CS7641 Project 2 Report
Randomized Optimization

Juejing Han
jhan446@gatech.edu

***Abstract*—This study reveals distinctive algorithmic strength with trade-offs: Genetic Algorithm (GA) shines in solving the Travelling Salesman Problem, Mutual Information Maximizing Input Clustering (MIMIC) excels in the Knapsack problem, Simulated Annealing (SA) stands out in the Flipflop problem, and Randomized Hill Climbing (RHC) delivers the best performance in the NN experiment. Additionally, key algorithmic characteristics are identified: MIMIC demonstrates the quickest convergence with the longest runtime, GA consistently performs well, and SA and RHC require fewer function evaluations at lower computational costs.

1 EXPERIMENT DESIGN

1.1 Optimization Problems

Three combinatorial optimization problems are used to highlight the variations among optimization algorithms. The Travelling Salesman Problem (TSP) and Knapsack Problem are analogies for real-world decision-making processes, such as efficient route arrangement and resource allocation with specific constraints. Flipflop is an artificial problem, designed as a simple testing ground for assessing optimization algorithms.

1.1.1 *Travelling Salesman Problem (TSP)*

The Travelling Salesman Problem (TSP) is an NP-hard (non-deterministic polynomial-time hard) problem in combinatorial optimizations[1], finding the shortest route among a set of cities, with the constraint of visiting each location once and ultimately returning to the initial city. TSP is a typical minimization problem. However, in this report, all problems are framed as maximization problems. To achieve this, a customized fitness function is deployed to transform TSP into a maximization problem by maximizing the negative value of the overall distance.

1.1.2 *Knapsack*

Knapsack, a typical resource allocation problem, determines which item to include in the collection to achieve the maximum value while ensuring the combined weight of the selected items does not exceed the capacity limit[2].

[1] https://en.wikipedia.org/wiki/Travelling_salesman_problem

[2] https://en.wikipedia.org/wiki/Knapsack_problem

1.1.3 *Flipflop*

Flipflop is a combinatorial optimization problem finding the maximum number of alternating 0s and 1s in a binary string by flipping bits[3]. The objective of the Flipflop problem is to transform a sequence of binary elements (usually represented as 0s and 1s) by flipping adjacent elements to achieve a desired pattern, such as alternating 0s and 1s.

1.2 Algorithms

Four machine learning optimization algorithms – Randomized Hill Climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA), and Mutual Information Maximizing Input Clustering (MIMIC) – are evaluated with Python libraries (mlrose_hiive, etc.).

1.3 Experiments & General Settings

There are two major experiments in this study. The first experiment applies RHC, SA, GA, and MIMIC to solve TSP, Knapsack, and Flipflop, respectively. Each problem is examined across three sample sizes: small, medium, and large (Table 1). Plots of large size are presented in this report. However, plots for both small and medium sizes are also generated by the corresponding code.

Table 1—Sample size for each problem.

Problem	**Small Sample Size**	**Medium Sample Size**	**Large Sample Size**
TSP	15	30	50
Knapsack	15	30	50
Flipflop	30	60	90

The best RHC restart results against iterations are displayed in comparisons to other algorithms. To account for randomness, five random seeds are utilized in the experiment to present the average results, and the standard deviations are shown in Table 2.

Table 2—Standard deviation for each algorithm with different problems.

Problem	**RHC**	**SA**	**GA**	**MIMIC**
TSP	28.0	24.2	20.7	7.8
Knapsack	26.2	40.3	3.01	2.7
Flipflop	2.5	2.3	1.5	1.7

[3] https://medium.com/@duoduoyunnini/introduction-implementation-and-comparison-of-four-randomized-optimization-algorithms-fc4d96f9feea

The second experiment applies Backpropagation (BP), RHC, SA, and GA to the Neural Network (NN) on a binary classification problem. A 5-fold cross-validation approach is applied throughout algorithms. Multiple random states are tested to counteract randomness, and the result with the highest accuracy of each algorithm is presented. For this experiment, one of the datasets (records of diabetes, preprocessed) from Project 1 (Han, 2023) is reused. The class-balanced data has 8 features and 1270 samples, and accuracy is introduced as the metric[4].

