# Randomized Optimization for 4-Peaks, n-Queens, and Neural Network Weight Training

Shawn Egan
*Georgia Institute of Technology*
*segan3@gatech.edu*

## I. Introduction

Three Randomized Optimization (RO) algorithms are applied to two discrete optimization problems: 4-peaks and n-queens. The three ROs are randomized hill climbing (RHC), simulated annealing (SA), and a genetic algorithm (GA). The purpose of the comparison on two separate problems is to elucidate the strengths and weaknesses of each RO on different tasks. Each RO is optimized on problems of different sizes for both 4-peaks and n-queens; results are then compared.

The same three ROs are then applied to a Neural Network (NN) weight training task. The NN architecture is optimized for a Spotify binary classification task with slight modifications from Assignment 1. Changes to the classification task are discussed in Section II: Methods. The NN is initially trained using gradient descent (GD), in order to have a baseline for comparison. Hyperparameters are tuned for all 3 ROs to maximize the performance of each algorithm, and then all 4 optimization algorithms are compared.

### A. The 4-Peaks Problem

The 4-peaks problem is concerned with optimizing the number of leading ones and trailing zeroes in a bit-string of $n$ bits. The fitness of a given bit-string also depends on a parameter, $T$, which is set to a given proportion of $n$. The fitness function is the maximum of the number of leading ones and trailing zeroes, plus a bonus of $n$ if the maximum leading ones and trailing zeroes are both greater than $T$. This fitness function leads to two global maxima (one with more leading ones and one with more leading zeroes), plus two local maxima (all ones or all zeroes: no bonus $n$). Increasing the value of $T$ leads to larger basins of attraction around the local maxima, making the problem more complex. The search space grows on the order of $2^n$, meaning larger bit-strings are computationally challenging to optimize. The complexity of larger problems in addition to the existence of local minima makes 4-peaks a good candidate for comparison of the RO algorithms.

### B. The n-Queens Problem

The n-queens problem pertains to organizing $n$ queens on an $n \times n$ chessboard such that no two queens are attacking one another. A given state $x$ of the n-queens problem is represented as an $n$-length vector of integers. The individual $x_i$'s represent the row location of the queen in column $i$. Fitness of a given state is computed as the number of non-attacking queens, therefore posing this as a maximization problem. The n-queens problem has a simple definition, but the search space grows on the order of $n^n (\text{states} = \Pi_{i=0}^{n-1}(n^2 - i))$, making it a nontrivial problem that will allow detailed comparison of the RO algorithms. Additionally, n-queens does not contain local maxima and therefore will allow RHC and SA to avoid becoming stuck in local basins.

### C. NN Weight Training

Neural network weight training is a significantly more complex task as compared to 4-peaks and n-queens. In the NN selected for comparison, there are 100s of continuous weight values to optimize. Optimizing the weights of a NN will allow detailed comparison of the RO algorithms on a large, continuous-valued problem. Additionally, the performance of the different models will provide additional metrics for comparison.

**Hypothesis 1:** *The NN utilizing GA optimization will perform worse than RHC and SA.*

*Justification*

Genetic algorithms are computationally expensive to compute [1]. The GA uses a "population" of problem states which, in the case of NN training, are individual settings for the model weights. The GA then has to evaluate the fitness of each individual state, and generate a new population to evaluate at the next generation. The process of evaluating each set of NN weights for the entire population involves orders of magnitude more computations than RHC or GA.

**Hypothesis 2:** *The NN utilizing SA optimization will outperform the RHC NN.*

*Justification*

SA is similar to RHC but uses additional parameters to switch from exploration of the space to exploitation [3]. RHC follows the direction of increasing fitness function, whereas SA has a small probability of selecting a random direction to explore. As training continues, the probability to select a random direction decreases and SA begins following the direction of increasing fitness. SA will outperform RHC because SA can explore more of the weight space due to the use of the temperature decay.

**Hypothesis 3:** *All three RO NNs will perform worse than GD.*

*Justification*

Gradient descent is the standard optimization algorithm used for neural networks [2]. If one of the RO algorithms generally performed better than GD for NN weight training, it would be more popular in the field of machine learning. The small dataset size and simple binary-classification problem should allow GD to show its strength in comparison to the RO algorithms.

## II. Methods

### A. Four-Peaks and n-Queens

The 4-peaks problem is inherently a maximization problem, therefore no modifications are necessary. The n-queens problem is traditionally a minimization problem where the number of attacking-queens is minimized. A custom fitness function is used for n-queens experiments that translates it to a maximization problem: fitness is incremented every time a queen does not attack another queen. The number of checks for queen interactions given a board of size $n$ is $\sum_{i=1}^{n-1} i$, which is the maximum fitness for a board of size $n$.

