Assignment 2: Randomized Optimization

In this analysis, I will explore 3 different optimization problems with 4 different search techniques: randomized hill climbing, simulated annealing, genetic algorithm, and MIMIC. The 3 optimization problems I studied are traveling salesman problem, flip-flop problem, and knapsack.

This report will be in 2 parts. In the first part, I will study the implementation of the 4 search techniques, and analyze their behaviors under different optimization problems. By experimenting on parameters in each search algorithm, I can find the best parameter for the algorithms, then compare 4 algorithms for each optimization problem.

In the second part, I will apply three search algorithms to a neural network problem. The network problem I picked is the Speech Emotion Recognition problem from assignment 1. I will first use the Backpropagation algorithm from ABAGAIL library to test the error rate of different neural network structures and pick the best structure for further test. Then, I will test the parameters for each randomized optimization algorithm (randomized hill climbing, simulated annealing, genetic algorithm), find the best parameter setting, and then compare the result with backpropagation result.

Part 1 : Randomized Search Algorithms & Optimization Problems

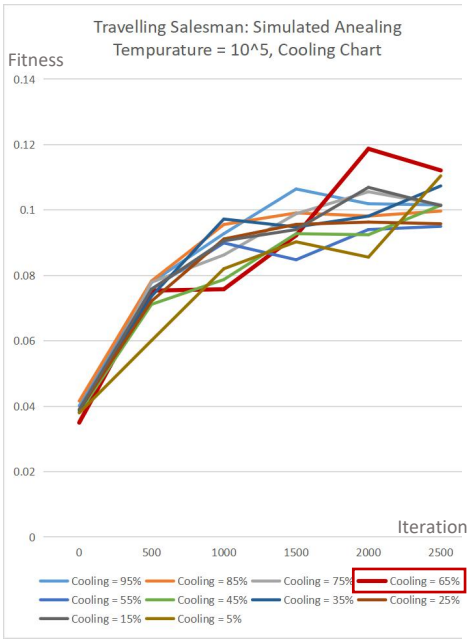
1. Traveling Salesman

The traveling salesman problem (TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" The problem was first formulated in 1930 and is one of the most intensively studied problems in optimization. It is used as a benchmark for many optimization methods.

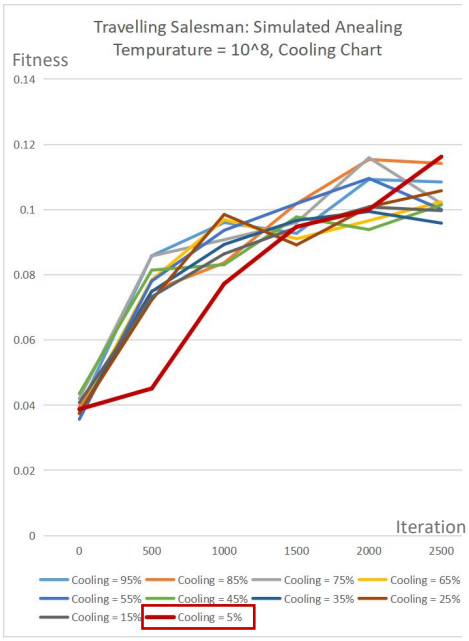
(1) Simulated Annealing

The name of the algorithm comes from annealing in metallurgy, a technique involving heating and controlled cooling of a material to alter its physical properties. When the temperature(T) is high, the search has more energy to travel to random places even though the starting point may not have better fitness than the current one. When the temperature(T) is low, the search tends to be stable, it's unwilling to take adventures on lower fitness places. Thus, the temperature(T) and the cooling speed of temperature are very important attributes in this process.

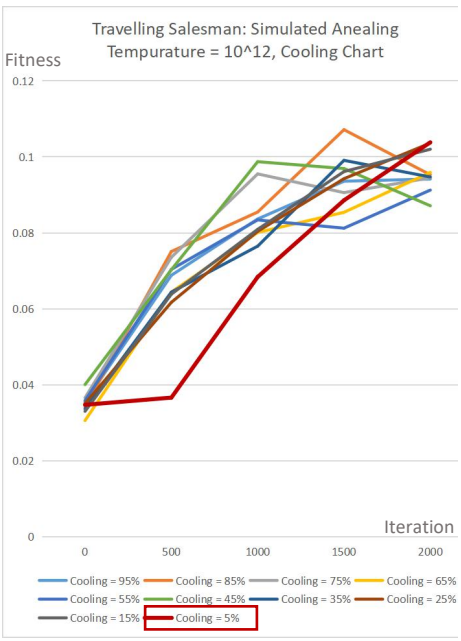
The first study is on the cooling parameter. I set 10 cooling effects as 5%, 15%, 25%,…, 95%. For example, cooling = 5% means T = T*(1-0.05), each iteration the temperature is down by 5%. The iteration goes from 1 to 2500, to save some experimenting time, I set the intervals to be 500. For the starting temperature, I set it to be 10^5 at first, then confused by the result, I tried another two values 10^8 and 10^12. Below is the charts I got.



