Assignment 2: Randomized Optimization

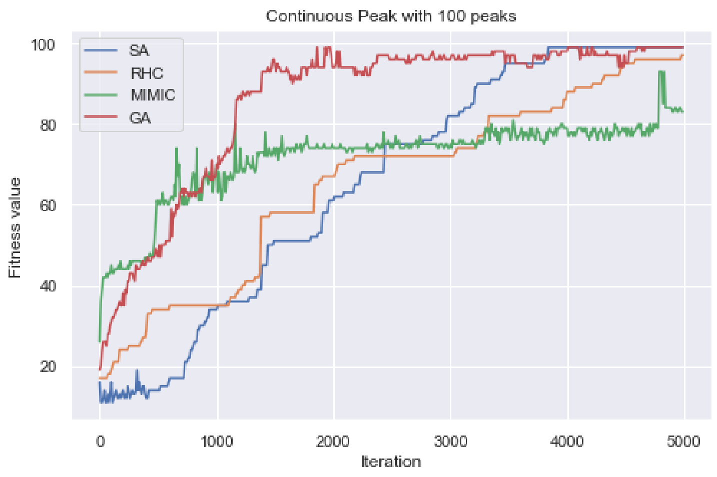
Part 1

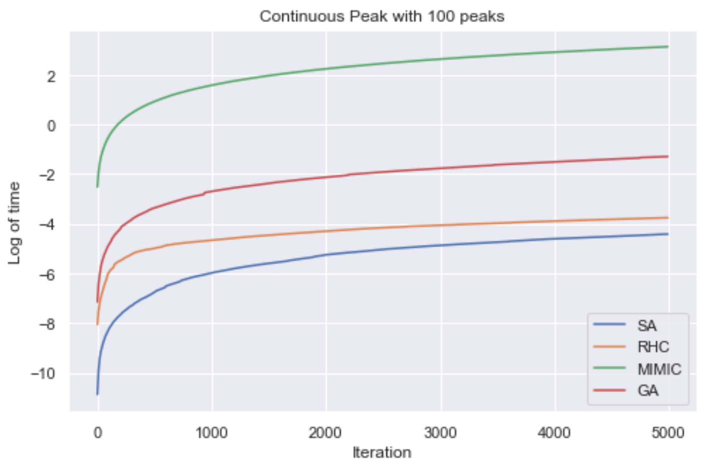
1. Introduction

For this assignment, we chose to implement random hill climbing, simulated annealing, genetic algorithm, and MIMIC algorithm for continuous peaks, flipflop, and knapsack problem. These three problems have three different type of structure. They are interesting because they show the performance differences of different algorithm at different types of problem.

2. Continuous peaks

In this problem basically the algorithm has to find the best picks in a one dimensional space. It is easy to visualize and also easy to observe the performance differences of different algorithms. It is simple but very effective in terms of algorithm comparison. That is why we chose this. We experimented with 100 peaks.

