Author: Michael Pecorino | GT Username: mpecorino3 | Assignment: Randomized Optimization

Plagiarism trigger: I partially took this course in Fall 2020 so some of the analysis is repeated.

Introduction

This assignment explores four randomized optimization algorithms: Randomized Hill Climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA), and Mutual-Information-Maximizing Input Clustering (MIMIC). In the first part of the analysis, the four algorithms are evaluated on three problems to highlight the strengths of each algorithm. Table 1 provides a description of the three problems and an initial hypothesis as to which algorithms will perform the best for each problem. Each problem used a problem length of 1,000 to ensure there is enough complexity in the problem to visualize the differences among the algorithms.

Table 1: Description of problems and initial hypothesis for the best algorithm

Problem	Description	To show strength of:
Four Peaks	A 2-D optimization problem that has 2 global optima and 2 less optimal local optima, with wide valleys separating the peaks. This problem is designed in this way to show that some algorithms are more likely to get stuck in the valleys or converge at the local optima.	Genetic Algorithm
Knapsack	Models a camping/hiking trip in which we place objects needed for survival in a knapsack that has some weight capacity. The goal is to maximize the survival rating of the knapsack. The optimization problem is a balancing between object weight and object survival rating. The weights and survival ratings for each object were chosen randomly and in the range [1, 5]. This is similar to a linear regression problem, subject to the constraint which is the total weight of the knapsack.	MIMIC
Flip Flop	Counts the number of alternating bits in a bit string. For example, 1010101 has maximum fitness because all bits are alternating. This problem is very non-linear, and so it is expected that an algorithm that does well on this problem will be able to explore much of the hypothesis space.	Simulated Annealing

Each algorithm has parameters that can be tuned to further optimize performance. The parameters and ranges are provided in Table 2.

Table 2: Parameters to tune for each algorithm and their ranges

Algorithm	Parameters
Randomized Hill Climbing	• Random restarts [1, 100] • Initial state: None (randomly chosen)
Simulated Annealing	• Decay schedule for the Temperature parameter: exponential decay with initial temp in [100, 200, 300], final temp [.001], and exponential decay factor [.01]. • Initial state: None (randomly chosen)

Algorithm	Parameters
Genetic Algorithm	• Population size: [100, 200, 300] • Mutation probability: [0.1, 0.2, 0.3]
MIMIC	• Population size: [100] • Proportion of samples to keep in each iteration: [0.1, 0.2, 0.3]

In the second part of the analysis, RHC, GA, and SA are implemented on the same Neural Networks from Assignment 1 and the results from these three random searches are compared to the backpropagation method using gradient descent.

All of the optimization was completed with the mlrose [1] Python package. All coding credits go to the package author, Genevieve Hayes. Visualization of the results presented here come from custom data collection and summaries of the data.

Part 1: maximizing fitness on toy problems

There are at least three ways we can compare the random search algorithms:

- Maximum fitness based on the number of attempts to find a better neighbor, or state, at each time step. Algorithms that require less attempts may be consider more efficient.

- Maximum fitness based on the number of function evaluations. In addition to achieving high fitness, we might be interested in how many calculations are required to get there.

- Maximum fitness based on the total time required. Just because an algorithm requires more calculations does not necessarily mean it requires more time, so function evaluations and time should be considered together.

This section is split by the problem type: four peaks (section 1.1., knapsack (section 1.2), and flip flop (section 1.3). Each section contains an analysis of the results followed by one figure for each individual algorithm that shows the max fitness over the number of iterations. Algorithms with more parameters, e.g. mutation probability in Genetic Algorithms, are shown through line coloring. The scale for the y-axis on these plots is customized to the algorithm so that the individual algorithm's trend can be observed first. A fifth plot contains the fitness comparison of all four algorithms on the same plot, in which it will be made clear which algorithms perform well for each problem. In all three problems, MIMIC will have the top performance when it comes to the maximum fitness, and there is a clear difference in the performance of the remaining three algorithms - GA also has high fitness on Four Peaks and SA on Flip Flop. However, maximum fitness is not the only output we should track. To conclude, each section a table is provided to show the number of function evaluations and total time required by each algorithm to acquire the best fitness. RHC and SA are generally very fast, but can require significantly more calculations, and often settle on local optima.