T = 10^5
Best Cooling = 65%



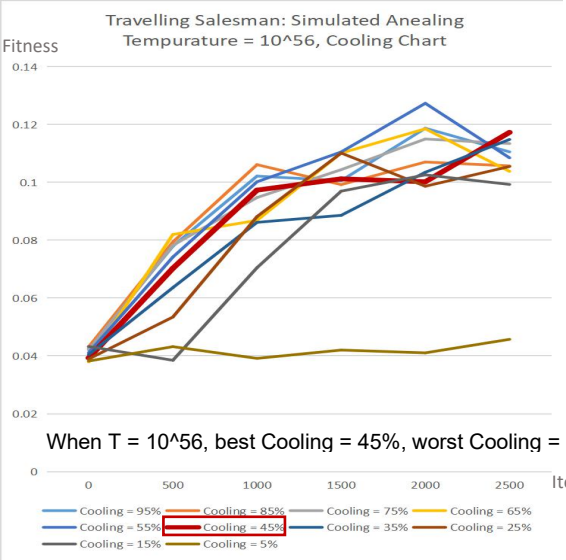
T = 10^8
Best Cooling = 5%



T = 10^12
Best Cooling = 5%

For the above result, my assumption is that when T is small, some cooling effects don't have a chance to lower the temperature to a stable state. I want all the cooling processes to be able to lower the temperature to near 0. So I did a counting for the slowest cooling(5%). I get the approximate starting T = 10^56:

$$10^{56} * (1-5\%)^{2500} = 2.037$$

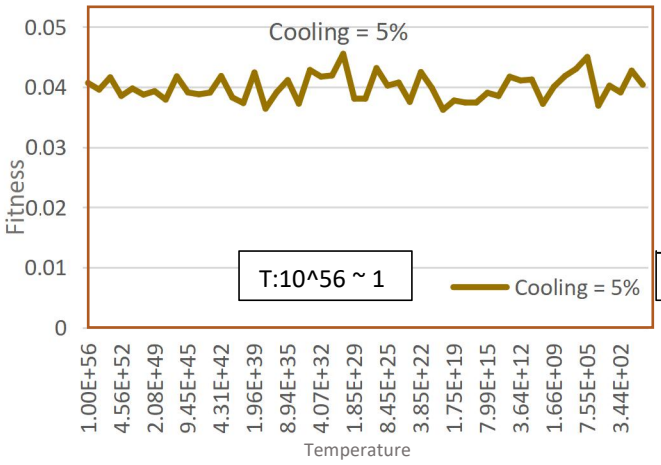


When T = 10^56, best Cooling = 45%, worst Cooling = 5%

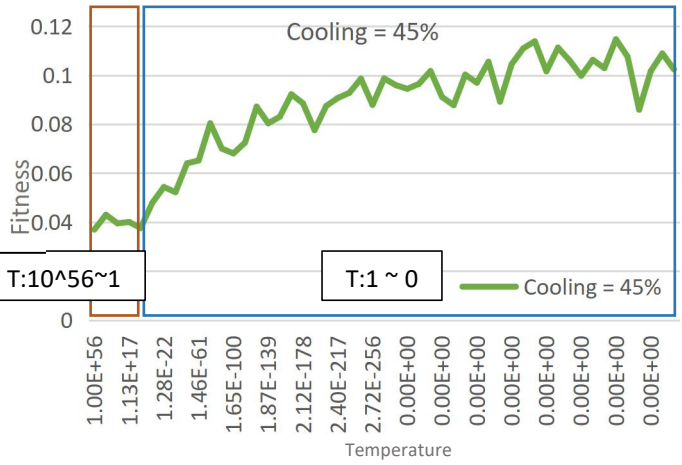
I experimented again with T = 10^56, and I got the chart on the left. The chart shows the best cooling is 45% at iteration 2500, the 5% cooling is the worst.

This made me want to explore the relationship between temperature and fitness evaluation.

So I kept the same parameters, but changed the iteration to temperature, set iteration interval to 50, and compared 45% cooling and 5% cooling. The results are shown below.