Each RO algorithm's hyper-parameters are optimized on eight 4-peaks problems ($T = 0.25$) of different sizes, as well as 4 n-queens problems of different sizes. Five random seeds are used for each optimization problem for reproducibility and to avoid the influence of outlier performance of individual algorithms. All three RO algorithms are optimized on attempt number, and there is no maximum iteration number set. The restart number for RHC, the temperature and decay rate for SA, and the population size and mutation probability are optimized for GA. Optimal parameters are listed in the results for each RO algorithm on each optimization problem for select sizes.

Once optimal parameters are found for all problems, performance of each RO algorithm is compared across all problem sizes for 4-peaks and n-queens. The best fitness, run time, and number of function evaluations are compared for each problem.

### B. NN Weight Training

*1) Modified Classification Problem*

The Spotify classification problem from Assignment 1 involves using a multi-class classifier to predict the "size category" of a hit-song. "Size category" is defined by the number of streams; songs are grouped together if they have a similar number of streams. Five different streaming categories are defined. This classification task was modified to create a binary-classification problem: the lowest 4 categories (streaming number <700M) are combined into one category, and the largest category is selected as the positive label.

*2) Hyper-parameter Tuning*

Hyper-parameter tuning is performed with 5-fold cross validation to select a gradient descent model for the new classification problem. Learning rate, hidden layer sizes, and activation functions are included in the search space, and $F1_{macro}$ is selected for scoring . The top 5 performing models are shown in Table I, where 'Performance' is the validation score of the given model compared to the rank-1 model, and 'Speed' is the relative fit time of the given model compared to the rank-1 model. Models 1-4 utilize a Sigmoid activation function, and model 5 uses ReLU. Model 3 was selected for comparison of the RO algorithms, as it has 98% of the performance of the top model but 51x faster training speed. This increased speed will assist in model training using the RO algorithms, as the RO algorithms are not as efficient as back propagation. All 3 RO models use the same hidden layer sizes and Sigmoid activation function as Model 3 from Table I and 5-fold cross validation for hyper-parameter tuning. The RHC model is tuned on number of restarts and learning rate. The SA model is tuned on learning rate and temperature decay; initial temperature was found to have no effect on NN training. The GA model is tuned on population size and mutation probability. Learning rate for RHC and SA algorithms is an effective 'step size' used when computing neighbor weights, since NN weight optimization is a continuous optimization task.

TABLE I: Top Models

| Rank | Layers | LR | Performance | Speed |
|---|---|---|---|---|
| 1 | (16,8,8) | 0.0001 | 100% | 1x |
| 2 | (16,8,8) | 0.001 | 100% | 10x |
| 3 | (8,8) | 0.01 | 98% | 51x |
| 4 | (8,8) | 0.0001 | 98% | 1x |
| 5 | (8,4,2) | 0.001 | 97% | 1x |

*3) Model Comparisons*

Once optimal hyper-parameters are selected for each of the 3 RO models, all models are trained with no maximum iterations set, and compared against each other and against gradient descent on training loss, training time, number of function evaluations, and other machine learning metrics.

## III. Results

### A. Four-Peaks and n-Queens

After hyper-parameter tuning, the optimal parameters for each algorithm for given problem sizes are shown in Tables II and III.

TABLE II: 4-Peaks Optimal Parameters

| | RHC | | SA | | GA | |
|---|---|---|---|---|---|---|
| n | Attempts | Restarts | Temp | Decay | Pop Size | Mut % |
| 5 | 10 | 1 | 1 | 1e-4 | 50 | 0.05 |
| 35 | 100 | 25 | 5 | 1e-4 | 200 | 0.1 |
| 65 | 100 | 100 | 1 | 2.5e-3 | 800 | 0.1 |
| 95 | 100 | 100 | 1 | 5e-4 | 400 | 0.1 |

TABLE III: n-Queens Optimal Parameters

| | RHC | | SA | | GA | |
|---|---|---|---|---|---|---|
| n | Attempts | Restarts | Temp | Decay | Pop Size | Mut % |
| 4 | 25 | 1 | 1 | 1e-4 | 50 | 0.2 |
| 12 | 50 | 100 | 2 | 1e-3 | 800 | 0.1 |
| 20 | 100 | 25 | 5 | 5e-4 | 400 | 0.4 |
| 28 | 100 | 100 | 1 | 2.5e-3 | 400 | 0.05 |

#### 1) Best Fit

Figs. 1 and 2 show the 5-seed average best fitness[1] for both problems over the ranges of problem sizes.