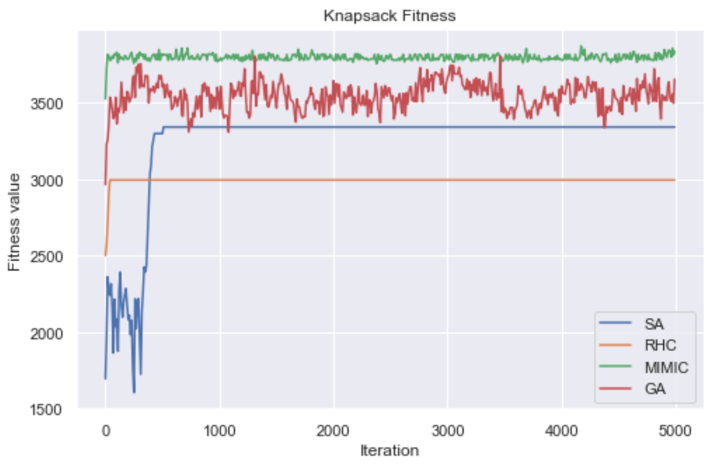


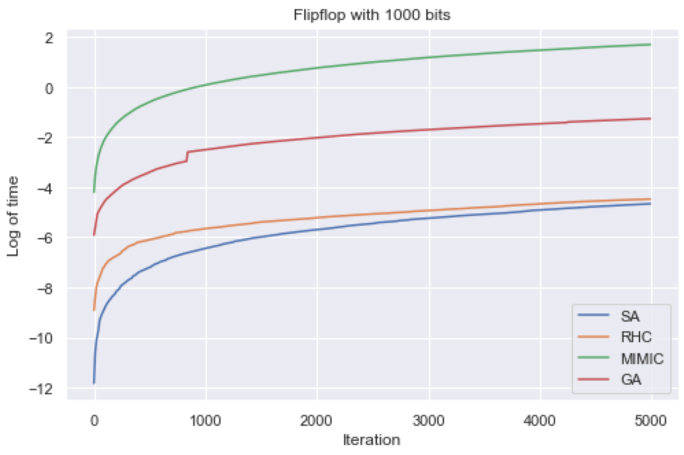


From the graph we can see that genetic algorithm performs the best. It plateaus around 1500 iterations and also reaches up to the value of 99. Simulated annealing also reaches 99 but after about 3800 iterations. MIMIC is outperformed by all the algorithm, because the overall structure of the hypothesis is very simple. So the MIMIC algorithm does not benefit much. Beside, due to bulk of extra calculation of probabilities at different iteration makes it slower than other algorithms. Genetic algorithm performs well because the sample most likely population is easy in this problem. However, it is has calculations at three steps: updating populations, cross-over, and mutations. Each of these steps adds extra time to the performance. On the other hand, Simulated annealing also performs well since it does not need much of exploration to find the optima. Time-wise thus simulated annealing is the fastest. Random hill climbing also reaches close to the max after 5000 iterations. If we increase the complexity of the problem by adding more peaks, these differences would be clearer. Due to lack of time we leave that for future experiment.

3. Knapsnack

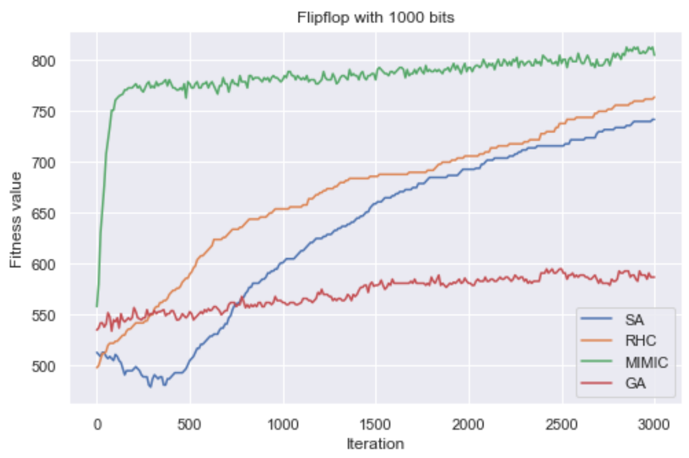
The knapsack is a hard non-deterministic polynomial optimizaiton problem. The idea is to how to make the best use of the space in the back pack. In this case we used 40 items, maximum volume 50 and maximum weight 50. Knowing the structure hypothesis space does have a benefit in this type of problem. That is why we can see mimic outperforms all the algorithm. However it takes longer to time to iterate over. It converges around 20 iterations. Genetic algorithm is another population based algorithm. It is also expected to perform well in this type of problem as it finds the most likely population. Instance based method like random hill climbing and simulated annealing are stuck at the local optima. As we can see in the fitness plot. Even with more iteration it won't improve due to the complexity of the problem.

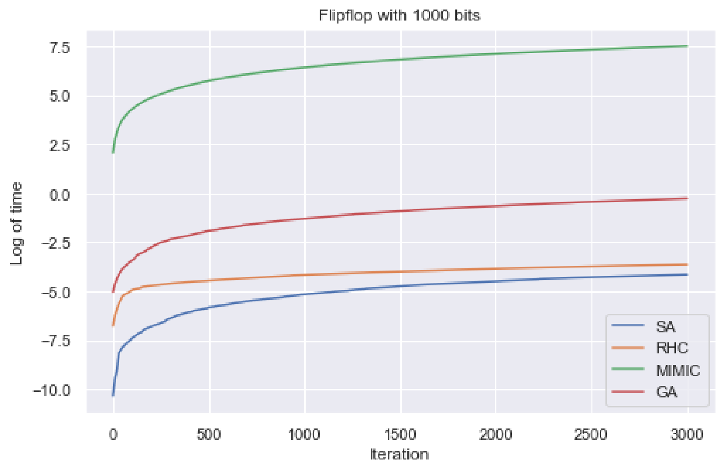




4. **Flipflop**

   Flipflop is more complex problem. Although genetic algorithm is a bit string type algorithm and expected to perform well in the Flipflop problem, it plateaus at very low fitness value.