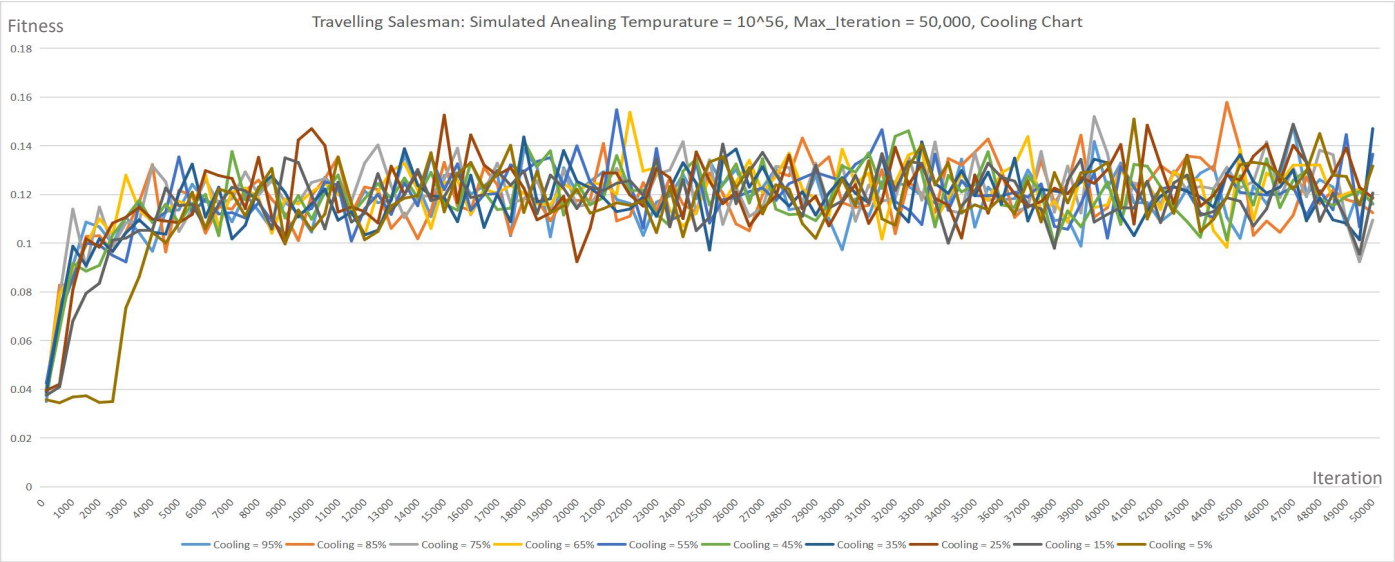
T:10^56~1



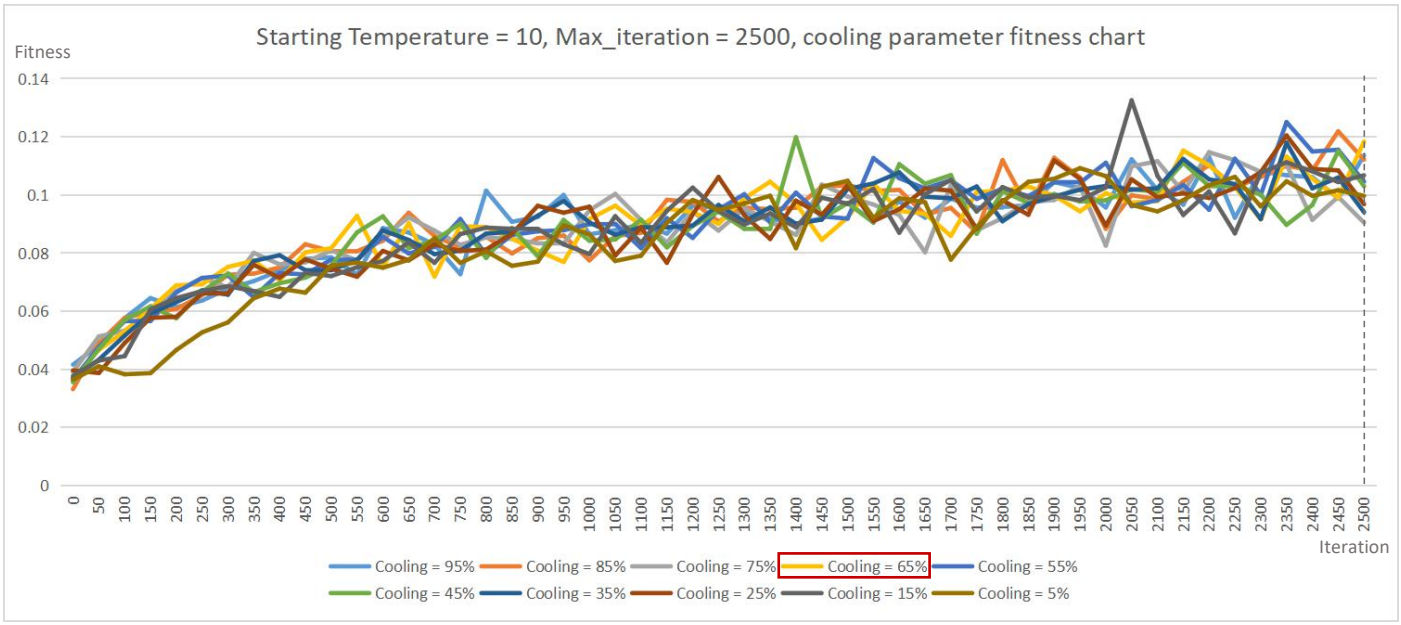
The charts say when the temperature is between 10^56 and somewhere near 1, both 5% and 45% cooling curve doesn't do well. But when the temperature goes from 1 to somewhere very near 0, the performance improves.