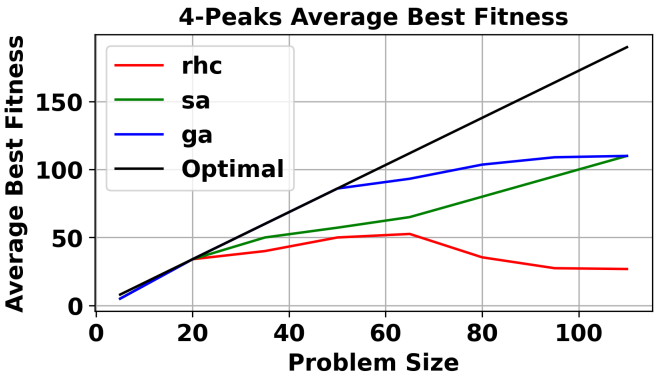


Fig. 1: 5-seed average best fitness for 4-peaks of different sizes

#### 2) Function Evaluations

Figs. 3 and 4 show the 5-seed average number of function evaluations for the 4-peaks and n-queens problems over the range of problem sizes, respectively.

#### 3) Run Time

Figs. 5 and 6 show the 5-seed average algorithm run time for the 4-peaks and n-queens problems over the range of problem sizes, respectively.

[1]4-peaks for n=5 shows all three models returning non-optimal states, as a result of the fitness function incorrectly evaluating [1,1,0,0,0] or [1,1,1,0,0] as having fitness of 5.0

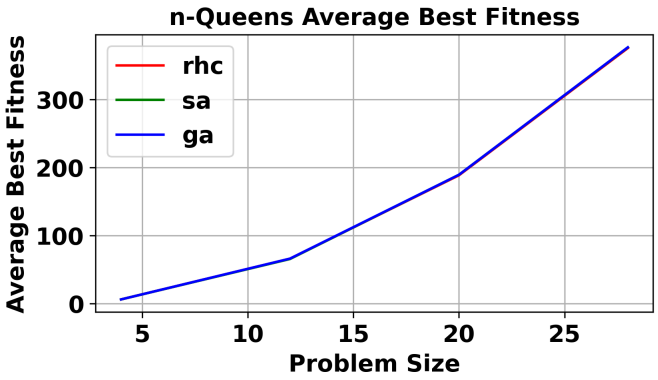


Fig. 2: 5-seed average best fitness for n-queens of different sizes

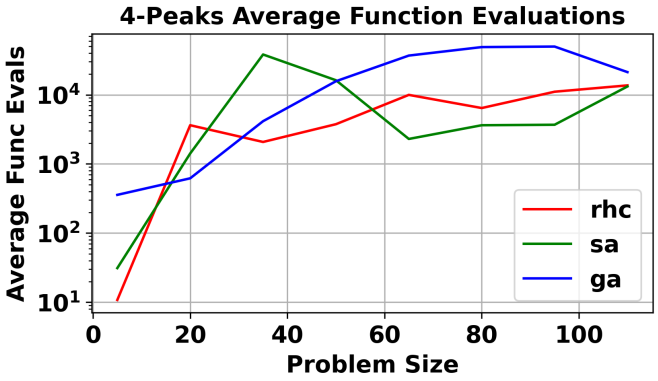


Fig. 3: 5-seed average number of function evaluations for 4-peaks of different sizes

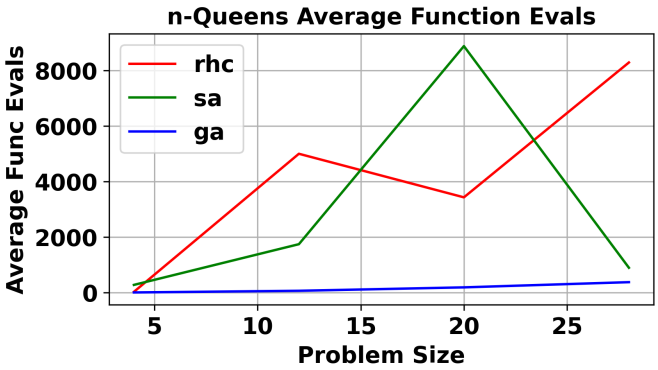


Fig. 4: 5-seed average number of function evaluations for n-queens of different sizes

### B. NN Weight Training

Table IV shows the fit time, number of function evaluations, $F1_{macro}$, and number of iterations for all 4 optimization algorithms on a NN with hidden layers (8,8) and Sigmoid activation function.