2 RESULTS & ANALYSIS

2.1 Experiment 1 – Three Problems

2.1.1 *Travelling Salesman Problem (TSP)*

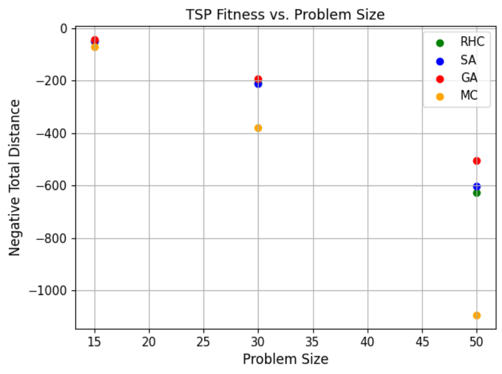


Figure 1—Average Fitness Score vs. Problem Size

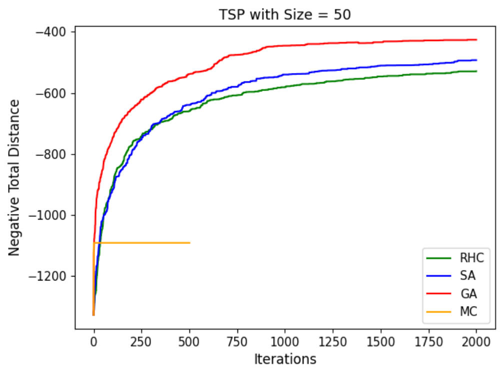


Figure 2—Fitness Score vs. Iteration

Fig. 1 illustrates that for a small problem size with 15 cities, algorithms share similar fitness scores, that is because when problem size is small, it is easy for algorithms to find the best solution. As the size increases, MIMIC falls behind, and **GA stands out by consistently achieving the best score** (Fig.1). GA is adept at navigating complex search spaces, its parallel exploration allows the ability to avoid being trapped in local optima[5]. GA (population size = 80) score peaks at 1400 iterations and levels off thereafter (Fig. 2). RHC (with zero restarts) and SA exhibit similar performance and tend to converge after 1800 iterations. Among all algorithms, MIMIC (population size = 80) converges to the lowest score but has the fewest iterations (500) and the fastest convergence. Besides, MIMIC also achieves the lowest standard deviation (Table 2). Excluding MIMIC, GA obtains the highest score with the lowest variance, while SA outperforms RHC with a higher score and lower variance.

In Fig. 3, GA and MIMIC involve more function evaluations (FEvals, their FEvals overlap because of the same population size). RHC with zero restarts shares the same number of FEvals with SA. MIMIC is the most computationally expensive, and GA requires a longer runtime than RHC and SA (Fig. 4).

[4] https://www.statology.org/f1-score-vs-accuracy/

[5] https://stats.stackexchange.com/questions/23106/benefits-of-using-genetic-algorithm

