CS 7641- Homework 2 Writeup - Randomized Optimization

Man Wai Yeung

March 3, 2019

Introduction

I have analyzed the performance for 4 randomized optimization algorithms in my analysis, Simulated Annealing, Randomized Hill Climb, Genetic Algorithm, and MIMIC. I used my Wine Quality dataset to formulate the ANN structure in Part 1. Afterwards, I defined three maximization problem for randomized optimizations algorithm to solve.

1. Continuous Hill Climbing - Best Performer: Simulated Annealing
2. Knapsack problem- Best Performer: MIMIC
3. Traveling Sales Man- Best Performer: Genetic Algorithm

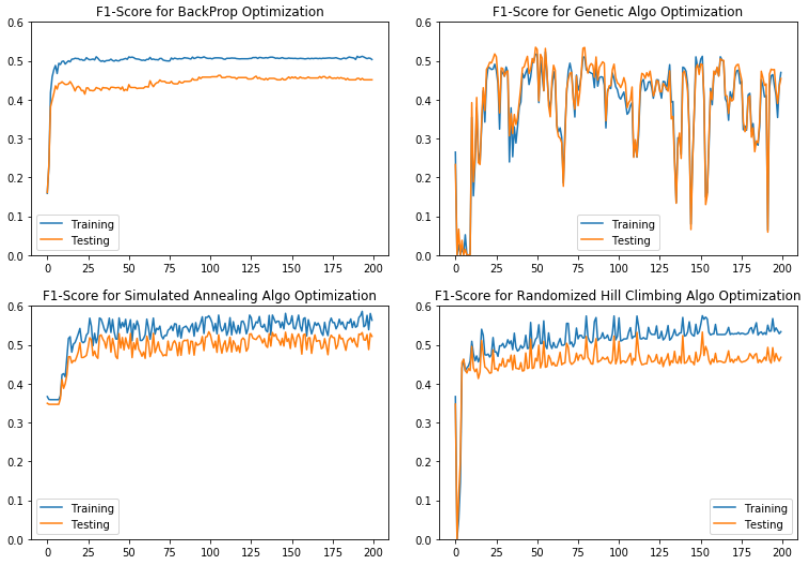
Neural Network Weights

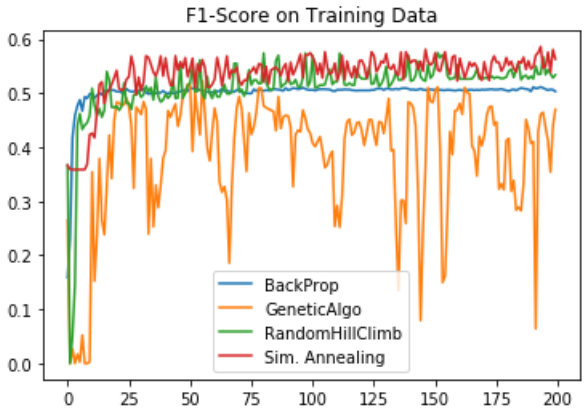
• I have used my wine quality dataset from HW1 and 'F1-score' as my metrics for evaluation. My dataset is relatively small and with 11 attributes. My previous analysis shows that F-1 score is difficult to be perfect and it will be a good metrics to differentiate performance.

ANN architecture: Based on past knowledge of this data set, the 'optimal' structure of hidden layer is one layer with 5 units, and activation function is logistic sigmoid. These were kept the same for all the tests.

• As a baseline model, I used backpropagation of 200 epochs to see how well the local search optimizations can perform compared to backpropagation in finding weights for neural network.