#### 1) Hyper-Parameters

Optimal model hyper-parameters for each algorithm are as follows:

- GD: learning rate: 0.01
- RHC: step size - 0.5, restarts - 20

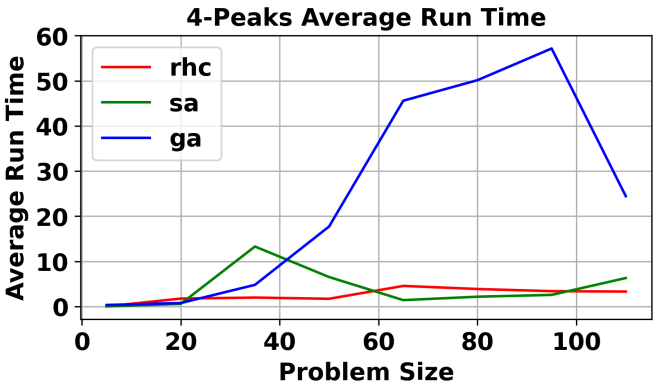


Fig. 5: 5-seed average run time for 4-peaks of different sizes

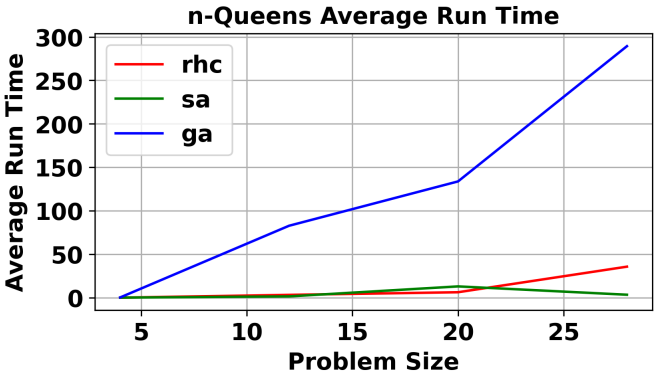


Fig. 6: 5-seed average run time for n-queens of different sizes

TABLE IV: Model Comparison

| Algorithm | Time (s) | FEvals | $F1_{macro}$ | Iterations |
|---|---|---|---|---|
| GD | 0.3 | 401 | 0.86 | 136 |
| RHC | 5.6 | 4678 | 0.85 | 453 |
| SA | 24.0 | 17834 | 0.83 | 45518 |
| GA | 56.1 | 80703 | 0.78 | 196 |

- SA: step size - 1.0, decay - 0.15
- GA: pop size - 400, mutation prob - 0.001

Fig. 7 shows the effect of restart number on validation performance for the RHC model. Fig. 8 shows the effect of decay rate on validation performance for the SA model.

Fig. 9 shows the effect of step size on validation performance for the SA and RHC models.

Fig. 10 shows the effect of mutation probability on validation performance for the GA model.

Fig. 11 shows the effect of population size on validation performance for the GA model.

*2) Learning Curves*

Fig. 12 shows the learning curves for all four models, where the red curves are training score and green curves are validation score.

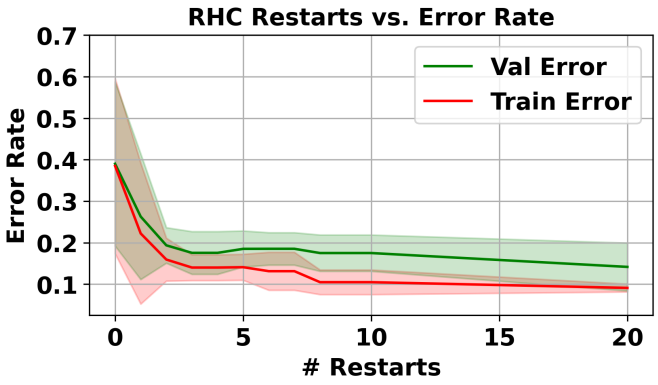


Fig. 7: RHC error rate vs restarts

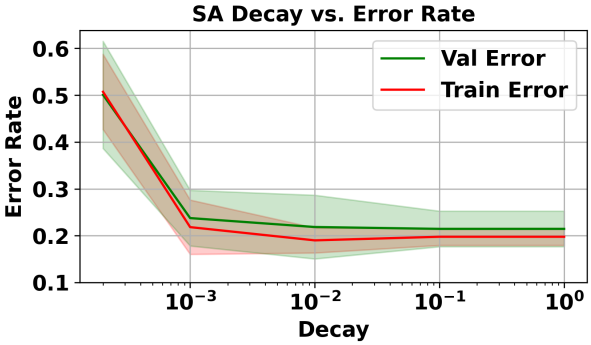


Fig. 8: SA error rate vs decay

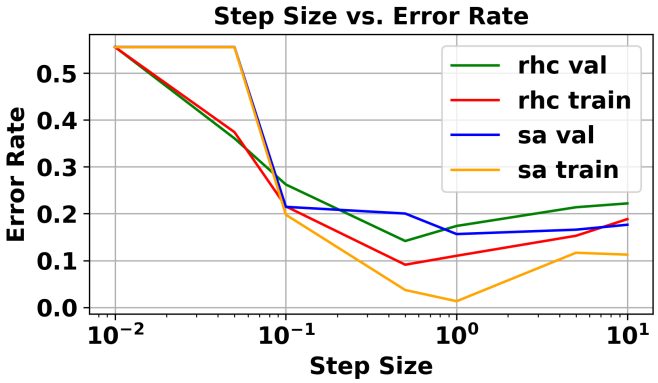


Fig. 9: SA and RHC error rate vs step size

*3) Training Loss & Time*

Fig. 13 shows the cross-entropy loss over training iterations for all four optimization algorithms.

