CS 7641 Machine Learning: Randomized Optimization

Jilong Cui
jcui@gatech.edu

1 OVERVIEW

This report is bifurcated into two primary sections.

1.1 Randomized Optimization Algorithms

In the first part of the report, three distinct randomized optimization problems are formulated. To tackle these optimization challenges, four different randomized operation algorithms were employed: a. Random Hill Climbing (RHC), b. Simulated Annealing (SA), c. Genetic Algorithm (GA), d. Mutual Information Maximizing Input Clustering (MIMIC).

Each of these algorithms has the objective to enhance the fitness of the optimization problem. The performance outcomes of the algorithms are compared and evaluated against each other to derive insights into their efficiency and applicability.

1.2 Neural Network Optimization using Randomized Algorithms

The second part of the report focuses on a practical application: using the first three randomized algorithms (excluding MIMIC) to determine suitable weights for a neural network. In this context, the algorithms serve as an alternative to the conventional backpropagation method. The neural network in question is the same as the one utilized in assignment #1, where the Wine dataset was the focal point. The Wine dataset comprises 14 distinct chemical and physical attributes of wines, aimed at determining a wine's class from three possible categories. The Wine dataset's focus on consistent chemical properties offers a straightforward classification challenge. This dataset is especially reliable due to the predictability of these properties in wines.

2 RANDOMIZED OPTIMIZATION PROBLEM

2.1 FlipFlop Optimization

The FlipFlop problem revolves around a binary string, aiming to maximize alternating bits. In simpler terms, it rewards consecutive differing bits (e.g., 1010 scores 3 while 1110 scores 1). While seemingly elementary, this problem presents a unique challenge for random optimization algorithms due to the localized dependency of each bit on its immediate neighbors. Through this problem, the nuances in the behavior of different algorithms under such constraints become evident.

2.2 N-Queen Optimization

The N-Queens challenge involves placing N queens on an NxN chessboard so that no two queens threaten each other. Classified as an NP-Complete problem, it's characterized by a non-convex optimization surface where a minor change in queen placement can drastically impact the overall solution. This problem is pivotal for comparing randomized optimization algorithms, highlighting their potential weaknesses. For a 64x64 chessboard, the objective is to achieve an optimal arrangement where no queens threaten each other, aiming for a fitness value representing 2016 non-attacking queen pairs (the maximum possible for this board size).

2.3 Knapsack Problem

A classic in the realm of optimization, the Knapsack problem entails selecting a group of items, each with a specific weight and value, to pack into a bag. The objective is to maximize the total value without exceeding the bag's weight capacity. As an NP-Complete challenge, the Knapsack problem features multiple local optima, serving as an excellent benchmark to assess the effectiveness of various optimization algorithms. Given its NP-hard nature, exact optimal solutions may be elusive for extensive item lists, but optimization algorithms strive to approximate the best solution in a feasible timeframe.

3 RANDOMIZED OPTIMIZATION ANALYSIS

The analysis was conducted by utilizing the **mlrose_hiive** Python library (RHCRunner, SARunner, GARunner, MIMICRunner). For every optimization algorithm, multiple parameter sets were experimented with. After evaluation, the best-performing parameter set was selected to derive the fitness performance metrics. Leveraging this optimal parameter set, a characteristic 'Iteration vs. Value' graph was plotted for each algorithm to visualize their performance trends. Figure 1 showcases a snapshot of the parametric analysis results for each algorithm. While the detailed parametric results are not included in this report because of the limited space, the performance trajectory based on the evaluated parameter sets is discussed. For the detailed analysis results, kindly consult the submitted notebook.