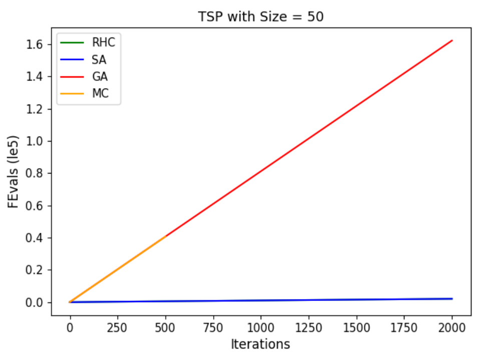


Figure 3—FEvals vs. Iteration

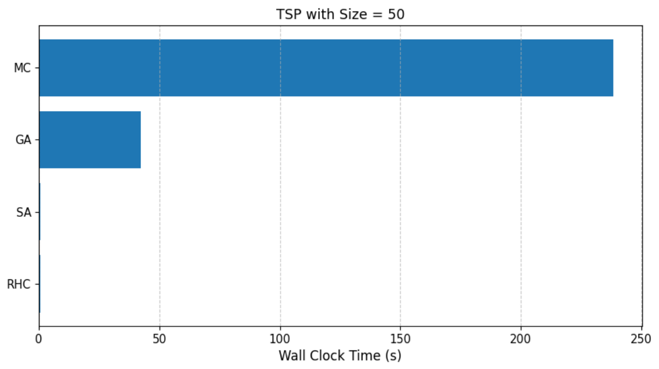


Figure 4—Wall Clock Time

In this specific case, optimizing GA's population size to 80 yields the best result. GA's population size represents the number of candidate solutions, while a mutation rate of 0.05 defines the likelihood of a mutation operation. With these parameters, GA successfully represents the TSP problem and achieves the highest fitness score. The use of a large population size in GA ensures diversity and thorough exploration of the solution space. However, it also demands more memory and computational resources. Additionally, its process of evaluating the fitness of each candidate in the population contributes to the high number of function evaluations and the total runtime.

2.1.2 *Knapsack*

Across various problem sizes, GA and MIMIC outperform other algorithms, while RHC consistently falls behind (Fig. 5). As the problem size grows, MIMIC achieves a slightly higher score than GA, and SA's performance deteriorates.

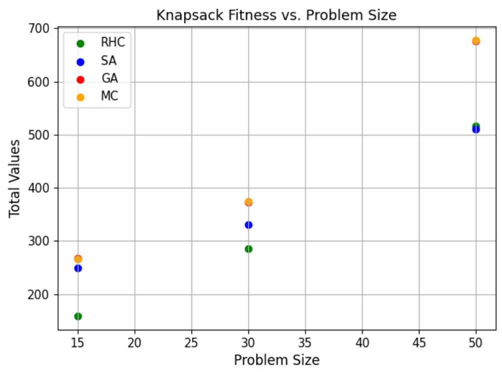


Figure 5—Average Fitness Score vs. Problem Size

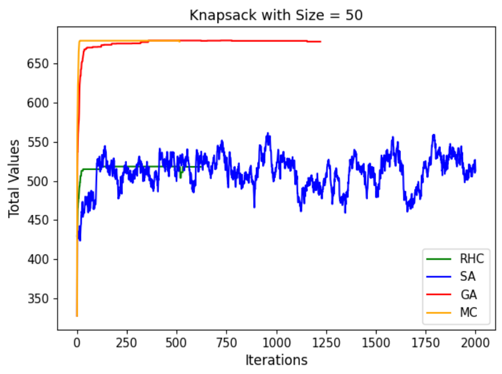


Figure 6—Fitness Score vs. Iteration

In Fig. 6, **MIMIC stands out with the fewest iterations and the fastest to the highest fitness score.** Although GA and MIMIC (population size is 60 and 220 respectively) eventually stabilize at the same highest score, GA converges after 300 iterations.

MIMIC shows the lowest standard deviation (Table 2), indicating consistent performance, while SA achieves the highest variance, signifying greater variability. During SA tuning, lower temperatures result in smoother fitness curves, while higher temperatures lead to fluctuations (Fig. 6). This is because higher temperatures increase the probability of accepting worse solutions[6]. SA with higher temperatures outperforms RHC (with 8 restarts); however, it also leads to higher uncertainty. Besides, RHC's randomness and lack of systematic exploration might be the reason for its poor performance.