This makes me wonder if increasing iterations and letting the 5% curve cool down more will better the performance.

So the same experiment, but set maximum iteration to 50000. The result shows with more iteration(more cooling down between 1 and 0) the performance improves.



I also did another experiment, set start temperature = 10, and max_iteration = 2500, get the result below. From the charts, when all cooling effects can reduce the temperature to somewhere very close to 0, the fitness increases, and when T is reduced between 1 and 0, the fitness increases more.



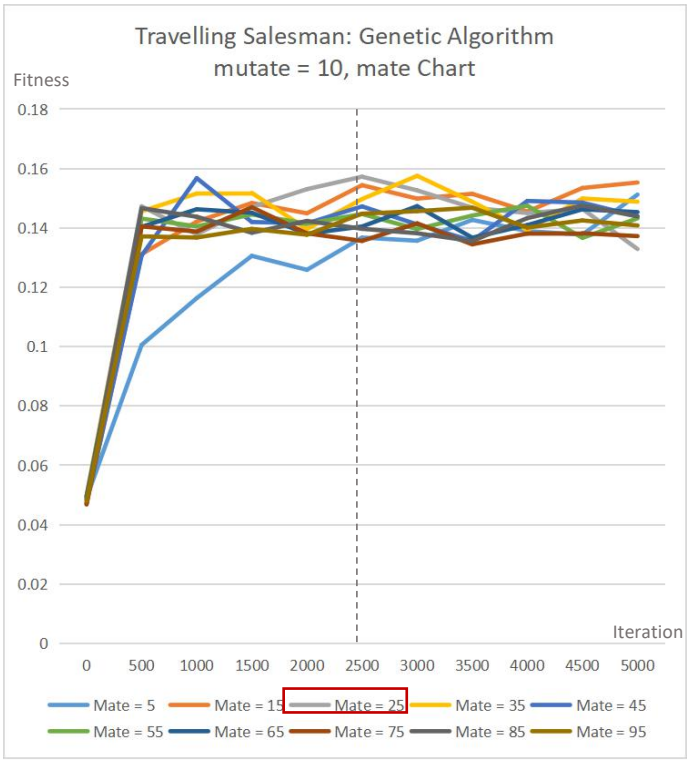
When T = 10, best cooling at iteration 2500 is 65%(T = T*0.35)

(2) Genetic Algorithm

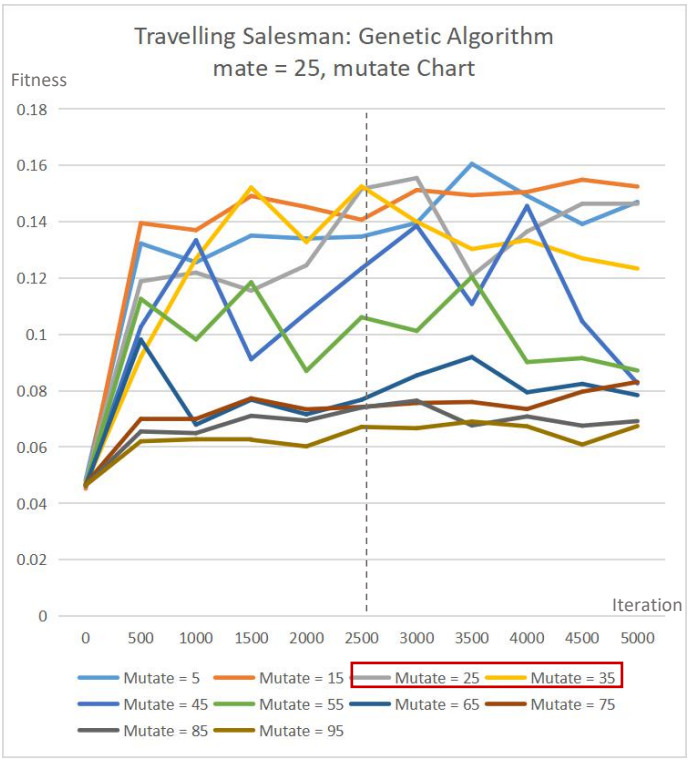
Genetic algorithm (GA) is inspired by the process of natural selection and is commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover, and selection.