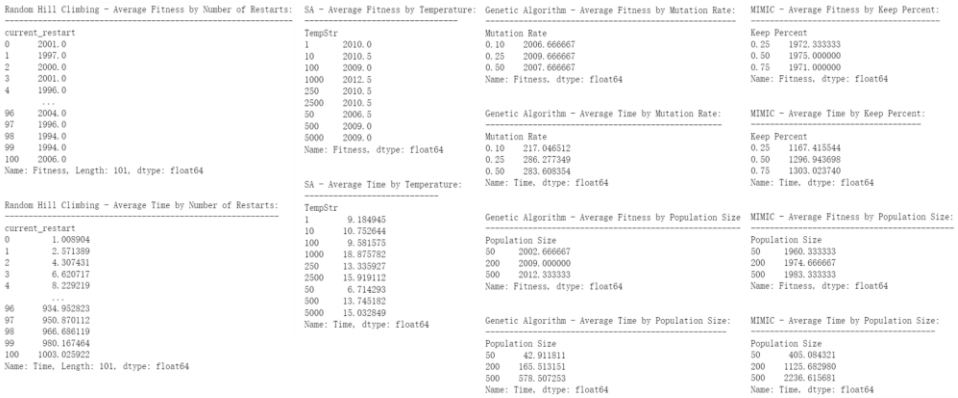


Figure 1. Sample Parameter Tunning Result

3.1 Parametric Analysis on Algorithms

In optimizing various algorithms, different parameters were tuned.

For ***Random Hill Climbing (RHC)***, "re-start" values were considered, determining how often the algorithm should start afresh to evade local optima. When adjusting these restart values in RHC, there was minimal change in fitness outcomes. Yet, as restarts increased, runtimes prolonged, suggesting that while more restarts can potentially assist in bypassing local optima, they come at a computational cost.

Simulated Annealing (SA) utilized "temperature," affecting the likelihood of accepting worse solutions to ensure broad exploration. In the context of SA, altering the temperature resulted in diverse fitness outcomes. The most optimal fitness was observed at a mid-range temperature, highlighting a balance between exploration and exploitation, heavily influenced by the decay mechanism.

The ***Genetic Algorithm (GA)*** leaned on two main parameters: "population size," which dictates the number of potential solutions, and "mutation rates," influencing genetic diversity. Within GA's framework, modifying these parameters saw a correlation between larger population sizes and higher mutation rates with enhanced fitness outcomes. However, these improvements in fitness also led to longer computational times, indicating a trade-off between the benefits of increased genetic diversity and broader search spaces against processing time.

Lastly, ***MIMIC*** adjusted both "population size" and the "keep percent." The latter determines the fraction of top-performing samples retained for generating subsequent samples. For MIMIC, as the population size and 'keep percent' were varied, there was a noticeable improvement in fitness with larger populations. This suggests MIMIC's methodology becomes more effective with larger sample sizes, which present a wider array of potential solutions. Yet, this effectiveness also resulted in increased computational demands, particularly as the size of the population grew.

Despite the specific parameter values being tailored for different randomized problems, a consistent trend emerged across all algorithms. This observation underscores the intrinsic characteristics of these algorithms and their responses to parameter adjustments.

3.2 FlipFlop Problem Analysis

The FlipFlop problem, analyzed using the mlrose_hiive.FlipFlopOpt function, was set with a string length of 500. The theoretical maximum for such a configuration stands at 499, reflecting the total number of bit alternations in the string. Table 1 offers a succinct view of the performance metrics, capturing insights on average fitness, maximum fitness, iteration time, and the typical convergence iteration. Complementing this, Figure 2 paints a visual narrative of the iteration-to-value trajectory for the examined algorithms.

Upon examining the provided metrics, distinct performance patterns emerge for each of the four algorithms applied to the FlipFlop optimization:

Random Hill Climbing (RHC) exhibits moderate results with an average fitness of 390 and a maximum fitness of 414. However, it presents a significant iteration time of 1026 s. On the brighter side, RHC shows a swift convergence, needing only 900 iterations.

Simulated Annealing (SA) excels in the fitness department, delivering an impressive average fitness of 484 and a near-perfect maximum fitness of 488. This prowess, however, comes at the expense of a longer convergence journey, requiring as many as 12,500 iterations. Despite this, its average time of iteration stands at a moderate 239 s.