1.1. Four Peaks

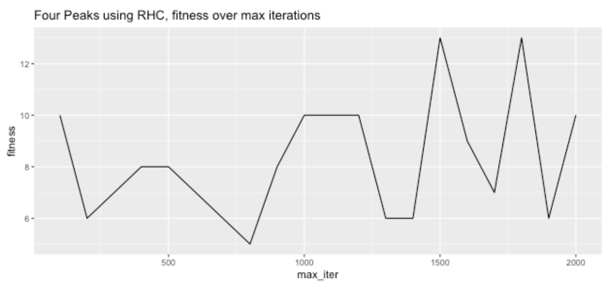
The max fitness found was 33 from MIMIC. This problem is designed so that some algorithms, like RHC and SA, are more likely to get stuck in the valleys or converge on a local optima, so it is expected that GA and MIMIC would perform much better.

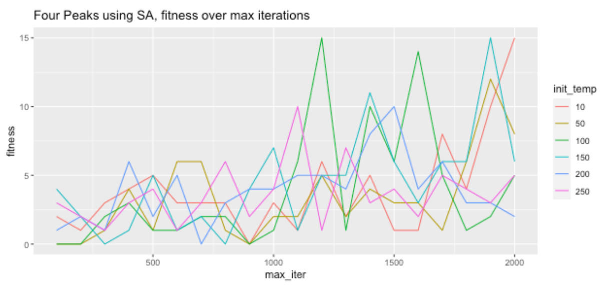
For Simulated Annealing, the initial temperature was varied using the same exponential decay schedule as initial temperatures of 10, 100, and 150 all eventually achieved the same maximum fitness. This shows that the complexity of the problem is not a good fit for SA, or just many more iterations, random restarts, or more complicated decay schedules are required.

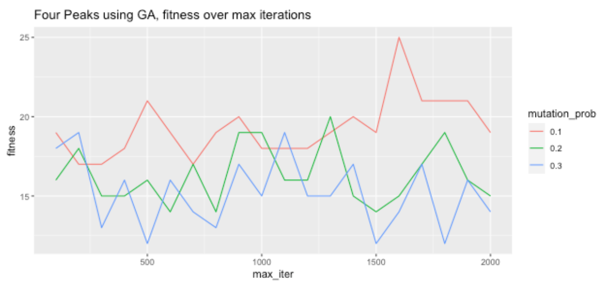
For GA, there is some evidence of separation between the three different mutation probabilities as a 10% mutation probability gives the max fitness and is almost monotonically better as the iterations are increased. There is similar separation between 20% and 30% mutation rates. This may indicate that, for the four peaks problem, over mutating the most fit hypotheses can steer the algorithm in the wrong direction. In search of better results, mutation probabilities between 0.01 and 0.09 should be tried.

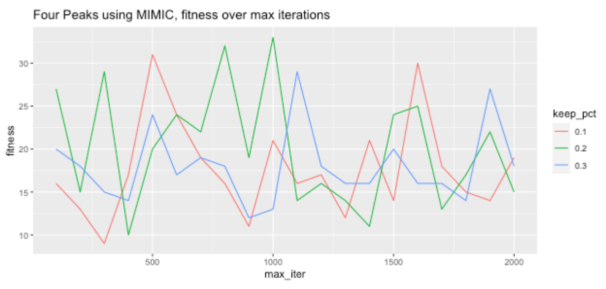
MIMIC not only provides the maximum fitness of 33, but does so in significantly less iterations. In fact, MIMIC was able to beat GA's best fitness approximately 16x faster (1,600 iterations to achieve a fitness of 25 in GA vs. 100 iterations to achieve a fitness of 27 in MIMIC). MIMIC's ability to model the structure of the four peaks problem provides the advantage needed to provide the best fitness. For MIMIC, there is no clear indication that one value of the keep percentage was better than another, although a value of 20% seems to be optimal. It should be noted though that GA performs much better than RHC and SA, and so is highlighted for this problem.