Fig. 14 shows the cross-entropy loss over training iterations for simulated annealing, since Fig. 13 does not display the full range of training iterations for simulated annealing.

Similar to Fig. 14, Fig. 15 shows the full range of cross-entropy loss for random hill climbing.

*4) Function Evaluations*

Fig. 16 shows the log-linear plot of function evaluations over the first 250 training iterations for all 4 optimization algorithms.

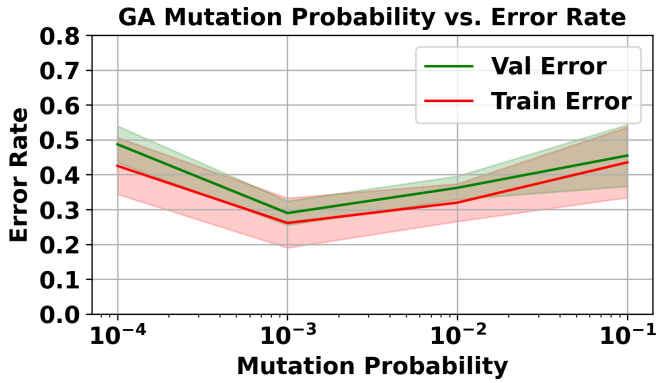


Fig. 10: GA error rate vs mutation probability (maximum iterations = 5000)

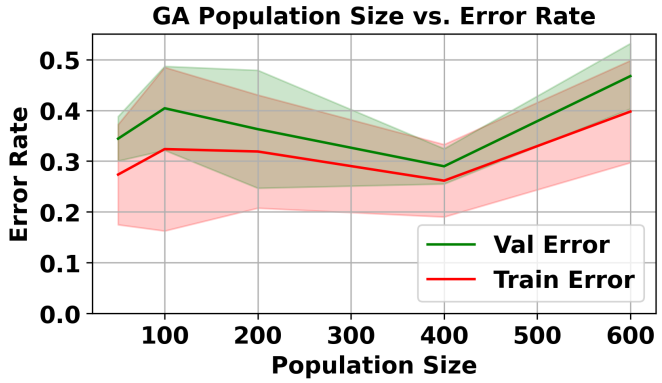


Fig. 11: GA error rate vs population size (maximum iterations = 5000)