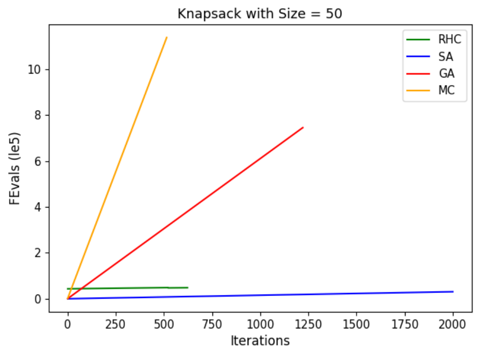


Figure 7—FEvals vs. Iteration

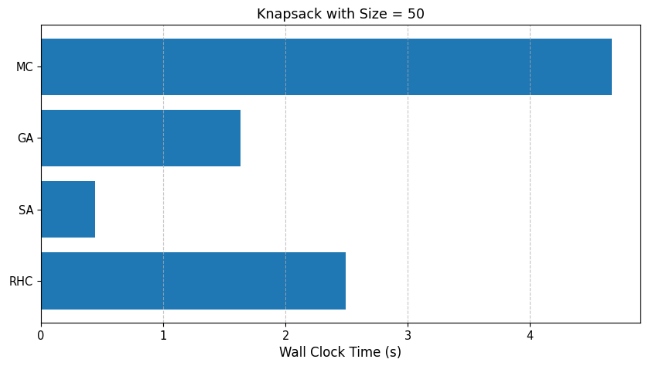


Figure 8—Wall Clock Time

MIMIC converges first but requires the highest number of FEvals (Fig. 7). Since RHC involves 8 restarts, it demands more FEvals than SA. Additionally, MIMIC has the highest computational cost, while SA achieves the shortest runtime (Fig. 8) – SA's stochastic nature enhances its effectiveness.

MIMIC converges faster and more reliably because it effectively shares information about the cost function throughout the optimization process and attempts to discover common underlying patterns about optimal solutions (Bonet, 1996). In this specific case, increasing MIMIC's population size to 220 yields the best result, indicating the number of candidate solutions significantly influences its performance. A large population size allows for better exploration of the solution space but requires more computational cost. GA and MIMIC evaluate the fitness of each candidate in the population during each iteration. Since GA's population size is smaller, MIMIC demands more FEvals than GA.

2.1.3 *Flipflop*

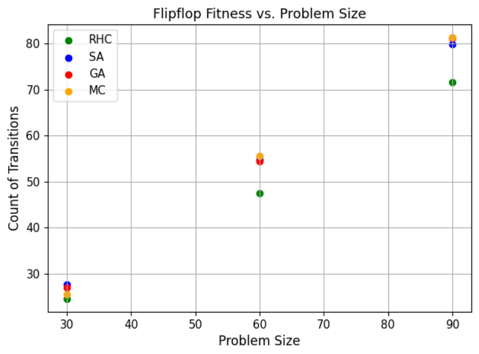


Figure 9—Average Fitness Score vs. Problem Size

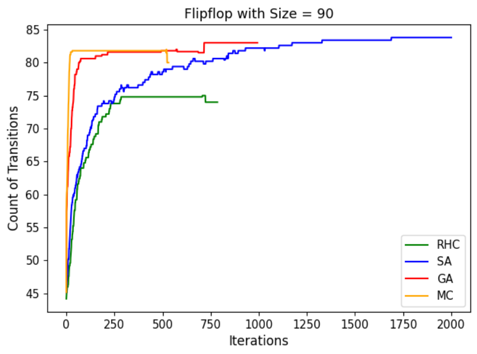


Figure 10—Fitness Score vs. Iteration

[6] http://webpages.iust.ac.ir/yaghini/Courses/AOR_881/Simulated_Annealing_01.pdf

With increasing problem sizes, SA, GA, and MIMIC maintain similar fitness scores, while RHC gradually falls behind (Fig. 9). GA and MIMIC (population size is 100 and 250 respectively) converge rapidly – MIMIC plateaus within 30 iterations, and GA levels off within 200 iterations (Fig. 10). **SA converges more slowly but eventually reaches a higher fitness score than others**. Meanwhile, all algorithms exhibit small variances (Table 2).

MIMIC converges in the fewest iterations but requires the highest number of FEvals, while GA achieves a higher score with fewer function evaluations (Fig. 11). SA requires the fewest number of FEvals, while 1-restart RHC requires slightly more. Moreover, MIMIC demands the highest computational cost, followed by GA, while SA and RHC cost much less than GA and MIMIC (Fig. 12).