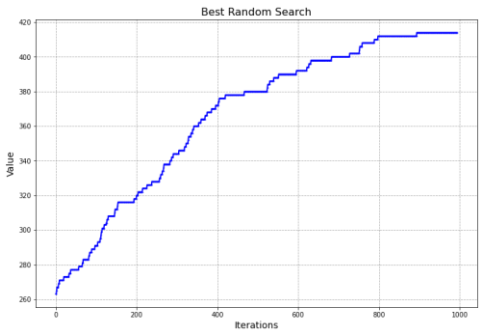
MIMIC's performance reveals certain challenges, registering the lowest average and maximum fitness scores of 323 and 363, respectively. While its average iteration time of 81 s places it between RHC and GA, its convergence at 980 iterations is commendable, only slightly surpassing RHC.

The Genetic Algorithm (GA), however, showcases a balanced performance. It achieves a notable average fitness of 411 and a maximum fitness of 442. Remarkably, GA excels with an exceptionally low iteration time of 3.64 s, emphasizing its efficiency. Though its convergence requires 5,250 iterations, which is higher than RHC but considerably less than SA, it offers a harmonious blend of fitness outcome and efficiency.

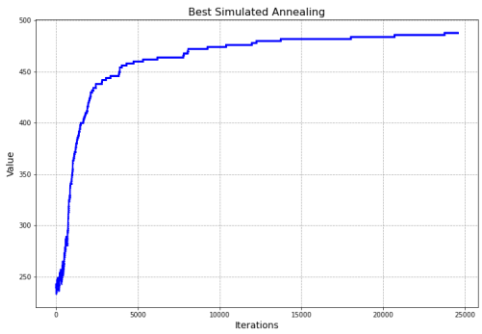
In synthesizing the data, the Genetic Algorithm (GA) emerges as the frontrunner. Its commendable fitness metrics, combined with an unparalleled iteration efficiency and a reasonable convergence rate, make it an optimal choice for the FlipFlop problem. While each algorithm has its merits, GA's balanced performance shines, suggesting a robust capability to navigate binary optimization challenges such as the FlipFlop problem.

Table 1. FlipFlop Overall Summary

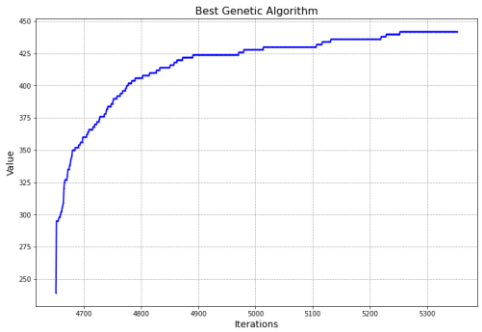
FlipFlop	**RHC**	**SA**	**GA**	**MIMIC**
Avg. Fitness	390.43	484.6	411	322.89
Max. Fitness	414	488	442	363
Avg. Time of Iteration	1026.61 s	239.29 s	3.64 s	81.34 s
Typ. Converge Iteration	900	12500	5250	980



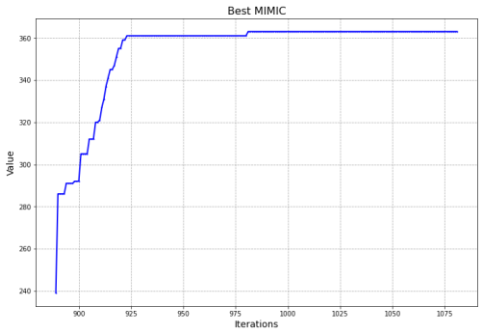
(a) FlipFlop – Random Hill Climb



(b) FlipFlop – Simulated Annealing



(c) FlipFlop – Genetic Algorithm



(d) FlipFlop – MIMIC

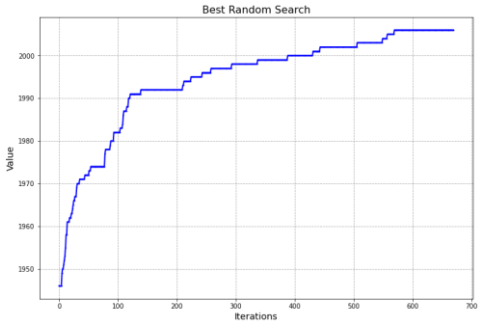
Figure 2. FlipFlop Best Fitness vs. Iteration Curve