• Here are the visualizations to compare the testing and training metrics over iteration:





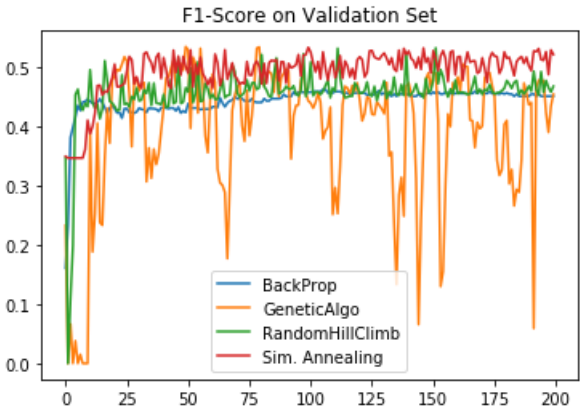


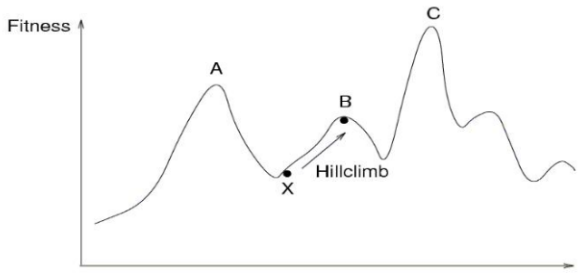
Table 1: Performance Comparison for optimizing ANN Weights

Algorithm	Final Testing Score	Final Training Score	Time Taken (Seconds)
Baseline	0.451	0.503	5.1
Genetic Algorithm	0.439	0.448	787
Randomized Hill Climbing	0.452	0.523	31.8
Simulated Annealing	**0.492**	**0.533**	197

Interpretation: The first observation I notice when comparing how the F1-score changes with respect to the four algorithms is that Simulated Annealing and Randomized Hill Clibings follows the curve in BackProp, which is monotonically increasing before plateauing. While SA and RHC are generally improving, it is choppier between epochs. Both simulated annealing and randomized hill climbing testing results exceeds Baseline model (BackProp). It is encouraging to see local search method for parameters can find a better results of weights than using numerical differentiation method, especially when function may not have a good gradient to analyze.
Why did RHC and SA work in this case? I think it has to do with weights are continuous values where you can finetune it in small steps within a neighborhood, similar to what stochastic gradient descent is doing, to take a small step into a direction of the gradient. In this case, the step is determined by improvement of fitness value within neighborhood and probability of the new value to do so. Local search works really well in this scenario.
Why did GA not function as well? I think this is largely due to the property of weights. If you have two possible functions in the form of $[ax_1 + bx_2 + cx_3$,$9x_1 + 81x_2 + 15x_3$, $30x_1 - 20x_2 + 18x_3]$, GA will combine weights to form a new function $9x_1 + 81x_2 + 18x_3$ by combining coefficients in the first and second function. However, weights are dependent on one another, switching over weights would affect dramatically how well the curve fits the data. In such case, the algorithm function as if generating randomly curves, that may find the general good directions, but so much randomness are built between iterations, which makes improvement to optimal points very difficult.
Overall Comparison Simulated annealing performs best. I would think in the long run, given backprop and simulated annealing with many iterations, both algorithms would be able to converge to some optima points (we know that backprop leads to local optima convergence, not necessarily global). However, in the comparison, simulated annealing required 20 times number of seconds to complete the optimization. For such a simple data set, it still takes 3 minutes to complete the run. I can see commercially, for large dataset with many attributes, time complexity for simulated annealing would increase in polynomial times compared to baseline. So in industry, backpropagation (baseline) may still be well adopted because of its efficiency, even though Simulated Annealing or Randomized Hill Climbing may be able to deliver better results.

Continuous Hill Climbing

Problem Statement: There is a 1-dimensional continuous function with multiple local maximas and the algorithm needs to perform local search to identify maximum of f(x). The problem seeks to illustrate advantage of **Simulated Annealing** (SA)
• In my formulation, the global maxima is 200 with 37 local minimas. No. of iterations = 20000 for each of the four

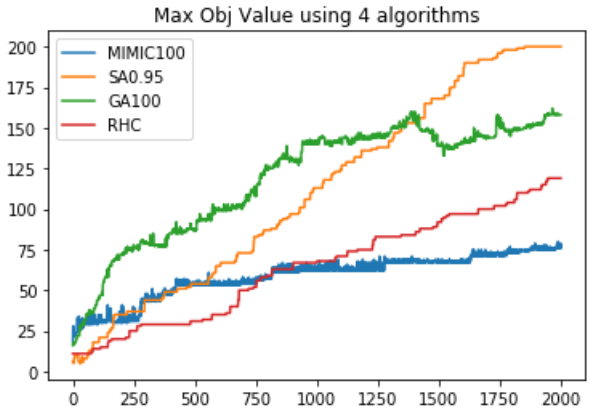


algorithms.

