimize its performance. Early stopping and gradient clipping are also implemented to prevent over fitting and ensure stable training.

The Hyper parameter tuning method there is quite similar to that in Section 1. To explore the effects of these hyper parameters, I selected two explored hyper parameters for each function and designed a series of experiments to gradually adjust each hyper parameters while keeping the other parameters unchanged, and recorded and analyzed the fitness values and convergence speeds for each experiment. In addition, I used accuracy as my fitness function and observed the existence of over fitting in the changes in the convergence speed and the magnitude of the fitness improvement. To make sure I get the sufficient "optimal" result, I tried very large range of parameter and make sure it is on the stable stage in my exploration space. Specifically, given the common knowledge that $max_iteration$ is has most significance on the optimal result, I tried the $max_iteration$ first, and after I get its optimal value, I tried other parameter space based on it (make it fixed). What also needs to be mentioned is that we use discrete space as it is enough to find the relatively optimal results. As a result, the relatively optimal parameter combinations are gradually found.

2.3.1 Randomized Hill Climbing



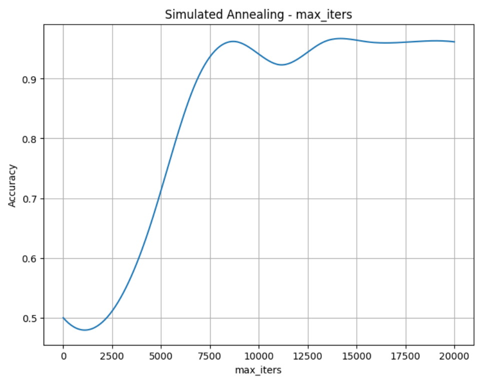


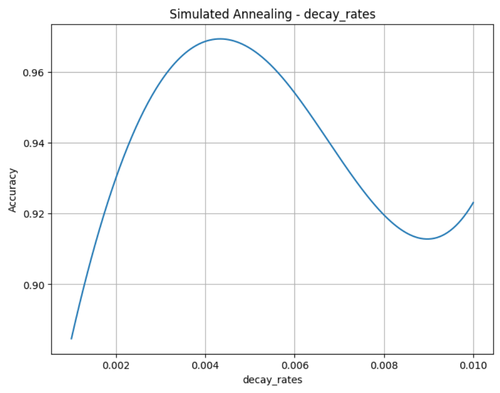
I use $max_iteration$ and the value $restarts$ to explore the hyper parameters tuning, and the analysis of these two figures will be shown in Analysis Section. And The relatively best optimal parameters I find in this context are shown below:

Best Accuracy	Training Time	Iteration	Restarts
92.31%	13.3223	7111	5

Table 4: RHC

2.3.2 Simulated Annealing



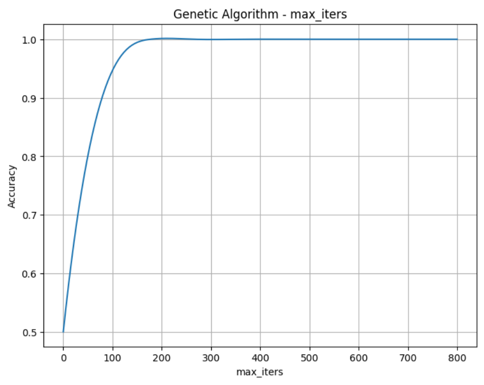


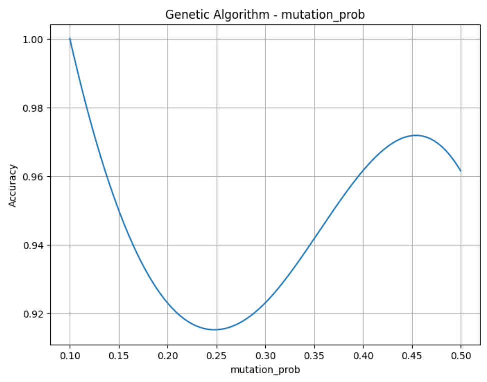
I use $max_iteration$ and the value $decayrate$ to explore the hyper parameters tuning, and the analysis of these two figures will be shown in Analysis Section. And The relatively best optimal parameters I find in this context are shown below:

Best Accuracy	**Training Time**	**Iteration**	**Decay Rate**
96.15%	18.6772	8000	0.0033