3.3 N-Queen Problem Analysis

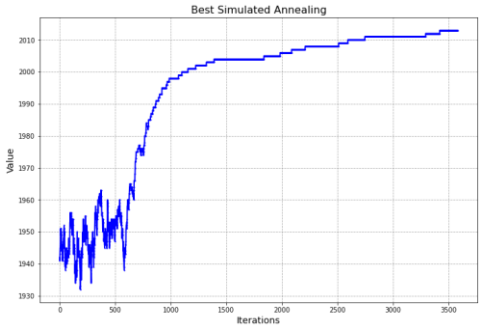
The N-Queen problem, a classic combinatorial challenge, has been set up using a custom function and is defined by the mlrose_hiive.CustomFitness module. In this configuration, a 64x64 chessboard is used, aiming to optimize the placement of queens such that the number of un-attacked pairs is maximized. The optimal solution would have a total of 2016 un-attacked pairs. Table 2 offers a succinct view of the performance metrics, capturing insights on average fitness, maximum fitness, iteration time, and the typical convergence iteration. Complementing this, Figure 3 paints a visual narrative of the iteration-to-value trajectory for the examined algorithms.

Table 2. N-Queen Overall Summary

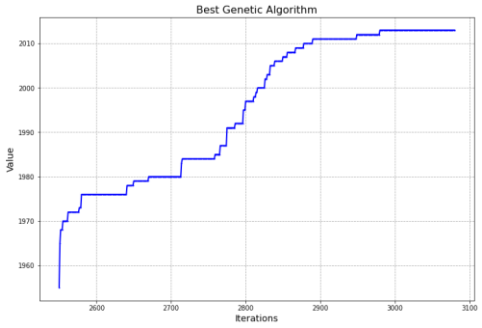
N-Queens	**RHC**	**SA**	**GA**	**MIMIC**
Avg. Fitness	1998.63	2009.72	2008	1972.78
Max. Fitness	2006	2013	2013	1986
Avg. Time of Iteration	374.75 s	12.57 s	262.31 s	1255.79 s
Typ. Converge Iteration	580	3500	2980	790



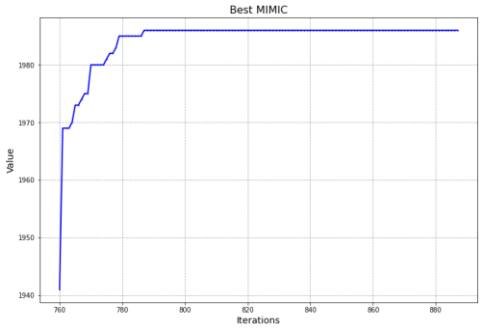
(a) N-Queen – Random Hill Climb



(b) N-Queen – Simulated Annealing



(c) N-Queen – Genetic Algorithm



(d) N-Queen – MIMIC

Figure 3. N-Queen Best Fitness vs. Iteration Curve

Random Hill Climbing (RHC) registers a commendable performance with an average fitness score of 1998 and peaks at a maximum fitness of 2006. However, it demands a relatively long iteration time, clocking in at 375 s. Its redeeming feature is its prompt convergence, wrapping up in just 580 iterations.

Simulated Annealing (SA) truly stands out for the N-Queen problem. With an exceptional average fitness of 2009 and a near-optimal maximum fitness of 2013, it clearly excels in navigating the solution space for this particular challenge. While its convergence needs 3,500 iterations, the swift average iteration time of 13 s ensures efficient progress. The efficacy of SA in this scenario could be attributed to its inherent mechanism. The N-Queen problem, with its combinatorial nature, benefits from SA's ability to escape local optima by occasionally accepting worse solutions. This trait aids in a broader exploration of the solution space, enhancing the chances of finding the most optimized configurations of queens.