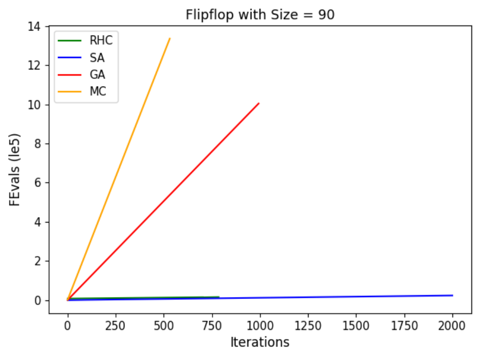


Figure 11—FEvals vs. Iteration

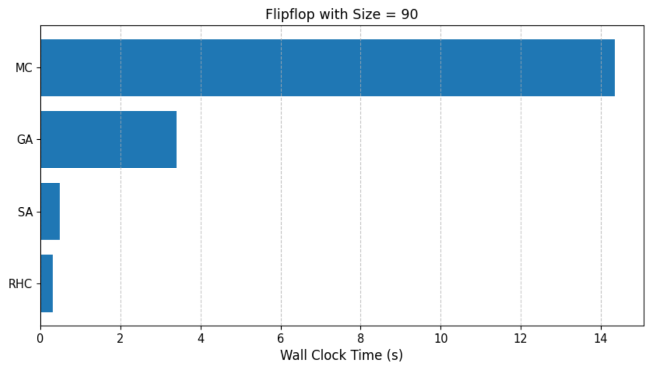


Figure 12—Wall Clock Time

SA can be effective in finding global optima, even after encountering a local optimum, as it accepts solutions that are worse than the best candidate[7]. However, its success depends on the choice of the cooling schedule and the initial temperature. As discussed in Section 2.1.2, different temperatures can introduce performance variability, and the preference represents a trade-off between exploration and exploitation. In this specific case, SA is configured with an Arithmetic Decay cooling schedule and an initial temperature of 20, which proves to be the optimal parameter selection for solving the Flipflop problem. Besides, as a stochastic optimization algorithm, SA makes use of randomness as part of its search process[8], and this significantly reduces the number of function evaluations and the runtime.

2.2 Experiment 2 – Neural Networks

Algorithms share the same general settings: 20 hidden nodes, ReLU activation, 2000 max iterations, 200 max attempts, and turned-on early stopping. Grid Search is employed for the optimal parameters: BP learning rate = 0.0001; RHC learning rate = 0.8 with N_restart = 6; SA learning rate = 1 with a Geometric Decay schedule; GA learning rate = 0.00001 with population size = 40.

[7] https://www.sciencedirect.com/topics/materials-science/simulated-annealing

[8] https://machinelearningmastery.com/simulated-annealing-from-scratch-in-python

2.2.1 *Backpropagation (BP)*

In general, adding samples mitigates overfitting and gradually enhances both scores (Fig. 13), which converge together and peak at 1000 instances. Additional samples may help to improve its performance. Smaller learning rates yield better results – it allows the algorithm to make smaller adjustments to the weights during each iteration and hence prevents overshooting the optimal weight values. Moreover, the training loss consistently decreases as the number of epochs increases, reflecting effective model learning (Fig. 14). The loss curve tends to plateau after 500 iterations.