MIMIC out performs other algorithms because this type of problems requires the understanding of the structure of the hypothesis. MIMIC thus has advantage as it uses the knowledge of previous iterations. However as usual, this added complexity requires extra time. We can see MIMIc takes highest amount of time. However it plateaus at about 500 iterations. Simulated annealing is an instance based method. It would take more iterations to find the best neighbors. Because of the same reason random hill climbing does not converge. Even after 3000 iteration both simulated annealing and random hill climbing are increasing in fitness. Genetic algorithm would also perform well if we increase the population and cross over number from 100, and 50 to some higher number.

Part 2

1. 'Adult' Dataset from UCI

Using some continuous and categorical data of a person such as age, education, race, job title etc., predict if a person's annual income is more than $50,000 or not.

To build the corresponding classifier, we used "adult" data set (Blake and Merz 1998) from the UCI repository. (source: http://archive.ics.uci.edu/ml/datasets/Adult). [1]

Provided training and test datasets have 32561 and 16281 instances respectively. After removing the unknown and duplicate data, we have 30162 training instances and 15060 test instances. Both sets are properly labeled and have 14 predictor variables of categorical and numeric types. One of the interesting features of the dataset is that it is not well balanced. In both training and test dataset about 25% instances belong to the class of people with higher income (>50k) and about 75% are in the other class. However, the datasets are already shuffled for experiments.

2. Neural Network Architecture

We used an artificial neural network with two densely connected hidden layers with 100 nodes at each. There are 14 predictor variables at the input layer and output layer just outputs either 1 for income greater than $50000 or 0 for less income. We used ABAGAIL library to code the algorithm at focus.

3. Implementations

Random Hill Climbing

Random hill climbing is a simple instance-based method. It starts at a random point as the current state. Then it compares the states of its neighboring instances. If the neighboring instances optimizes the function better, it makes an increment to that point as the current state. If no better optimization state found in the neighboring instances, it stops and mark it as the local maximum. We used the ABAGAIL lib

Simulated Annealing

It is a probabilistic method of finding the best solution. First, we randomly select an instance at a certain temperature T. Then we sample a neighboring instance. If the neighboring instance optimizes the function better, we move the current state to the neighboring instance. If the neighboring state is worse than the current state, we use the probability of acceptance to determine whether to move to that instance. Mathematically, Probability of acceptance = exp((f(x_neighbor) – f(x_current))/T) . Then at every iteration, we decrease the temperature by a cooling factor (T = CE * T) and repeat the process for a fixed number of iteration. In out experiments, we tried T values from 1e9, 1e4 and 10 to observe the behavior at different temperature. We also experimented with Cooling exponent with values 0.15, 0.5, and 0.99.

Genetic Algorithm

Genetic Algorithms initially computes the fitness of all the individuals from a population of solutions. Then the most fitted individuals are selected as the parents for the crossover step. At the crossover step, the best individuals are paired up as parents and a random crossover point is selected to produce offspring solutions. Finally, at the mutation step, some of these offspring are randomly mutated to maintain diversity. This process is repeated until it reaches the convergence. Due to the time constraint we only experimented with the population size 50, crossover samples 40, 20, and 10 and mutations 20, and 10.

MIMIC

Mutual-Information-Maximizing Input Clustering (MIMIC) algorithm uses the knowledge of cost function at each iteration and a probability density estimate to find the best structure of the solution space, which in turn guides to the search to the optima.