The Genetic Algorithm (GA) also delivers strong results. It reports an average fitness of 2008, closely tailing SA, and a maximum fitness mirroring SA's peak at 2013. However, with an iteration time of 262 s, GA demands more time compared to SA, even though its convergence at 2,980 iterations is relatively efficient given the problem's complexity.

MIMIC, in contrast, faces challenges in the N-Queen landscape. It records the lowest average fitness of 1973 and a maximum of 1986. The iteration time for MIMIC is notably extended, taking 1256 s, which is the lengthiest among the four. Its convergence at 790 iterations, though quicker than both SA and GA, is offset by its lower fitness metrics.

Drawing from the results, Simulated Annealing (SA) emerges as the superior choice for the N-Queen problem. Its ability to maintain a balance between exploration and exploitation, coupled with its mechanism to sidestep local optima, aligns well with the demands of the N-Queen puzzle. The problem's inherent need to explore vast configurations to maximize the number of un-attacked pairs is adeptly addressed by SA's randomized approach, making it the most fitting algorithm for this challenge.

3.4 Knapsack Problem Analysis

The Knapsack problem was set using mlrose_hiive.KnapsackOpt with 200 items. Each item's weight ranged from 10 to 40, and its value between 20 and 30. A weight percentage of 0.5 determined the knapsack's capacity, challenging the optimization to maximize value within weight constraints. Table 3 offers a succinct view of the performance metrics, capturing insights on average fitness, maximum fitness, iteration time, and the typical convergence iteration. Complementing this, Figure 4 paints a visual narrative of the iteration-to-value trajectory for the examined algorithms.

Random Hill Climbing (RHC) posted an average fitness of 1478, peaking at a maximum of 2697. While the iteration time is fairly quick at 17 s, RHC's convergence is almost instantaneous, occurring in just 25 iterations.

Simulated Annealing (SA) displayed commendable results with an average fitness of 2612 and a top score nearing the ideal at 2814. Its iteration time is astonishingly brief, a mere 0.22 s. However, convergence demands a slightly extended 900 iterations, contrasting its rapid per-iteration time.

The Genetic Algorithm (GA) also fared well with an impressive average fitness of 3207, almost touching the apex with a max of 3208. Its iteration time is on the higher side at 25 s, but the model converges relatively quickly in 357 iterations.

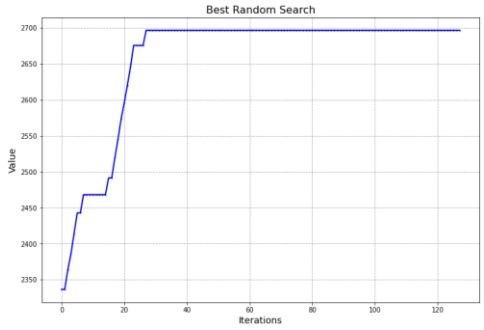
Yet, it's MIMIC that truly stands out. It garners the highest average fitness of 3276 and an unmatched maximum of 3301. While its iteration time of 126 s may seem protracted, the results it delivers more than compensate. Convergence for MIMIC occurs within 780 iterations, which, considering the quality of solutions it finds, is worthwhile.

In reflection, MIMIC emerges as the best-suited algorithm for the Knapsack problem. Its exceptional fitness scores underscore its capability to effectively balance weight and value constraints, maximizing the knapsack's value. One potential reason for MIMIC's superiority is its probabilistic model-building approach. By understanding patterns and dependencies between the items, it can make more informed decisions about

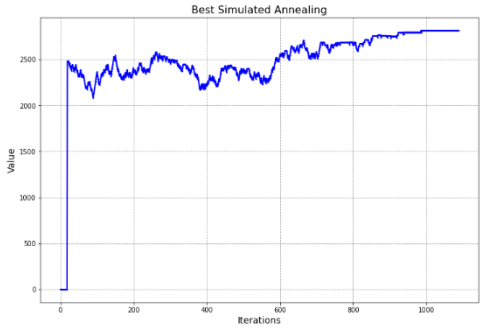
which items to include, optimizing the balance between total weight and value. This holistic approach allows MIMIC to efficiently navigate and pinpoint optimal or near-optimal solutions for complex problems like the Knapsack.