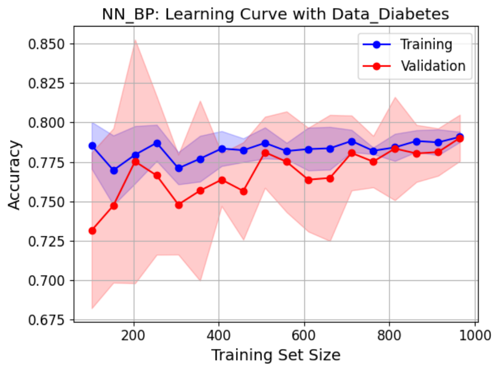


Figure 13—Learning Curve

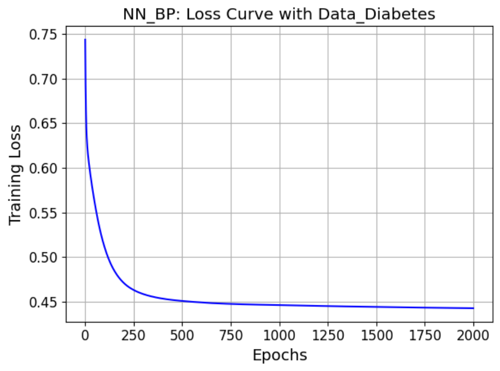


Figure 14—Loss Curve

2.2.2 *Randomized Hill Climbing (RHC)*

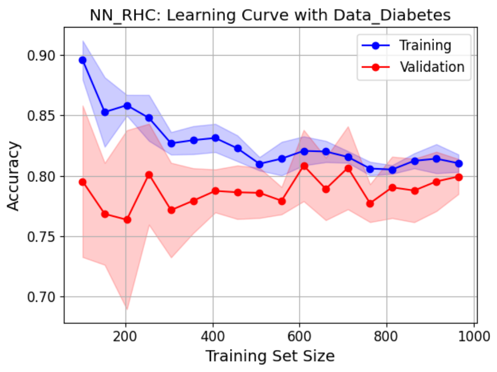


Figure 15—Learning Curve

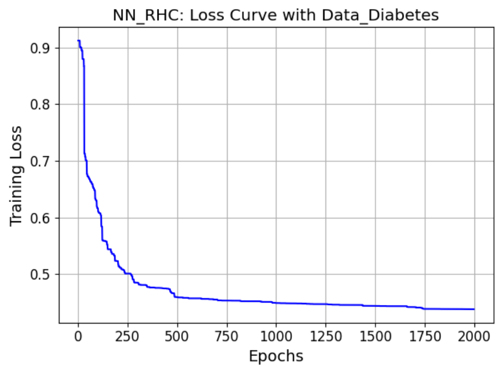


Figure 16—Loss Curve

A larger training size generally mitigates overfitting (Fig. 15). Training and validation scores converge and peak at around 600 samples, with no performance gain beyond this point. Increasing the number of restarts improves its performance. 6-restart RHC takes 500 epochs to gradually plateau (Fig. 16).

2.2.3 *Simulated Annealing (SA)*

Increasing the training size does not significantly enhance the validation score when the sample size is below 450 (Fig. 17). Beyond this point, validation performance gradually improves, reaching its peak at around 800 instances. Further adding samples does not yield additional improvements.

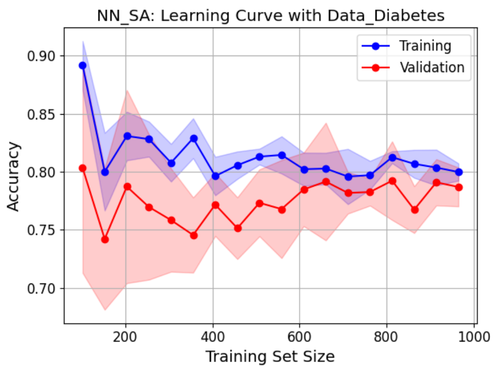


Figure 17—Learning Curve

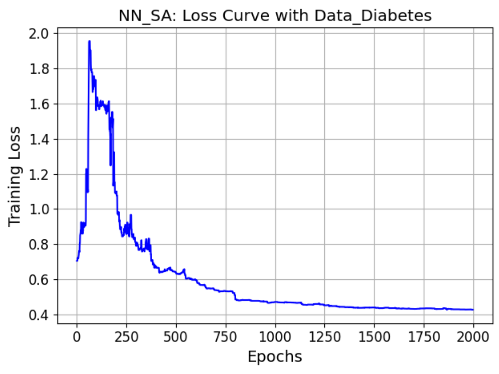


Figure 18—Loss Curve

It is noteworthy that both scores start at a high level in the initial stages of the learning curve. This may be due to the relatively small training sample size, consisting of approximately 100 instances in this case. When working with a small sample, the model trained on this subset may effectively capture the characteristics and perform well on the validation set. Different cooling schedules have notable impacts on SA performance, and the Geometric Decay schedule yields the best result.