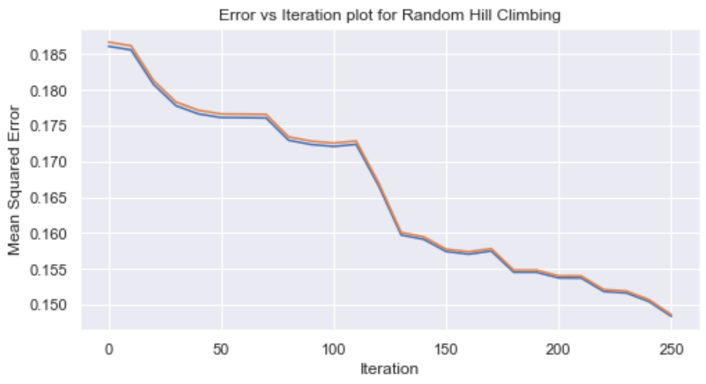
4. Analysis:

A brief summary of the analysis:

Algoritms	Train Accuracy(%)	Test Accuracy(%)	Time (on average per iteration)
Backpropagation (Assignment 1)	78.72	78.70	~18
Random Hill Climbing	75.18	75.58	15.82
Simulated Annealing	75.36	75.76	15.64
Genetic Algorithm	75.85	76.24	~24

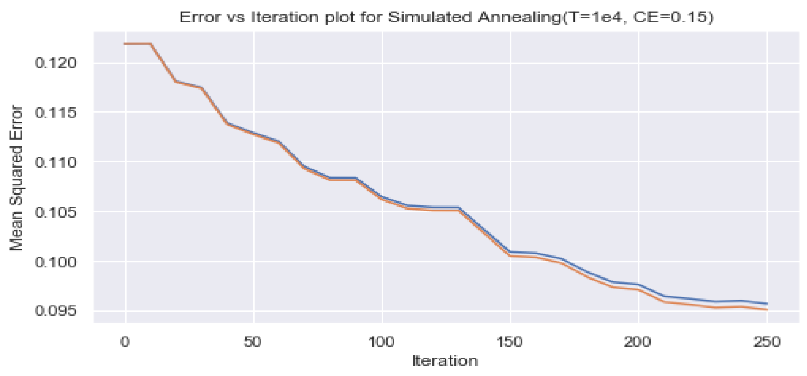
Random hill climbing:

1. We have experimented with different number of iterations and ran the experiments multiple times as well. As we can see from the graph that the error continues to decrease. Since the data set is comparably big, and the number of nodes in each layer is quite a lot, the iteration time was long too. Eventually we had to limit the experiments to 250 iterations.
2. If we allow more iterations, it would apparently improve the performance of the algorithm. The improvement however is very slow.
3. At some experiments the accuracy was about 25% for both test and training dataset. It is because, the random start was unlucky. Random hill climbing starts from a point and incrementally improves the performance. Since the starting point was at a low accuracy and improvement was very slow, even after 250 iterations, the accuracy reached to about 25%.

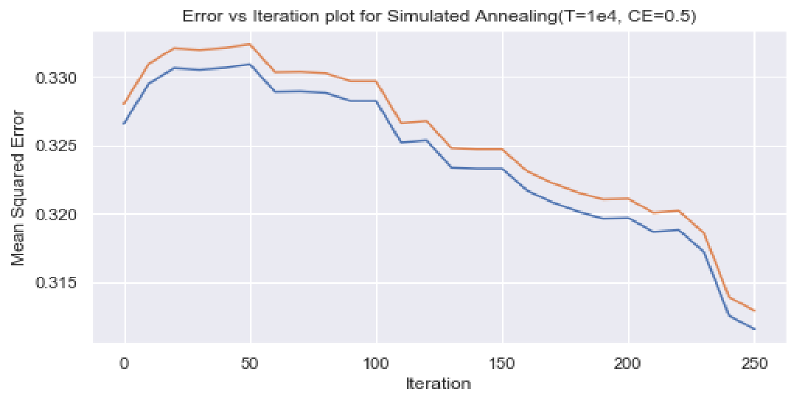


Simulated Annealing

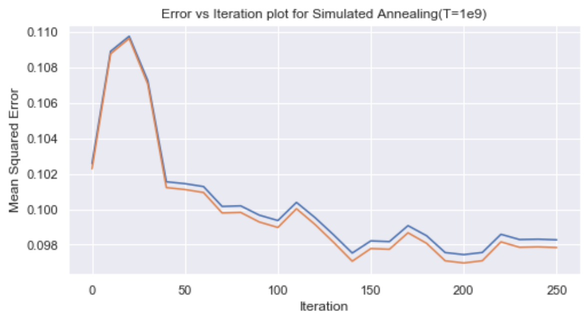
1. At moderate temperature (T =1e4) with fast cooling(0.15), the algorithm behaves more like Random hill climbing. As we can see the error graph which is very similar to the random hill climbing. With decreasing temperature, it explores less and exploits more. As a result it does not considers the worse states. The curve as we see only going downward. Around 250 iterations it starts to plateau.