Hence, I first did 2 experiments focused on how mate ratio and mutation ratio influence the result. In the left chart below, I set the population size to be 100, mutation size to be 10, and test on different mate sizes (5, 15, 25,…, 95). The result shows when the mutation ratio is 10%, for this particular problem at 2500 times iteration, the 25% mate rate can achieve the highest fitness value.

The second experiment shows in the right chart below. When the population size = 100, mate rate is 25%, the best mutation rate at 2500 times iteration is 25% and 35%.

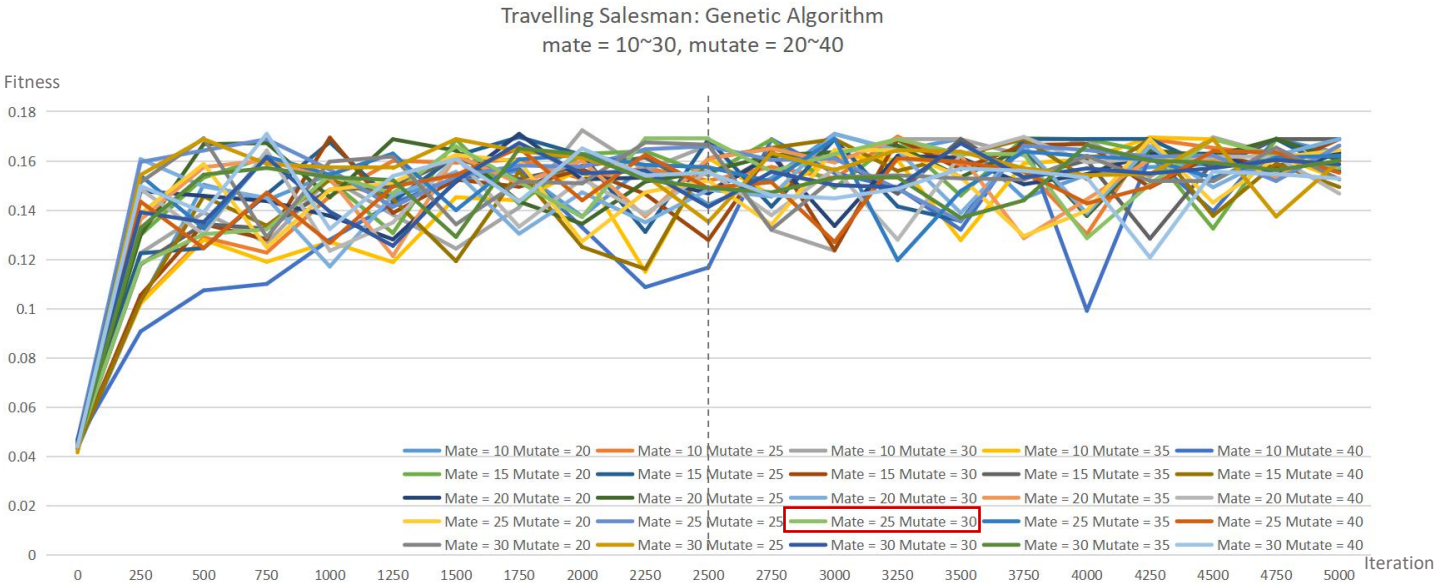


When mutate = 10, best mate = 25



When mate = 25, best mate = 25 or 35

The experiments above are limited to certain mutation or mate rate. Below I tested a combination of different mutation and mate rates(Mate = 10,15,20,25,30; Mutate = 20,25,30,35,40), and got the chart below:



Overall, a lot of combinations achieved pretty good fitness values. For iteration times at 2500, the best performance parameters are Mate = 25, mutate = 30.