While MIMIC and GA perform the best, they also take require significantly more time to compute relative to RHC and SA, and MIMIC is about 45x slower than GA. These results must be considered in a practical setting, because sometimes it is required to balance performance with time. These algorithms require more time due to their complexity, but they are more efficient in that they also perform less calculations compared to RHC and SA.









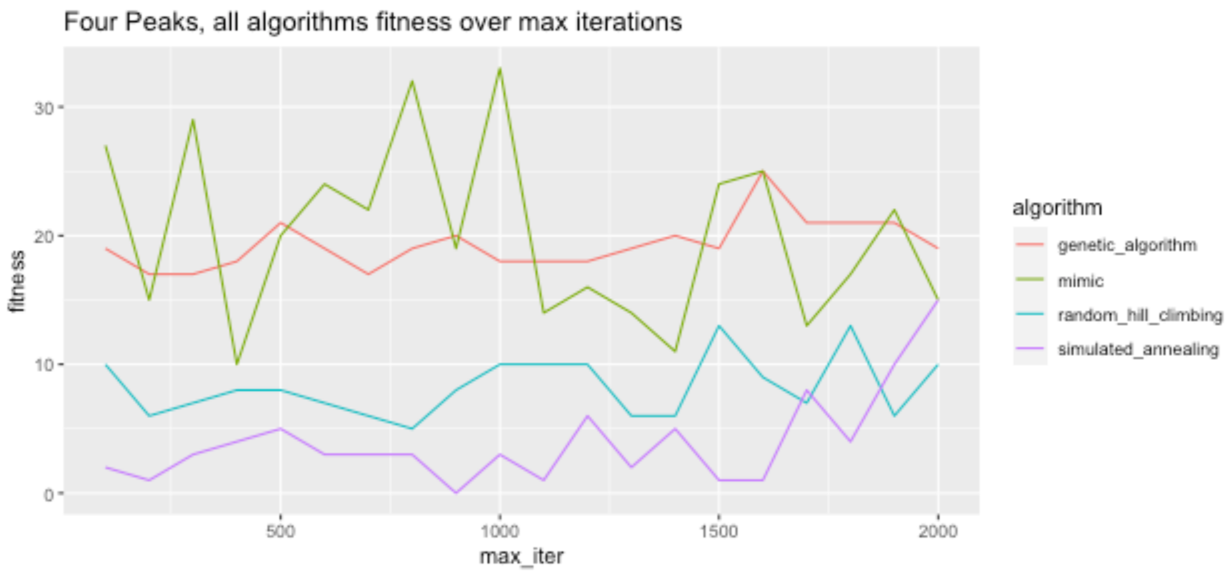


Table 3: Four Peaks function evaluations and time required

Algorithm	Best fitness	Function evaluations	Time (Seconds)
RHC	13	412	0.03
SA	15	712	0.06
GA	25	16	2.66
MIMIC	33	38	90.54

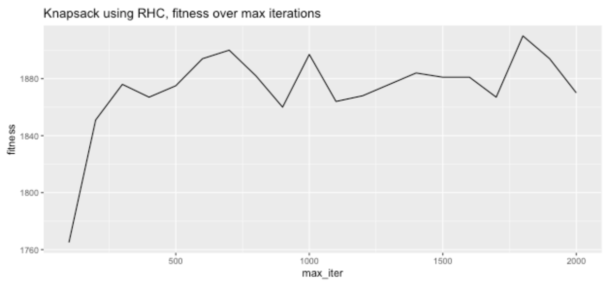
1.2 Knapsack

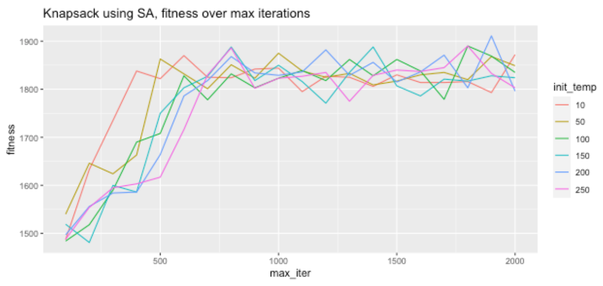
The max fitness found was 2,335 from MIMIC. Simulated Annealing was ranked second, RHC third, and Genetic Algorithm fourth.