Table 3. Knapsack Overall Summary

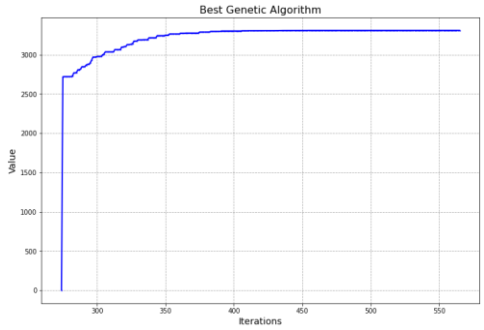
Knapsack	**RHC**	**SA**	**GA**	**MIMIC**
Avg. Fitness	1478.07	2612.19	3207.66	3275.81
Max. Fitness	2696.7	2814.52	3208.36	3301.49
Avg. Time of Iteration	16.89 s	0.22 s	25.1 s	126.85 s
Typ. Converge Iteration	25	900	375	780



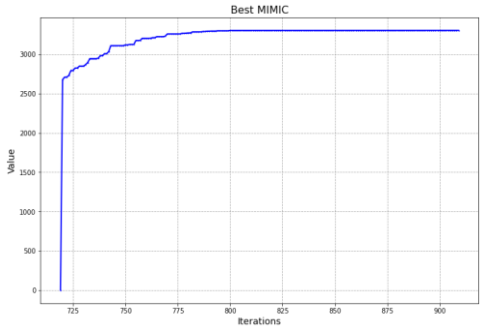
(a) Knapsack – Random Hill Climb



(b) Knapsack – Simulated Annealing



(c) Knapsack – Genetic Algorithm



(d) Knapsack – MIMIC

Figure 4. Knapsack Best Fitness vs. Iteration Curve

4 NEURAL NETWORK OPTIMIZATION ANALYSIS

In the exploration of neural network optimization using RHC, SA, and GA on the Wind Data set, distinct performance trends emerge. The dataset, which classifies into three classes based on 14 features, was tackled using mlrose_hiive.NNGSRunner employing random_hill_climb, sa.simulated_annealing, and

ga.genetic_alg algorithms. While the best Neural Network from Assignment 1 attained an F-1 score of 0.93, this study aimed to determine if these optimization techniques could achieve or surpass this score. For a comprehensive view of these results, Table 4 summarizes the overall performance metrics of each algorithm, Figure 5 provides the F1 scores for each model, the confusion matrix for the NN-SA model, and the iteration versus value curve for the NN RHC.

Table 4. Neural Network Overall Summary

Neural Network	**RHC**	**SA**	**GA**
Avg. Fitness	3.78	2.17	8.57
Max. Fitness	4.01	4.05	14.06
Avg. Time of Iteration	295.7 s	185.1 s	499.3 s
Typ. Converge Iteration	-	6000	175
Weighted F-1 Score (Prediction Accuracy)	0.26	0.91	0.76

4.1 Neural Network – Random Hill Climbing:

Boasting an average fitness of 3.78 and peaking at a maximum of 4.01, RHC took a substantial iteration time, averaging at 296 s. However, its Achilles' heel was its inability to converge (Figure 5(f)). This lack of convergence culminated in a rather dismal F1 score of 0.26. This limitation is an echo of RHC's inherent characteristics. Given its local search nature, RHC is susceptible to becoming ensnared in local optima. Without sophisticated mechanisms to extricate itself from these regions, the algorithm inevitably grapples with optimization challenges. This limitation manifests as the pronounced inability to converge and subsequently low F1 scores, indicating suboptimal predictive accuracy.

4.2 Neural Network – Simulated Annealing:

Clocking an average fitness of 2.17 and a zenith of 4.05, SA's performance showcased its efficiency with an average iteration time of 185 s, managing to converge at the 6000th iteration. Impressively, despite these metrics, SA clinched an F1 score of 0.91. This juxtaposition of lower fitness scores when measured against GA, combined with superior prediction accuracy, is intriguing. The magic of SA lies in its adaptive essence. Unlike rigid algorithms, SA has the finesse to occasionally accept solutions that may initially appear suboptimal. This flexibility ensures it evades the pitfalls of local optima, embracing a more expansive search space. It's plausible that this nuanced exploration endowed SA with enhanced generalization, thus explaining its heightened prediction prowess.