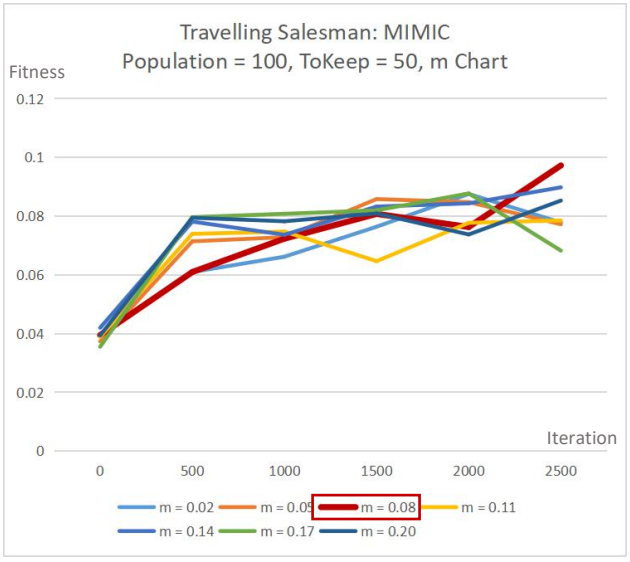
(3) MIMIC

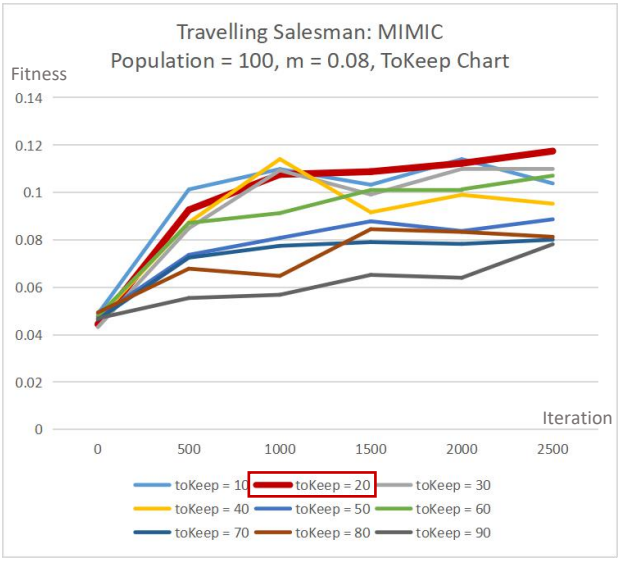
MIMIC is an algorithm that can covey the model structure and refine the model by probability distribution. The MIMIC algorithm begins by generating a random population of candidates chosen uniformly from the input space, from this population the median fitness is extracted and is denoted θ0. The algorithm then proceeds:

1. Update the parameters of the density estimators of Pθi(x) from a sample
2. Generate more samples from the distribution Pθi(x)
3. Set θi+1 equal to the Nth percentile of the data. Retain only the points less than θi+1.

Experiment 1: Observing the ABAGAIL library, I set the initial input space size to be 100 (same as the genetic algorithm). Another important parameter is the small positive number m to add when making a dependency tree. The left chart below is the experiment of m from 0.02 to 0.20, each time increment by 0.03.

Experiment 2: For each iteration, MIMIC will choose the Nth percentile and keep the samples that have fitness value above that. So the second experiment is to set how many samples to keep for the next population. The toKeep parameter is set from 10 to 90, increment by 10.





From the result, at 2500 times iteration, the best performance parameter is m = 0.08, toKeep = 20.

(4) Comparison

Obtained the best-performance parameters for each algorithm, I compared the four algorithms together. The experiment setting:

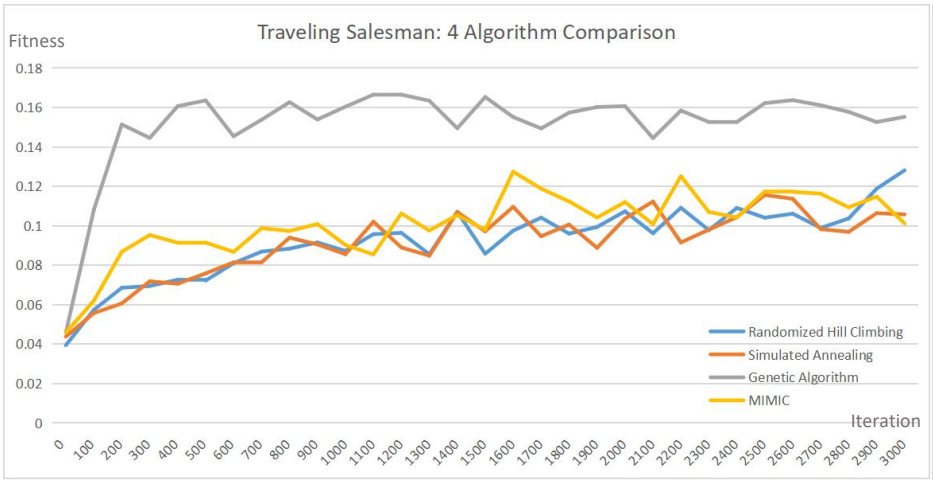
Test iteration from 0 to 3000, increment 100

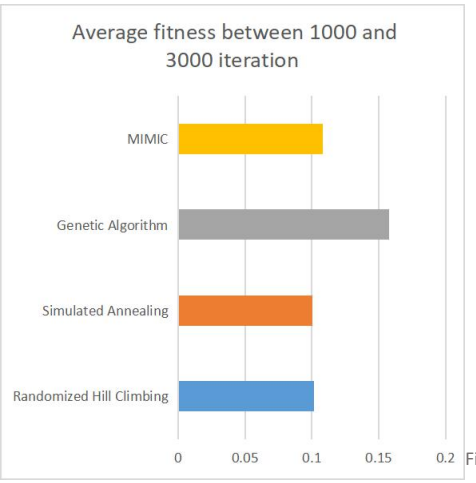
Each iteration:

*Simulated Annealing: starting temperature=10, cooling 65%(T = T*0.35)*

Genetic Algorithm: population size=100, mate=25, mutate=30

MIMIC: sample size=100, toKeep=20, m=0.08





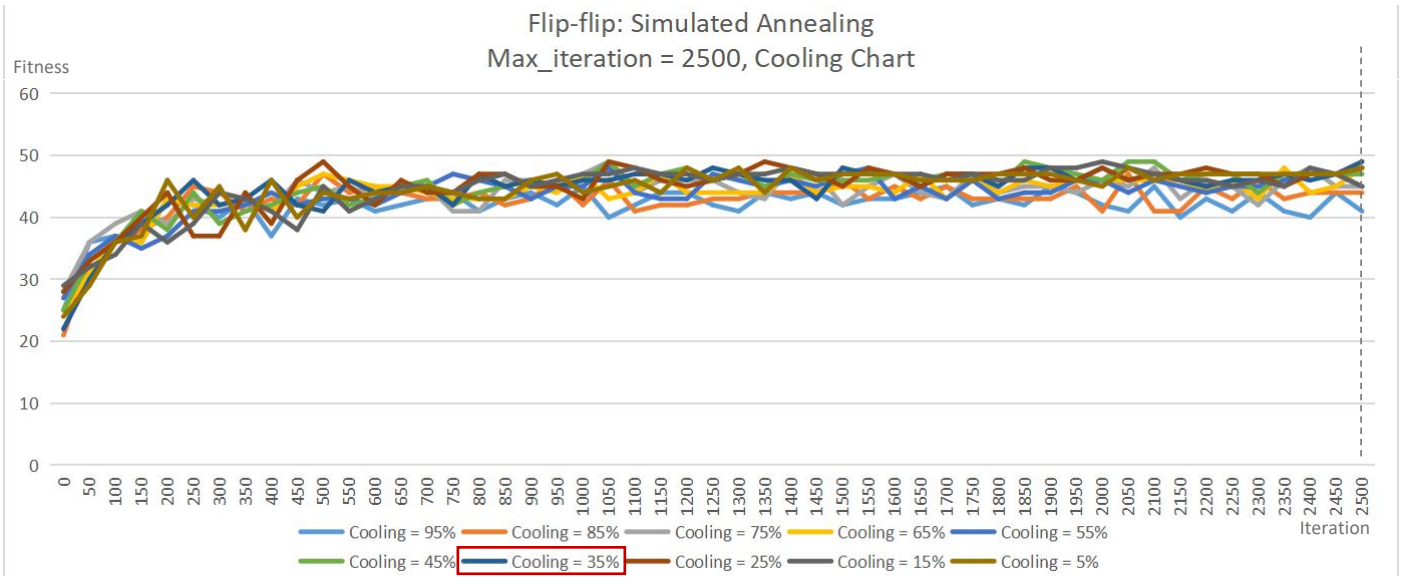
The result of the experiment is shown in the charts above. The left chart shows the fitness values change for each algorithm along with the increase of iteration. The Genetic Algorithm curve increases pretty fast and achieves the highest fitness. The right chart is the average fitness between 1000 and 3000 iterations. The performance is Genetic Algorithm > MIMIC > Randomized Hill Climbing > Simulated Annealing.

2. Flip-flop optimization

A flip flop is an electronic circuit with two stable states that can be used to store binary data. Flip-flop is a problem that counts the number of times bits alternation in a bit string.

(1) Simulated Annealing

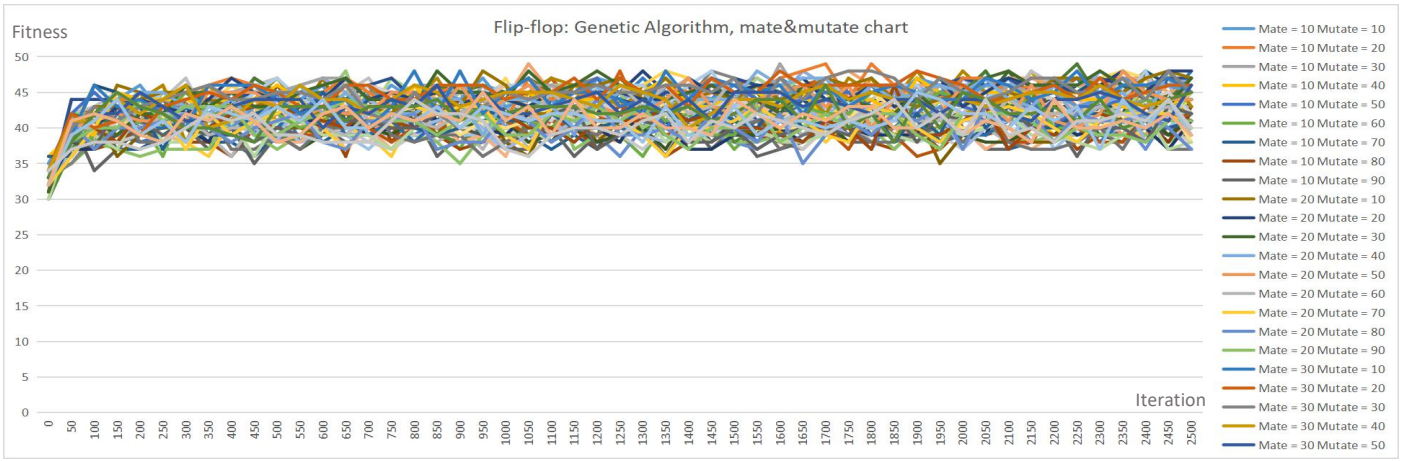
Based on the experience of the simulated annealing on traveling salesman problem, I set the iteration to be 2500, starting temperature to be 100, and test 10 different cooling parameters as before. The result is shown below.