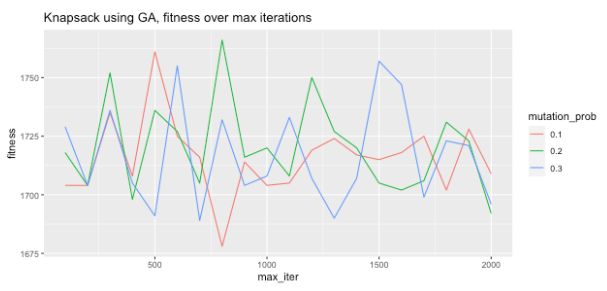
GA does not perform well on this problem, and there is much more overlap in the mutation probabilities compared to the four peaks problem. Due to the complexity of the Knapsack problem, the assumption that the most fit hypotheses can mutate or crossover to produce more fit hypotheses may not hold true.

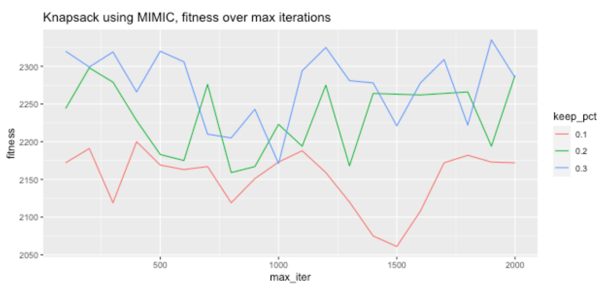
RHC increased its fitness almost linearly in the number of iterations up to about the 300th iteration, and then began to plateau with some small gains in fitness for the remainder of the iterations. SA increased less sharply than RHC at first, but was able to land at about the same plateau as RHC. Again, there is no clear indication that changing the starting Temperature has an impact on the result. It is interesting that both RHC and SA perform similarly for this problem. While SA has some balance between exploration and exploitation, it is not very helpful relative to RHC in terms of achieving a better fitness. However, SA can achieve the same fitness 20x faster.

For MIMIC, there is some separation between the various keep percentages, with higher keep percentages providing larger fitness. Due to the complexity of the knapsack problem, the results indicate that maintaining a larger portion of the most fit hypotheses is important for ensuring a more accurate structure can be derived.