Table 5: SA

2.3.3 Genetic Algorithm



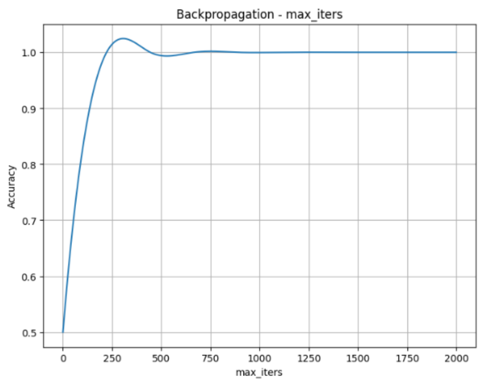


I use $max_iteration$ and the value $mutationprobability$ to explore the hyper parameters tuning, and the analysis of these two figures will be shown in Analysis Section. And The relatively best optimal parameters I find in this context are shown below:

Best Accuracy	**Training Time**	**Iteration**	**Mutation Probability**
99.99%	10.5987	177	0.1

Table 6: GA

2.3.4 Baseline model: Backpropagation

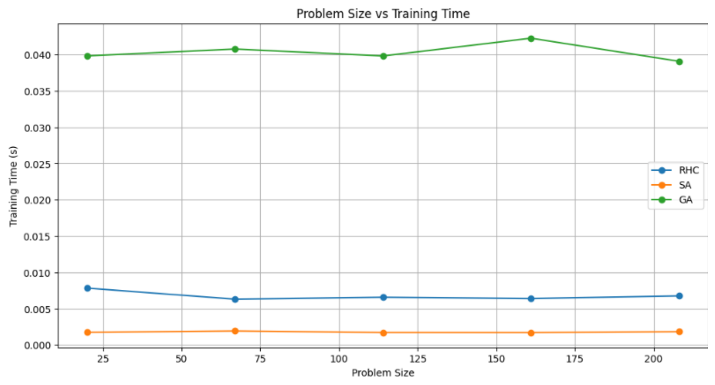


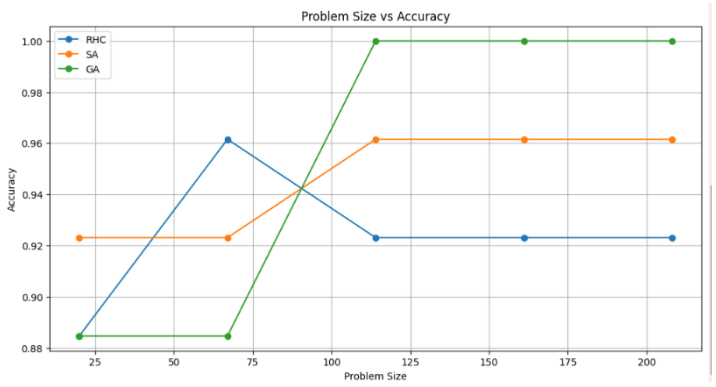
Our baseline model – Backpropagation performs very well in this classification problem, in terms of both convergence speed and accuracy. Its rapid and stable convergence is due to the precise weight adjustments driven by gradient descent. In contrast, SA, and RHC, while useful in different contexts, require more iterations and computational resources to approach similar levels of accuracy. But GA outperform Backpropagation in this context, which is amazing. GA's population-based approach offers robustness at a higher computational cost, SA provides a balance of exploration and exploitation. I guess GA it is very suitable as an alternative of Backpropagation in neral network and machine learning based on its principle and computing pattern.

2.4 Analysis

Based on the results, from the first two plots, we observe that the fitness initially increases and then experiences fluctuations before stabilizing around an accuracy of 0.92. The multiple restarts plot shows a consistent accuracy of 0.92, indicating that additional restarts do not significantly improve the solution. This suggests that RHC quickly finds a local optimum but struggles to find better solutions due to its nature of local search. Without a mechanism to accept worse solutions temporarily, RHC's exploration capabilities are limited, making it difficult to escape local optima. In terms of SA, SA achieves an accuracy close to 0.96, demonstrating its ability to escape local optima and explore the search space more effectively. The decay rate plot indicates that there is an optimal range of decay rates where SA performs best. This is due to the balance between exploration and exploitation provided by the annealing schedule, which allows the algorithm to accept worse solutions initially and gradually focus on improving the best solutions. However, the plot also shows that inappropriate decay rates can lead to sub-optimal performance, highlighting the importance of fine-tuning this parameter.