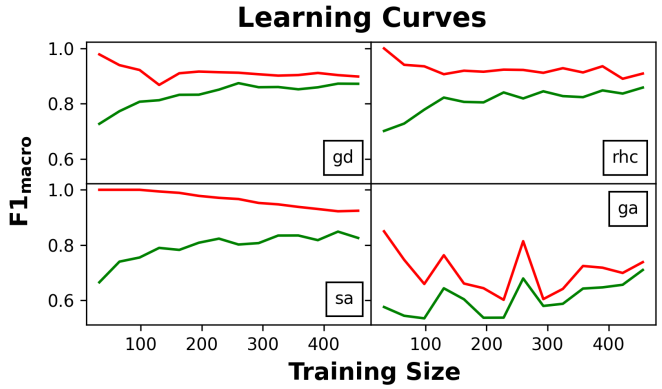


Fig. 12: All 4 model learning curves

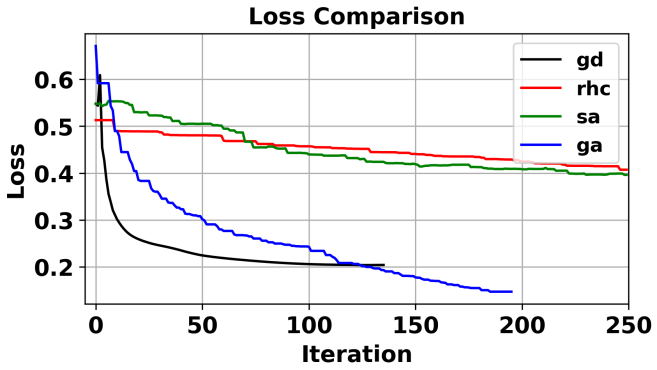


Fig. 13: All 4 model training cross-entropy losses

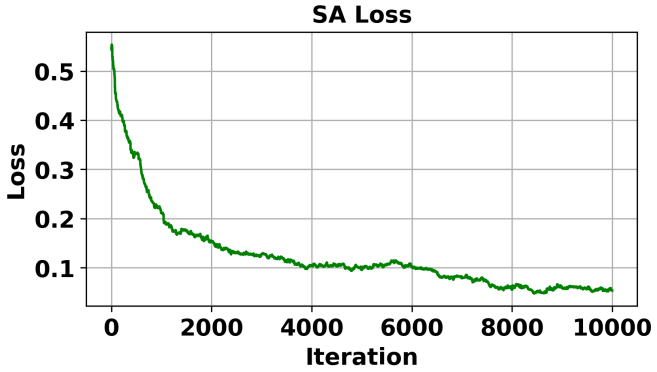


Fig. 14: Simulated annealing cross-entropy loss


Fig. 15: Random hill climbing cross-entropy loss

## IV. Discussion

### A. Four-Peaks and n-Queens

#### 1) Fitness

From Figs. 1 and 2, the performance of RO algorithms over problem sizes for both 4-peaks and n-queens can be compared.

All three ROs are able to find optimal states for all n-queens problem sizes. This is due to the nature of the n-queens problem: the number of optimal solutions grows on the same order as the state space (approx. $(0.143n)^n$[4]). Assuming optimal solutions are evenly distributed through the state space, random optimization should be able to traverse enough of a given subspace to reach one of the global maxima. There are no local maxima, therefore RHC and SA do not get stuck in local basins.

The difference in performance of the RO algorithms is apparent for the 4-peaks problem. Since the 4-peaks problem contains 2 local maxima and 2 global maxima, the RHC and SA algorithms are seen to under-perform the GA algorithm for problem sizes greater than 20 bits. This is due to RHC and SA becoming stuck in the local

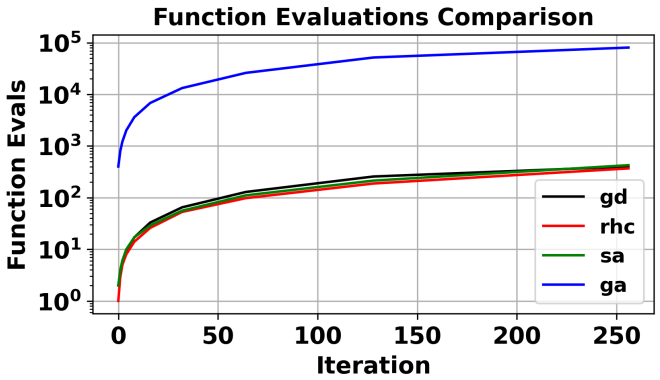


Fig. 16: All 4 model function evaluations

maxima basins. Table II shows the optimal number of restarts for RHC increasing from 1 to 25 to 100 over the range of problem sizes, as RHC needs additional restarts to search the problem space and attempt to escape the local maxima. For problem sizes over 65 bits, RHC shows decreasing fitness. This could be avoided by increasing the number of restarts beyond 100, at the cost of computational complexity.

SA uses larger initial temperatures and/or smaller decay values for larger 4-peaks problems. This is similar to RHC increasing the restart number; larger temperatures and smaller decay values allow the SA algorithm to explore more of the state space before switching to an exploitative strategy.

The GA finds global maxima up to 4-peaks problem sizes of 50 bits. Past problem sizes of 80 bits, GA struggles to improve optimization performance even with larger population sizes. Lower mutation probability would aid in performance improvements for larger problem sizes, as the generated populations of future iterations would contain less random states from mutation. For $n = 80$ there are $1.2e24$ possible states for the algorithm to evaluate. Increased population sizes would help GA see more of the problem space for evaluation.

*2) Function Evaluations*

Figs. 3 and 4 show the stark differences of the ROs on the two discrete problems.

For 4-peaks, all 3 RO algorithms use between $10^3$ to $10^{4.5}$ function evaluations for problem sizes greater than 20 bits. This is a result of the complexity of 4-peaks scaling on the order of $2^n$; for $n = 20$ there are 1,048,576 states but only 2 are global maxima. The number of required function evaluations scales exponentially for the 4-peaks problems, and every problem only has 2 global maxima. The problem gets exponentially harder with increasing size.

For n-queens, GA uses significantly less function evaluations than RHC and SA. This is due to the nature of the state space for n-queens in conjunction with the utilization of populations by GA. GA generates hundreds of different chess board states and evaluates each one at every iteration. RHC and SA need orders of magnitude more function evaluations in order to calculate the next state step.