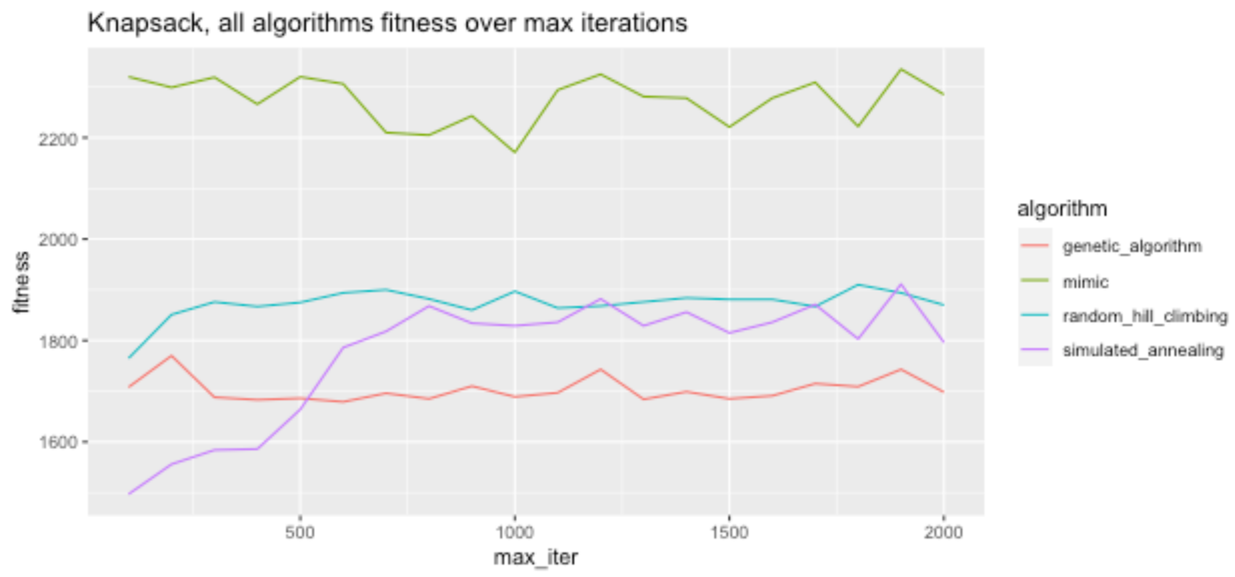


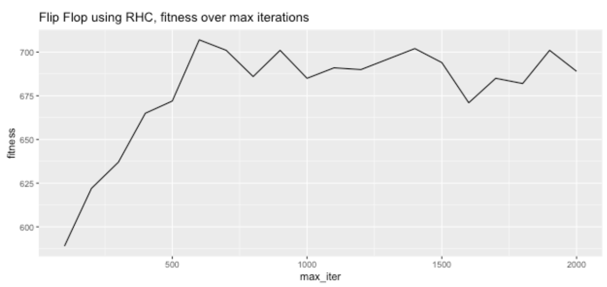
Table 4: Knapsack function evaluations and time required

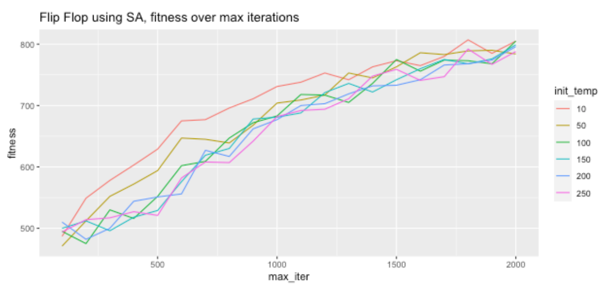
Algorithm	Best fitness	Function evaluations	Time (Seconds)
RHC	1,910	22,197	0.67
SA	1,911	752	0.03
GA	1,770	16	1.43
MIMIC	2,335	86	85.50

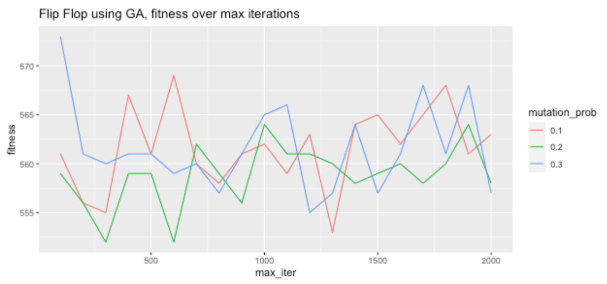
1.3 Flip Flop

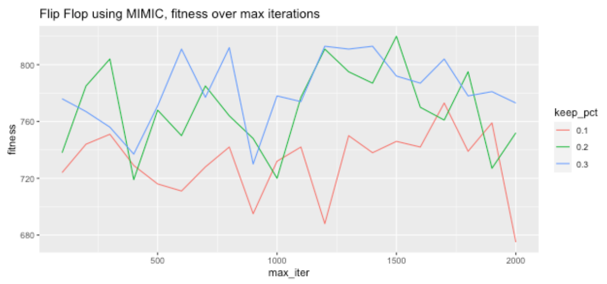
Again, MIMIC provides the best fitness at 820, with SA in close second with 807. RHC was ranked third and GA fourth. It needs to be stressed that the SA fitness may suffer from the lack of iterations here, as convergence may not have occurred yet at the 2,000th iteration. Based on this pattern in SA and the lack of any trend with MIMIC, it is reasonable to expect SA to have provided a better fitness if given more iterations. For SA, it seems that a low temperature was more important in the early iterations, but all of the temperature lines begin to overlap by the 1,500th iteration. Further, the fact that SA was so close to the performance of MIMIC and arrives at similar fitness 50x faster is noteworthy. That is why SA is highlighted for this problem.

Similar to the Knapsack problem, RHC increases almost linearly in the early durations and then plateaus and GA performs poorly on this problem with no clear trend in the mutation probability. For RHC, we can only increase the number of random restarts. For GA, we may want to consider increasing the population size and/or the mutation probability.