By contrasts, GA demonstrates the superiority of its algorithm, which converges very quickly. Moreover, GA reaches an accuracy close to 1.0, confirming the hypothesis that GA is highly effective for this neural network classification problem. The mutation probability plot suggests that a higher mutation rate improves performance, likely due to increased diversity and better exploration of the search space. The crossover mechanism in GA allows for combining beneficial traits from different parents, leading to superior offspring and higher accuracy. This GA mechanism is particularly useful in the wine classification problem, where the combination of various features can lead to more robust classifiers.





I explore the influence of problem size on accuracy. For Randomized Hill Climbing (RHC), we observe that as the problem size increases, the accuracy initially improves and then plateaus and slightly decreases. This indicates that RHC struggles to maintain high accuracy as the problem complexity grows, likely due to its local search nature and inability to escape local optima effectively. Simulated Annealing (SA) shows a more stable performance across different problem sizes, maintaining an accuracy close to 0.96. This stability suggests that SA's ability to accept worse solutions initially and then focus on improving them helps it manage the increased complexity of larger problem sizes. The controlled exploration and exploitation balance provided by the annealing schedule enables SA to perform consistently well even as the problem size increases. The Genetic Algorithm (GA), however, consistently achieves the highest accuracy, close to 1.0, across all problem sizes. This robustness highlights GA's strength in handling complex classification problems. The population-based approach and the combination of crossover and mutation operations allow GA to explore a wide search space effectively and combine beneficial traits, leading to superior solutions regardless of the problem size.

As for the influence of problem size on training size. For RHC, the training time remains relatively constant and low across all problem sizes, which indicates its efficiency. However, this efficiency comes at the cost of lower accuracy, as previously discussed. SA shows a slightly higher training time compared to RHC but remains efficient overall. The training time for SA also remains relatively stable as the problem size increases, which is likely due to its single-solution approach. While SA takes more time to fine-tune the solution compared to RHC, its ability to balance exploration and exploitation allows it to achieve better accuracy without a substantial increase in training time. By contrasts, GA exhibits the highest training time across all problem sizes. It may be due to the computationally intensive operations involved in maintaining and evolving a population of solutions through crossover and mutation, which is very useful in neural network. The increased training time reflects the additional complexity of managing a population-based approach.

3 Conclusion

The Four Peaks problem and the wine classification problem serve as excellent benchmarks for me to evaluate the performance of various random optimization methods—Randomized Hill Climbing (RHC), Simulated Annealing (SA), and Genetic Algorithm (GA). Each method exhibits unique strengths and weaknesses influenced by its inherent strategy for navigating complex search spaces.

In the context of the Four Peaks problem, RHC quickly finds local optima but struggles to improve further due to its local search focus. SA, on the other hand, shows a stronger performance due to its ability to accept worse solutions initially, which helps it escape local optima and explore the search space more effectively. The annealing schedule that balances exploration and exploitation is key to SA's success. GA performs well by leveraging a population-based approach, using crossover and mutation to explore a wide search space and combine beneficial traits from different solutions. However, GA can be challenged by the disruption of necessary sequences and may suffer from premature convergence without adequate population diversity.

For the wine classification problem, RHC demonstrates quick convergence to good solutions but struggles with higher complexity due to its local search nature. SA balances exploration and exploitation effectively, performing consistently well across different problem sizes. GA outperforms the other methods by achieving the highest accuracy, thanks to its population-based approach that allows for the combination of diverse features to create robust classifiers, which I think it is very similar to backpropagation to some extent. The computational cost of GA is higher due to its intensive operations, but the significant improvement in performance justifies this investment.

Overall, GA's population-based approach consistently shows the highest performance in terms of fitness and accuracy, making it suitable for complex problems requiring robust solutions. SA provides a good balance and performs well with proper parameter tuning, while RHC, despite its efficiency, struggles with more complex landscapes due to its local search limitations. It reminds of the importance of tailored optimization strategies in the specific machine learning and optimization tasks.

For my further study, I will explore hybrid approaches that combine the strengths of these methods, to see if they could yield even better results. For instance, integrating RHC's efficiency with SA's exploration capability or combining GA's diversity maintenance with SA's annealing schedule might offer a powerful optimization framework.