*3) Run Time*

Figs. 5 and 6 show the difference in iteration time for the two problems. GA has exponentially increasing run-times up to a problem size of 65 bits for 4-peaks. Past the problem size of 65 bits, GA is limited by the population size and cannot discover the global optimum. As problem size increases, the 4-peaks state space becomes exponentially more sparse with respect to the global optima. GA has approximately linearly increasing run-times for n-queens problems of increasing size. Since the number of optima increases at a rate similar to the size of the state space, the sparsity of the state space with respect to solutions increases quadratically ($n^2$). If n-queens problems of larger size are evaluated, GA run time would be expected to increase quadratically. RHC and SA have much faster run-times for n-queens even though RHC and SA use many more function evaluations. This is due to the complexity of evaluating hundreds of states in each population for the GA algorithm. Though GA uses significantly less function evaluations for n-queens, each function evaluation is much more computationally complex than those for RHC and SA.

## B. NN Weight Training

*1) Hyper-Parameters*

Fig. 7 shows the importance of restarts for RHC optimization. Without restarts, the RHC algorithm is unable to explore the weight space effectively to reach an optimal state. Greater than 8 restarts shows negligible validation performance increase, and greater than 20 restarts (not shown) gives no performance increase. If there are too few restarts, the algorithm is likely to get stuck in local optima and cannot explore the full weight space. These results reinforce the importance of optimizing the restart parameter for the RHC NN.

Fig. 8 shows the effect that temperature decay has on validation performance for SA optimization. Decay rates that are too small ($< 10^{-3}$) do not end up transitioning from exploration to exploitation in under 10k iterations, and end up randomly wandering the state space. Decay rates greater than $10^{-3}$ are able to effectively transition from exploration to exploitation and can move towards optimal weight settings.

Fig. 9 shows the influence of step size on RHC and SA algorithms. For RHC, the optimal step size is 0.5 for the given NN. This step size is an order of magnitude higher than the learning rate for GD. Step size for RHC is used to generate "neighbor" weights which are then scored and evaluated as potential new states. RHC necessitates a large enough value for step size so that the next state evaluated is sufficiently different in terms of training score.

For SA, the optimal step size is 1.0 for the given NN. This step size is used in the same way as for RHC, and is 2 orders of magnitude higher than the learning rate

for GD. The step size for SA is twice that of RHC due to SA utilizing the exploration stage (hot temperature). SA needs a higher step size in order to explore more of the state space during the exploration stage.

Fig. 10 shows the effects of mutation probability for GA optimization (pop=400) with a maximum of 5000 iterations. Iterations were limited to enhance computation speed. Mutation probabilities too small ($< 10^{-3}$) do not allow proper exploration of the weight space. Without mutations, the algorithm is only evaluating combinations of weights already seen. Mutation probabilities too large ($> 10^{-3}$) are unable to exploit the performance of previous iterations, as mutations may remove weights that perform well.

Fig. 11 shows the effects of population size on GA optimization (mutation probability=0.001) with a maximum of 5000 iterations. Population sizes greater than or less than 400 show worsening validation performance. Smaller population sizes do not allow enough exploration of the weight space, and larger population sizes need smaller mutation probabilities to limit the number of mutated child states.

*2) Learning Curves*

Fig. 12 shows that GD, RHC, and SA learn over the full range of training sizes. GA, on the other hand, shows erratic behavior for training sizes below 400 samples. This suggests that the GA optimizer will benefit from larger training sets, as the training and validation curves improve beyond training sizes of 400 samples. GD converges with the smallest training proportion, followed by RHC and then SA. The RHC optimizer would benefit from additional training samples, as the validation performance is on an upward trajectory at the furthest right point of the learning curve. The SA algorithm will not benefit from additional training samples, as the training curve flattens out at the furthest right point of the graph. The GD algorithm would not benefit from additional training samples, as the training and validation curves flatten out beyond 300 training samples.

These results show the power of gradient descent for neural network weight optimization, and the weakness of genetic algorithms for small datasets.

*3) Cross-Entropy Loss*

Figs. 13, 14, and 15 show the difference in required iterations for each optimization algorithm as well as the number of iterations required to find optimal weights. Gradient descent arrives near the minima in less than 25 iterations, shown by the steep decline to the left of the GD loss curve. The genetic algorithm has a similar shaped curve that is less steep and takes more iterations to arrive near the minima. Random hill climb needed double the iterations as compared to GA to arrive near the minima. Simulated annealing needed orders of magnitude more iterations than all other algorithms.

Simulated annealing requires more iterations due to the focus on exploration at earlier iterations. Since SA selects a random next state some of the time (high temperature), it will take more iterations to begin heading toward the minima. Once the temperature has "cooled off," SA begins exploiting the shape of the weight space and heading towards the minima.

*4) Function Evaluations*

Fig. 16 shows that gradient descent, random hill climb, and simulated annealing all use roughly the same number of function evaluations at each iteration. Since RHC and SA are computing a surrogate gradient (evaluating the slope/direction using the score of the neighboring weights), they are similar to GD for number of function evaluations.