4.3 Neural Network – Genetic Algorithm:

GA was a powerhouse in terms of fitness metrics, registering an average of 8.57 and hitting a high of 14. Though its iteration time averaged at a lengthier 499 s, it exhibited alacrity in convergence, settling down

at the 175th iteration. However, the enigma lies in its F1 score of 0.76, which stands in stark contrast to its high fitness metrics. This incongruity underscores a pivotal lesson in neural network optimization: peak fitness during the training phase isn't synonymous with superior real-world prediction. A probable explanation is overfitting. GA's meticulous fine-tuning to the training data might have robbed it of its flexibility, rendering it less adept at generalizing to novel, unseen data.

4.4 Neural Network Summary

Upon a granular examination of the Wind Data set neural network optimization, Simulated Annealing unquestionably takes the crown, deftly harmonizing between optimization nuances and prediction efficiency. On the flip side, Random Hill Climbing, with its challenges rooted in its intrinsic design, underscores the pitfalls of limited exploration. The Genetic Algorithm, despite its commendable fitness feats, offers a salient reminder: unparalleled training optimization can sometimes be a double-edged sword, not always guaranteeing impeccable predictive accuracy in real-world scenarios.

	precision	recall	f1-score	support
1	0.91358	0.93968	0.92645	315
2	0.93970	0.92804	0.93383	403
3	0.93885	0.92553	0.93214	282
accuracy			0.93100	1000
macro avg	0.93071	0.93108	0.93081	1000
weighted avg	0.93123	0.93100	0.93103	1000

(a) Best NN from Assignment 1 (Wine Dataset)

	precision	recall	f1-score	support
0	0.34	0.98	0.51	315
1	0.86	0.01	0.03	403
2	0.73	0.21	0.33	282
micro avg	0.38	0.38	0.38	1000
macro avg	0.64	0.40	0.29	1000
weighted avg	0.66	0.38	0.26	1000
samples avg	0.38	0.38	0.38	1000

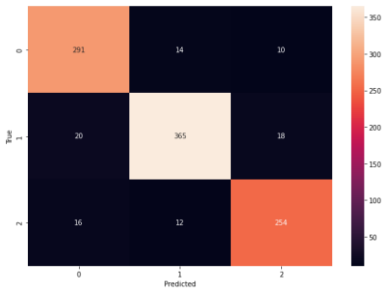
(b) NN – Random Hill Climbing

	precision	recall	f1-score	support
0	0.89	0.92	0.91	315
1	0.93	0.91	0.92	403
2	0.90	0.90	0.90	282
micro avg	0.91	0.91	0.91	1000
macro avg	0.91	0.91	0.91	1000
weighted avg	0.91	0.91	0.91	1000
samples avg	0.91	0.91	0.91	1000

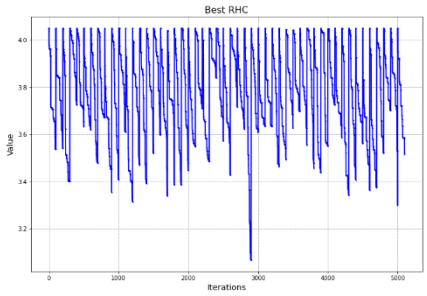
(c) NN – Simulated Annealing

	precision	recall	f1-score	support
0	0.72	0.77	0.75	315
1	0.81	0.78	0.80	403
2	0.74	0.72	0.73	282
micro avg	0.76	0.76	0.76	1000
macro avg	0.76	0.76	0.76	1000
weighted avg	0.76	0.76	0.76	1000
samples avg	0.76	0.76	0.76	1000

(d) NN – Genetic Algorithm



(e) NN SA Confusion Metrics



(f) NN RHC Iteration Curve

Figure 5. Wine Dataset Neural Network with Randomized Optimization Models