• I am going to compare both the time and performance for Randomized Hill Climbing, Genetic Algorithm, Simulated Annealing, and MIMIC.

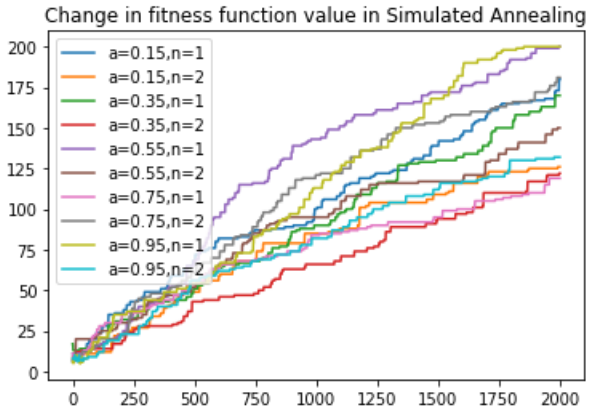
Table 2: Performance on Continuous Peak

Algorithm	Best Obj Value	Time Taken (Seconds)
MIMIC	76	379
Genetic Algorithm	158	1.72
Randomized Hill Climbing	119	0.16
Simulated Annealing	**200**	**0.0599**



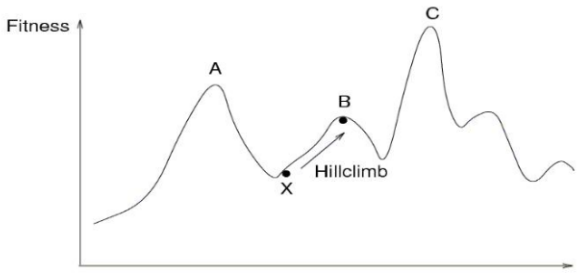
In this simple hill climb problem, Simulated Annealing (with α=0.9) performed best and genetic algorithm performed second. Randomized hill climb increases steadily only time but it does not improve as quickly as simulated annealing. MIMIC improves very slowly and is ranked last in maximum objective value achieved out of 200.

Why SA performed better than Randomized Hill Climbing in hill climbing? While Randomized Hill Climbing restarts at a random schedule once it gets stuck at local minima, Simulated Annealing has a finer balance between exploration and exploitation with cooling schedule, which enables it to find the global max fast and earlier.



Effects of tuning cooling schedule in Simulated Annealing For each alpha, which is the cooling exponent in cooling schedule, I conducted two experiments. High alpha represents slow decrease in temperature. The best results is obtained by a=0.95, meaning that slowly decreasing has the best result in looking for global maximum. When alpha in high like a=0.35, the best objective is only 100 because it is jumping around too much (i.e. too much exploration). However, we can see that

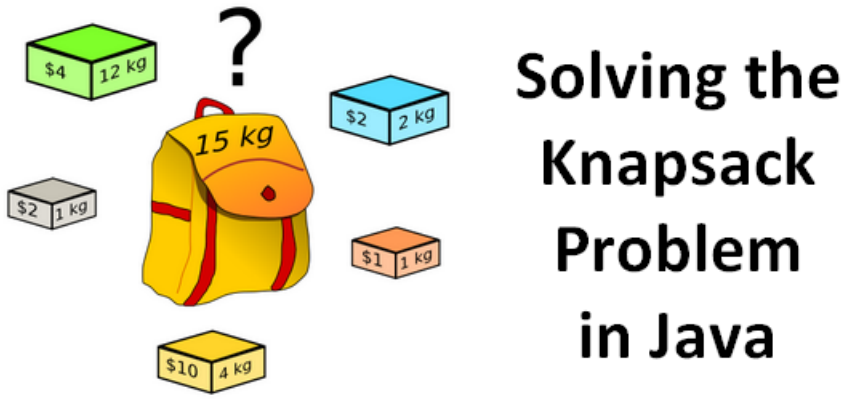
different trials yield different results. In one trial of a=0.95, best objective is 200 while the other is 120. It attests to the fact that simulated annealing does not guarantee convergence to global maximum. It has a PROBABILITY to reach the global maximum, which in some cases like that in trial 2, it was not able to reach the maximum value of 200.
Why Genetic Algorithm works pretty well in this case? There is only 1 parameter to construct this 37 peaks functions, which is x. While beginning no. of potential function is 50 in all trials, % crossover of crossover and % of mate varies between 10,30,and 50. In one dimensional space, it means the algorithm is **searching around sections in a line segment that both parents do well** and may add some perturbations to the offsprings. The best result comes from a balance between crossover and mutation (c=30, m=30). While doing a lot of crossover to combine weights from parents help search thoroughly in the hypothesis space, it is also important to do mutation to escape local maxima.