The loss curve tends to plateau after 750 iterations (Fig. 18). During the initial 300 epochs, it displays greater volatility in comparison to BP and RHC. This may be attributed to the higher learning rate (set to 1), resulting in larger weight updates. Additionally, the initial loss value is small – it could be due to the random initialization of weights being close to an optimal solution. However, the loss curve exhibits a general decreasing trend, which suggests that the model is effectively learning.

2.2.4 *Genetic Algorithm (GA)*

Increasing the training size can help reduce overfitting, although model performance does not show a consistent improvement and exhibits fluctuations (Fig. 19).

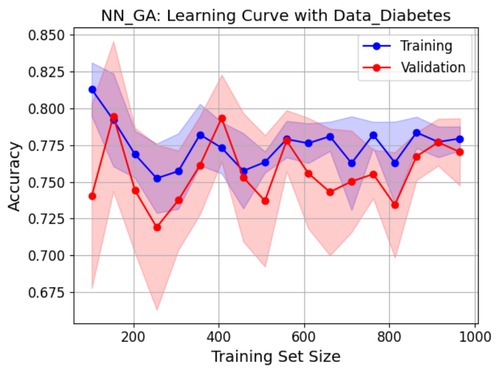


Figure 19—Learning Curve

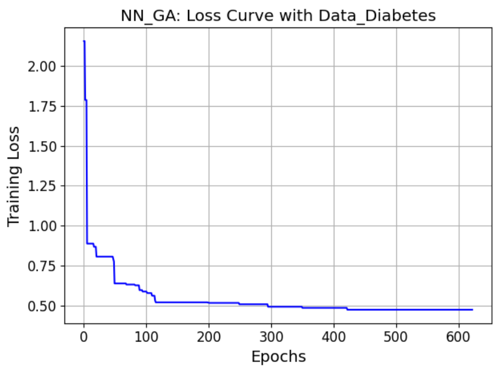


Figure 20—Loss Curve

This could be attributed to population diversity – the initial population may contain a mix of good and poor solutions, leading to performance variance. Interestingly, at a training size of 400, the validation score surpasses the training score. It can be caused by random mutations and crossovers, which may have resulted in a solution that performs better on the validation set. Tuning different population sizes results in various performances, with 40 being the optimal choice. GA requires the fewest epochs for its loss curve to converge (110 epochs) and plateaus afterward (Fig. 20). In the Knapsack and Flipflop problems, GA also exhibits rapid convergence. Its ability to maintain a diverse population and genetic operators may contribute to identifying promising solutions efficiently.

2.2.5 NN Comparison

Fig. 21 shows that in the neural networks experiment, RHC achieves the highest accuracy at 0.819, followed by SA at 0.807, GA at 0.803, and BP at 0.795. BP is gradient-based and can get stuck in local optima, and it is highly sensitive to the initial weights. In contrast, an evolutionary algorithm like GA is more flexible in finding a global optimum and often explores a broader range of initial solutions. Besides, stochastic algorithms like SA and RHC also have more ability to escape local optima. RHC with multiple restarts increases its chances of escaping local minima and finding a favorable solution. Furthermore, the typical problem and dataset may align well with the strength of RHC.

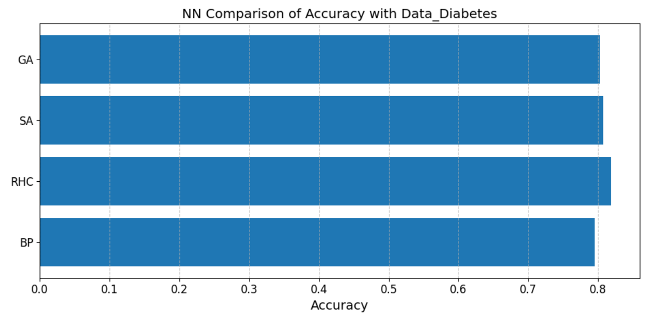
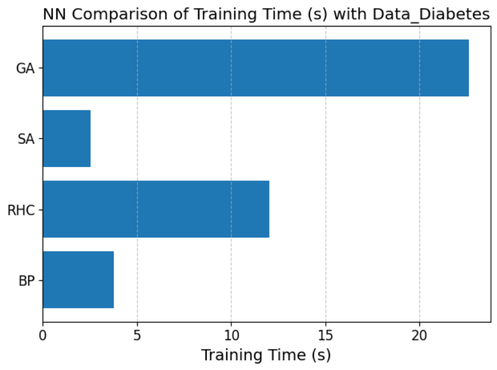