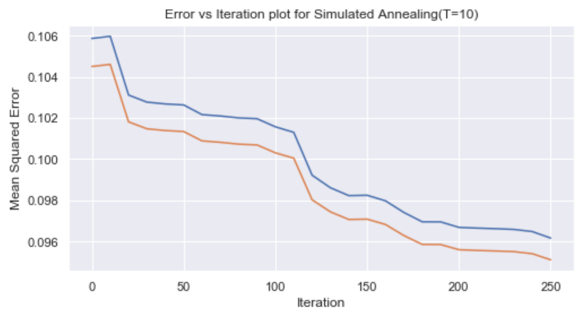


2. If the temperature decrease rate lowers, we can see the curve at first explores and finally at low temperature it only exploits and goes towards better options only.



3. We also experimented with higher and lower temperatures to observe the effect. However, with descent decreasing factor and higher temperature it seems to converge faster. Due to the exploration, it has more chances to find the optima and less chance to get stuck. Eventually, with decreasing temperatures, it converges to the optima found through exploration.

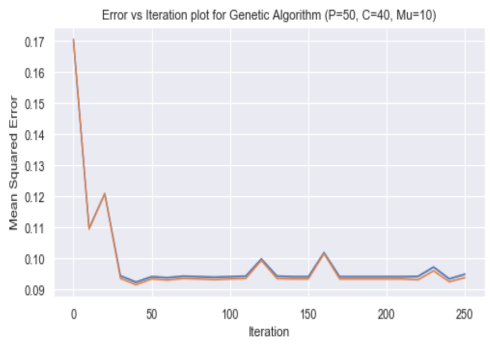


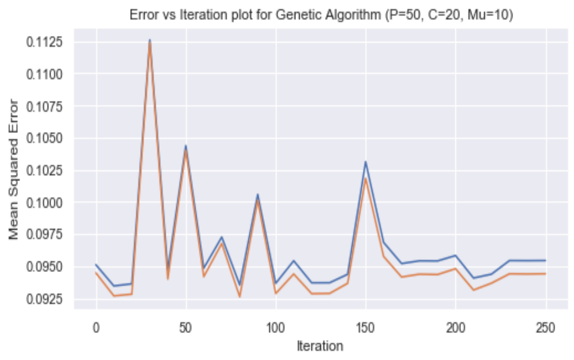


4. Accuracy score for test and train dataset is better than the hill climbing algorithm (as presented in the table). Time-wise it took almost same time per iteration as random hill climbing. However, it converges with less iteration than random hill climbing. It may not be the case all the time, since the hill climbing depends on the random start. If it starts near the optima, it may converge faster.

Genetic Algorithm

1. As we can see in the table that genetic algorithm has better accuracy than random hill climbing and simulated annealing. Even though we only experimented with a low sample population of 50. With increased population, it may perform as good as backpropagation with gradient descent in this problem.
2. One of the reasons behind the better performance is that it starts with a sample population instead of just one instance. As a result, bigger the population size, the better it performs. With smaller population size, there is a chance that it may never reach the optima.
3. How many instances would pair up in the crossover step also influences the performance. We found with higher number of parents in crossover steps results in better accuracy and faster convergence. As we can see in the graph with crossover instances 40 performs better than that with 20.





4. It performs slower than other algorithms due the higher number of computations at each iteration. At each iteration it goes through, updating population, performing crossover for each pair, and performing mutation for certain number of instances. For population sample of 50, crossover of 40 instances and mutation of 10 takes longer per iteration than population sample of 50, crossover of 20 and mutation of 10. For random hill climbing, and simulated annealing, it only considers one neighbor.

Conclusion:

From the analysis we can see that there is no free lunch theorem is true. There is no such algorithm that will universally perform well. With complexity and structure of the hypothesis space algorithms performs differently.

However with more time and iteration it is possible to get better results but the amount of time it would take is not reasonable. With improved coding like parallel programming, this problem may be solved.

References

[1] : http://archive.ics.uci.edu/ml/datasets/Adult

[2] https://github.com/pushkar/ABAGAIL [3] https://gist.github.com/mosdragon/ad893f877a631260e3e8

[3] http://www.dudonwai.com/docs/gt-omscs-cs7641-a2.pdf?pdf=gt-omscs-cs7641-a2