The genetic algorithm uses roughly 2 orders of magnitude more function evaluations at a given iteration as compared to all other optimization algorithms. This is due to the evaluation of hundreds of weight matrices for every iteration's population.

*5) Iteration Number, Training Time, and Model Performance*

Table IV gives a summary of the four metrics investigated for each optimization algorithm. Gradient descent performed best with respect to time (0.3s) and iterations (136). The genetic algorithm had the highest training time (56.1s), but the second least iterations (196) behind gradient descent. RHC required only 453 iterations with the second fastest training time of 5.6s. Simulated annealing had the second longest training time (24s) and the highest number of iterations by 2 orders of magnitude (45k).

Gradient descent marginally outperformed both RHC and SA for $F1_{macro}$ score, and the genetic algorithm performed worst. RHC marginally outperformed SA for $F1_{macro}$ score, and SA took over 4x longer to train with 100x the iterations.

The genetic algorithm shows very little benefits for NN weight training, as it's high training time and required function evaluations result in worse performance compared to all other optimization algorithms. Even though GA required only 196 iterations, each iteration requires $\sim$0.288s for $\sim$400 function evaluations resulting in the poor training time observed.

Additionally, the genetic algorithm is poorly suited for the task of optimizing continuous NN weights. The crossover and mutation steps of the genetic algorithm do not simulate the heuristic of the other three algorithms: moving towards an optimal point on a surface. Crossover and mutation incorporate additional randomness in the new population for the next iteration.

## C. Conclusion

The investigation of performance on 4-peaks and n-queens enabled the comparison of the RO algorithms for two very different discrete-optimization problems. It was found that RHC and SA algorithms have a tendency to get "stuck" in local maxima (4-peaks), and this can be reduced for RHC by increasing restart size. SA is less prone to local maxima basins due to the use of the temperature parameter to explore, then exploit the state

space. The n-queens problem comparison showed that all three RO algorithms can be successful in problems with very large state spaces, and GA optimization is prone to long run times for problems with large state spaces.

The evaluation of ROs for NN weight training showed the strength of gradient descent in comparison. Additionally, the comparison of performance metrics like time and function evaluations showed the challenge that genetic algorithms pose with respect to computational complexity. The performance of RHC and SA algorithms was found to be similar with respect to the $F1_{macro}$ score, but SA required more iterations and had a slower fit time due to the use of the temperature parameter for exploration.

Hypothesis 1 is proven true by the results discussed. The GA NN is unable to outperform the RHC and SA algorithms, due to high computational complexity. Additionally, the GA is limited by the small dataset size and would perform better on larger datasets.

Hypothesis 2 is refuted by the results discussed. Taking only $F1_{macro}$ into account, RHC outperforms SA for NN weight training. RHC is able to outperform SA with significantly less training time and iterations. RHC's utilization of restarts leads to a more efficient exploration of the weight space for this NN as compared to SA's temperature decay.

Hypothesis 3 is proven true by the results discussed. GD outperforms all 3 RO algorithms for all 4 metrics considered. This result is supported by decades of research in the field of machine learning.

### D. Limitations and Future Work

The performance of the GA optimizer is limited by computational capacity. Large problem sizes for 4-peaks, as well as the NN weight training would benefit from additional compute power.

Additionally, the use of the Spotify binary-classification task with 800 sample dataset limits the conclusions drawn from the experiments. Further investigation involving comparison of performance on separate machine learning tasks with larger datasets will allow broader conclusions to be made about each RO algorithm.

## V. Resources


[1] Nopiah, Z. M., Khairir, M. I., Abdullah, S., Baharin M. N., and Arifin A. "Time Complexity Analysis of the Genetic Algorithm Clustering Method". *World Scientific and Engineering Academy and Society (WSEAS)*, pp. 171-176. Accessed: March. 2, 2025. doi: 10.5555/1807817.1807849. https://dl.acm.org/doi/abs/10.5555/1807817.1807849

[2] Ruder, S. "An overview of gradient descent optimization algorithms". *Insight Centre for Data Analytics, NUI Galway*. Accessed: March. 2, 2025. doi: https://doi.org/10.48550/arXiv.1609.04747 https://arxiv.org/abs/1609.04747

[3] Carnegie Mellon Univerisity, School of Computer Science. "Simulated Annealing". https://www.cs.cmu.edu/afs/cs.cmu.edu/project/learn-43/lib/photoz/.g/web/glossary/anneal.html

[4] Wikipedia. "Eight queens puzzle". *Wikipedia*. (2025). Accessed: Mar. 2, 2025. https://en.wikipedia.org/wiki/Eight_queens_puzzle

[5] Nakamura, K. "ML LaTeX Template". *GitHub*. (2023). Accessed: Feb. 5, 2025. https://github.com/knakamura13/cs7641-ml-study-materials-2023/tree/main