How would randomized hill climbing or genetic algorithm may be performing better than simulated annealing? When it is multivariate scenario with high-dimensional data, it would be more difficult for simulated annealing to work especially with presence of diagonal ridges since one variable may function well under one cooling schedule but another dimension remains jumpy around the same schedule.
Comment on time performance: In this one dimensional space, evaluation of function is fast and hence SA annealing didn't take much time, and in fact the fastest among all four algorithms. However, if faced with high-dimensional complex functions, where evaluating fitness function becomes expensive, I would expect the time difference between SA , GA, and RHC closing up the gap.

Knapsack Problem

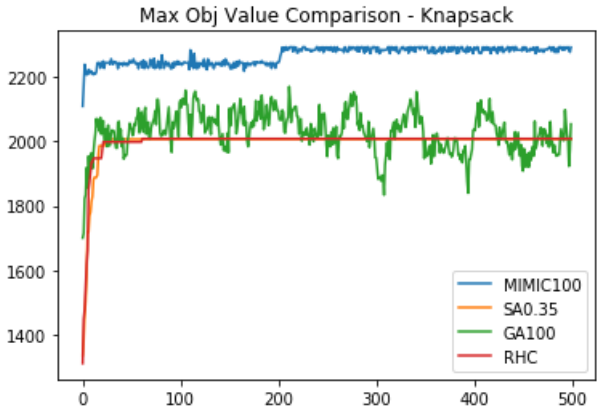
Problem Statement: There is a knapsack in which we want to put objects such that the objects carry the most combined value and that the combined weights and volume do not exceed knapsack's capacity.



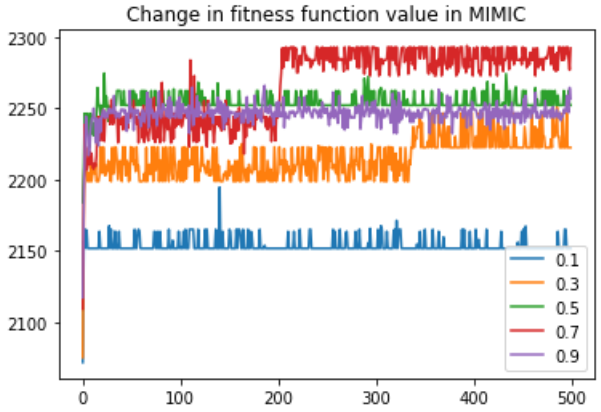
- No. of items = 50
- Max. number of duplicate copies per unique item = 3
- The maximum weight for a single element = 30
- The maximum volume for a single element = 40
- The volume of the knapsack = 2400
- Number of iteration = 5000
- I am going to compare both the time and performance for Randomized Hill Climbing, Genetic Algorithm, Simulated Annealing, and MIMIC on Knapsack Problem.