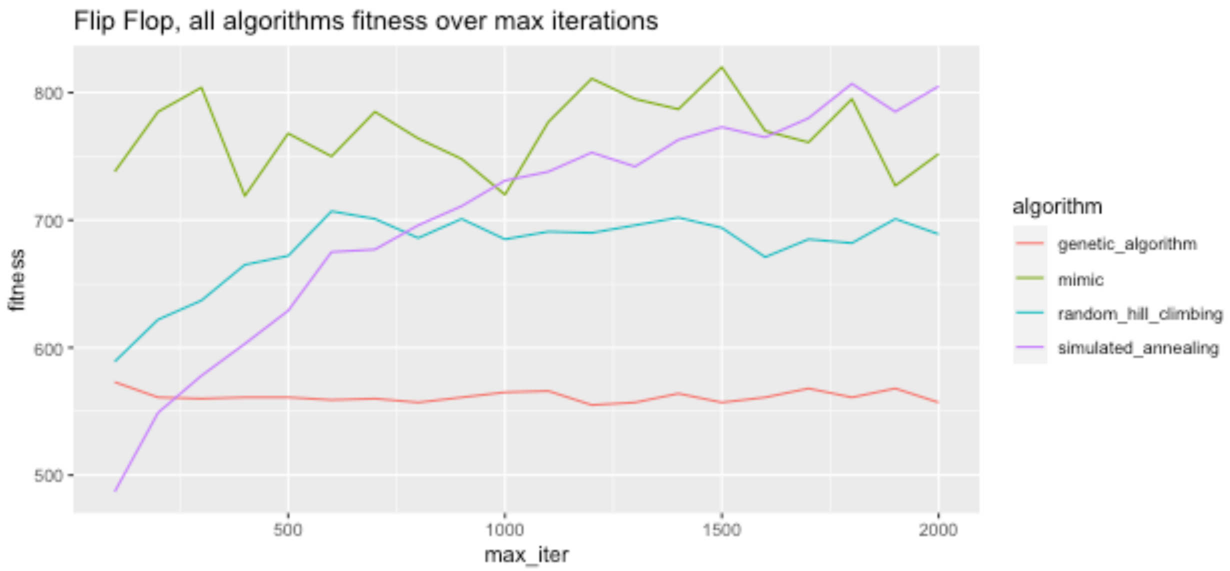


Table 5: Flip Flop function evaluations and time required

Algorithm	Best fitness	Function evaluations	Time (Seconds)
RHC	707	9,829	7.70
SA	807	1,722	1.07
GA	573	7	6.62
MIMIC	820	32	50.62

Part 2: using random search on Neural Networks

The Neural Network built in the previous assignment for the smartphone sensor (SENSOR) dataset is converted to a binary problem and fit using the RHC, GA, and SA random search methods. The response variable takes a 1 when the label is "sitting" (18.67%) and takes 0 otherwise (81.33%). The imbalance arises from the fact that the original problem had 6-classes, so making this a binary problem is really a 1 vs. all approach of one class vs. the other 5 classes. Model performance and total time required are compared to the back propagation method from the previous assignment.

2.1 Comparison of Neural Networks on SENSOR data

All four algorithms were given 50 iterations to get the best performance on the test data. Backpropagation with gradient descent outperformed any of the random search methods, but GA also provided good performance in less time. As expected, gradient descent required many more iterations, and was still improving its performance on the 50th iteration. GA was able to provide its best performance on the test data in just 10 iterations, but just like the top problems, requires significantly more time to complete each iteration.

RHC and SA suffer from the imbalance in the dataset. Because this 6-class problem was converted to binary, the solutions found by RHC and SA in 50 iterations are actually worse than always selecting the majority class.