Figure 21—Comparison of Accuracy



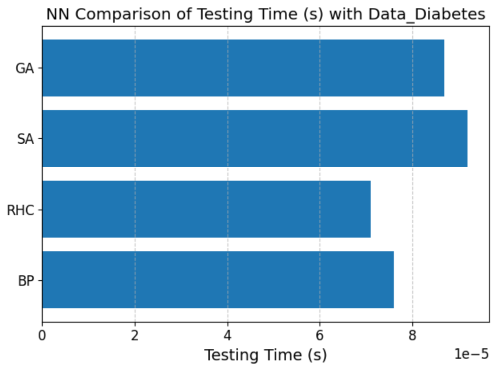


Figure 22—Comparison of Training Time

Figure 23— Comparison of Testing Time

GA exhibits the lengthiest training time, followed by RHC, BP, and SA (Fig. 22). This is due to: GA's characteristics discussed in Section 2.1.1 (its population size and fitness evaluation), RHC with 6 restarts requires more time for training since the algorithm needs to restart from different initial states, and SA benefits from its stochastic and probabilistic characteristics. Conversely, although SA achieves the shortest training time, it demands the longest testing time among all algorithms (Fig. 23). RHC and BP demonstrate shorter testing times, signaling their efficient implementations for testing.

3 CONCLUSIONS

In the Travelling Salesman Problem (TSP), the Genetic Algorithm (GA) stands out as the top performer, achieving the best fitness score with minimal variance. However, it comes at the cost of longer computation time compared to Randomized Hill Climbing (RHC) and Simulated Annealing (SA). The utilization of a large population size in GA ensures diversity and comprehensive exploration of its solution space. Nevertheless, this leads to increased memory and computational requirements. On the other hand, MIMIC achieves the lowest score. MIMIC highly relies on the quality of the probabilistic model it builds, if the model fails to capture the underlying structure of TSP, it tends to underperform.

In the Knapsack problem, MIMIC distinguishes itself with the fewest iterations, the fastest convergence to the highest fitness score, and the lowest variance. On the other hand, it also demands the highest computational cost. MIMIC effectively shares information about the cost function across different iterations and leverages underlying patterns to guide its search. It enables MIMIC to converge fast and produce stable results at a high computational cost. Population-based algorithms like MIMIC and GA may naturally align with Knapsack structure and hence outperform RHC and SA.

In the Flipflop problem, SA stands out by converging to the highest fitness score with the fewest FEvals at almost the lowest computational cost. SA can effectively discover global optima even after finding a local optimum. Its stochastic and probabilistic nature can often lead to fewer computational resources. Due to the simpler structure of the Flipflop problem compared to TSP and Knapsack, the algorithms achieve similar fitness scores with RHC slightly underperforming.

In the Neural Networks experiment, RHC with 6 restarts excels with the highest accuracy and the shortest testing time. However, RHC with multiple restarts demands a longer training time because the algorithm restarts from different initial states to find global optima. In contrast, as a gradient-based algorithm, BP has a higher likelihood of getting stuck in a local optimum. Besides, GA's diverse set of candidate solutions and genetic operators make it efficient in finding optima and hence converge fast.

Fine-tuning parameters and adding more samples impact results. Further parameter experiments and additional samples could improve algorithm performance.

4 REFERENCES

1. J. Han (2023). CS7641 Project 1 Report Supervised Learning. Class Assignment Report of Machine Learning, Georgie Institute of Technology. Georgie, United States of America.
2. J. D. Bonet, C. Isbell, P. Viola (1996). MIMIC: Finding Optima by Estimating Probability Densities. Advances in Neural Information Processing Systems 9 (NIPS 1996).