Table 3: Performance on Knapsack Problem

Algorithm	Best Obj Value	Time Taken (Seconds)
MIMIC	**2292**	**16.5**
Genetic Algorithm	2053	0.2295
Randomized Hill Climbing	2008	0.01385
Simulated Annealing	2006	0.01099



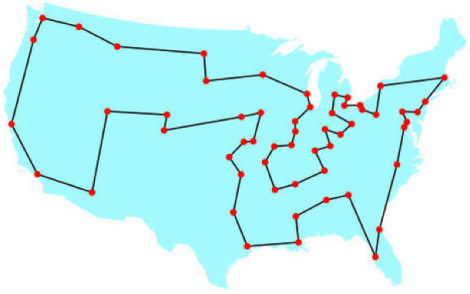
Overall MIMIC has much better performance early on since iteration 10. While others are stuck in 2000 range, MIMIC found good objective at 2200 and later broke through to almost 2300.
Why MIMIC outperformed significantly? Comparing to the TSP problem, the searching space of this problem is smaller, and it is easier to find the dependency relation of each item to the overall fitness. So dependency tree is a good and realistic approach to estimate the probability distribution for each pair of item.



I also compared between different starting probability between items in the dependency trees. As the chart suggests, p=0.7 is the best initialization. According to my research for smaller knapsack, an activation with p=0.9 can better. But given the number of objects is quite high, it is better to start lower that probability a little, which is more realistic to the context of this problem.
Why other algorithms underperformed? Genetic algorithm searches for combinations of bits with mutation, however, this approach might not satisfy all the constraints. With mutation, it might be able to fit the constraints but the performance would have gone down already. For SA and RHC, it works better in continuous value where as here the selection is in binary bits, it is difficult for the two algorithms to move along 'hills' since each solution is an point on its own, hence explaining the difficulty to escape local maximum by move around 'neighborhood' since there is none to begin with.
Time Performance: In terms of time performance MIMIC takes 71x more time than GA, 1191x more than RHC and 1650x more than SA. In this scenario, the time-performance trade-off is well worth it for MIMIC's super performance. As we have millions of objects, it might be difficult to afford the computational power to use MIMIC and instead we would adopt the next best, GA or RHC.

Traveling Salesman Problem

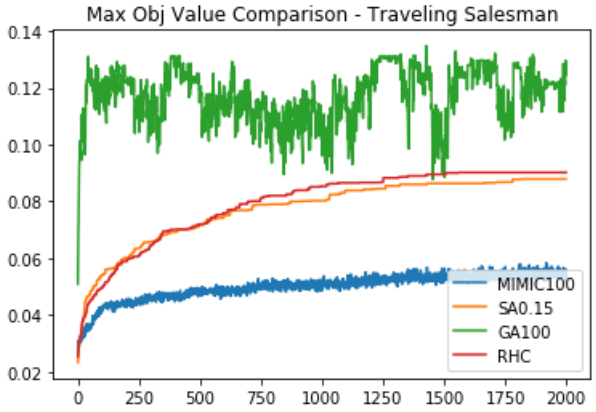
• TSP is known to be a NP-hard problem. This is a classic graph theory problem to find an optimal traversal route with minimum weights sum. It has edges representing distances from one city to another. It is not just finding the shortest path to go around all the city, it's also about finding a solution that would not go into or leaving the city twice.
• No. of stops = 80



• I am going to compare both the time and performance for Randomized Hill Climbing, Genetic Algorithm, Simulated Annealing, and MIMIC on Knapsack Problem.

Table 4: Performance on Traveling Salesman Prob.

Algorithm	Best Obj Value	Time Taken (Seconds)
MIMIC	0.0559	1566
Genetic Algorithm	**0.1250**	**5.665**
Randomized Hill Climbing	0.090	0.07524
Simulated Annealing	0.087	0.131



Overall Genetic Algorithm outperforms SA and RHC by 20-30%. Genetic algorithm found some traveling routes that's much better than other methods. Cost of a traveling route is related to the order of places it visited. The mating and mutation of GA helped optimize small local routes, thus more likely to get the global optimal result.
What about MIMIC? MIMIC in theory, remembers the structure of the problem. However, there are many permutation sequences to go from one city to another, which makes the dependency tree very complex, hence, it is improving in objective value, but very incrementally. I would think give n much more iterations, it would slowly achieve results similar to GA.
Time Performance Genetic algorithm takes 5 seconds comparing with 0.07 of RHC and 0.13 of SA while MIMIC takes 1566 seconds. The slight trade-off of time for performance is warranted for GA since the performance exceeds the others by 20-30