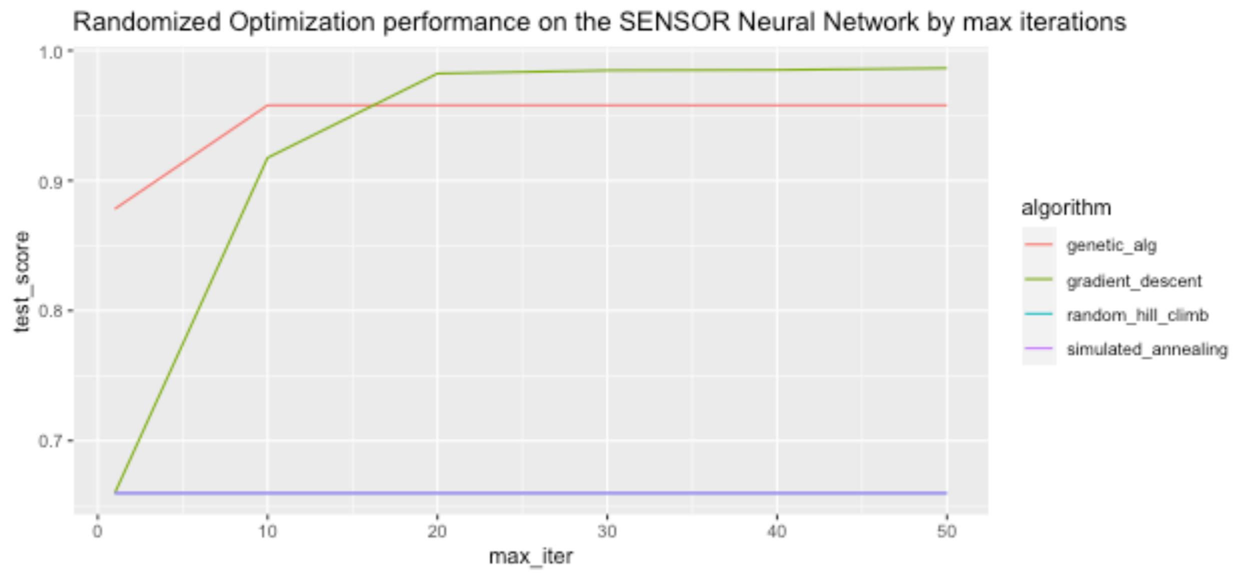


Table 6: Best performance on SENSOR Neural Network

Algorithm	Test accuracy	Iteration	Function evaluations	Time (seconds)
RHC	0.6597	50	48	1,182.45
SA	0.6597	50	46	24.89
GA	0.9580	10	5	2,179.86
Gradient Descent	0.9864	50	50	45.90

Conclusion

Randomized optimization algorithms provide alternative solutions to methods like gradient descent. Gradient descent is susceptible to converging on local optima and is dependent on the choice of initial weights. Just like we can vary the mutation probability of a genetic algorithm, the choice of initial weights is the parameter to vary for gradient descent.

MIMIC performed the best on all four problems. Algorithms like Random Hill Climbing, Simulated Annealing, and Genetic Algorithm don't have a memory property to help guide the search. For example, although with Random Hill Climbing we can run the algorithm with 100 restarts, the path to arriving at a fitness value makes no use of any prior iterations. As another example, although Genetic Algorithms carry some information forward in the evolutionary way, it is still a greedy algorithm that exploits the most fit hypothesis and makes the assumption that hypotheses can be combined or mutated and still provide good performance. MIMIC's advantage is its ability to create structure and a distribution of where the global optimum may be. While MIMIC has provided the best performance on the toy problems, it also requires the most amount of time to complete. It is important to keep this in mind in the practice setting when weighing the trade-off between model performance and time required to complete a Machine Learning model or solving an optimization problem.

Besides MIMIC, we can clearly see that Genetic Algorithm performed well on the Four Peaks problem and Simulated Annealing performed well on the Flip Flop problem. The results also indicate possible improvement opportunities. For Genetic Algorithm, we can continue searching for better fitness using higher population sizes or even lower mutation probabilities, since there is a trend with decreasing mutation probability and increased performance. For Simulated Annealing, the fitness over the number of iterations indicates that convergence was not reached yet. We could also try different decay processes other than exponential.

For the SENSOR neural network, gradient descent outperformed the random search algorithms in both performance and time. However, promise was shown with an algorithm like Genetic Algorithm. In just 5 function evaluations in 10 iterations, GA was able to come within 97% of the total performance of gradient decent. However, it required 50x more time for computation.

References

1. mlrose, Genevieve Hayes, https://mlrose.readthedocs.io/en/stable/index.html