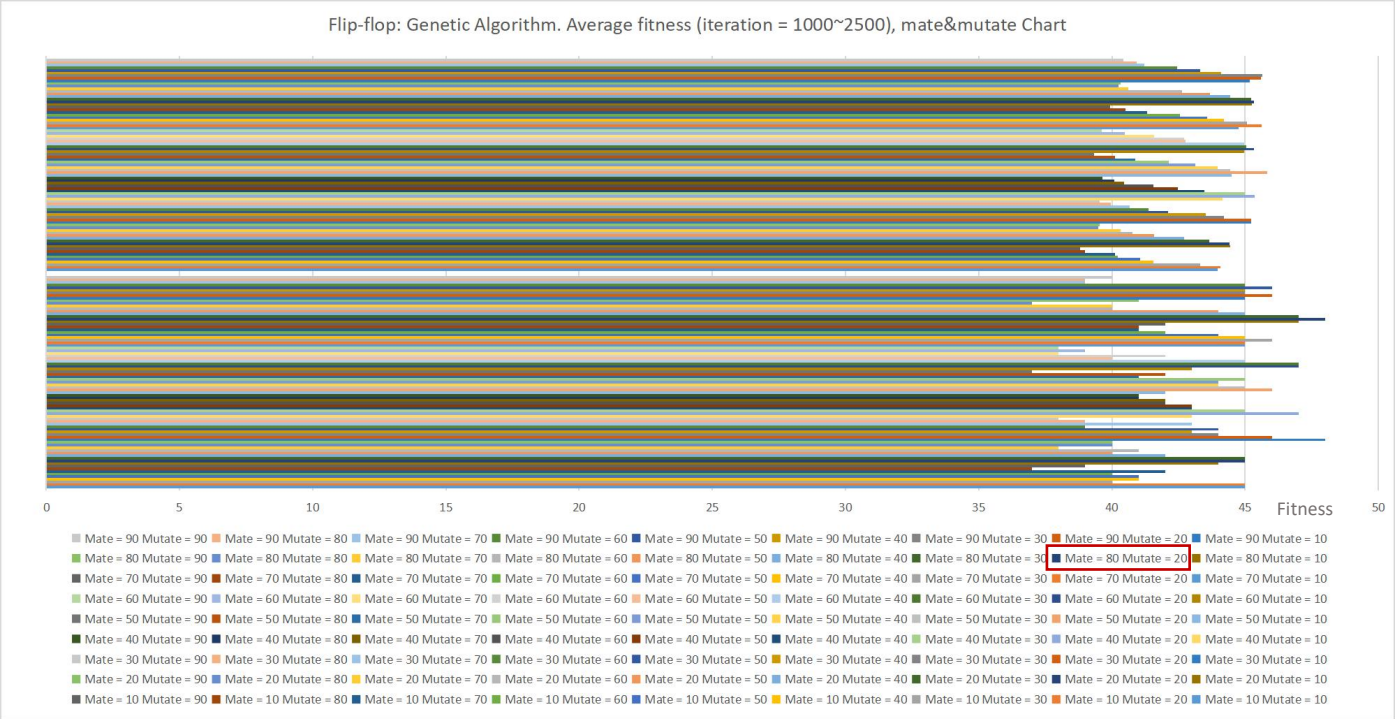


Best parameter at iteration = 2500 and starting temperature = 100: Cooling = 35% (T = T*0.65).

(2) Genetic Algorithm

As in the previous optimization problem, I tested a combination of mate parameter and mutate parameter(Mate = 10, 20, 30,…, 90. Mutate = 10, 20, 30,…, 90). The result is shown below.



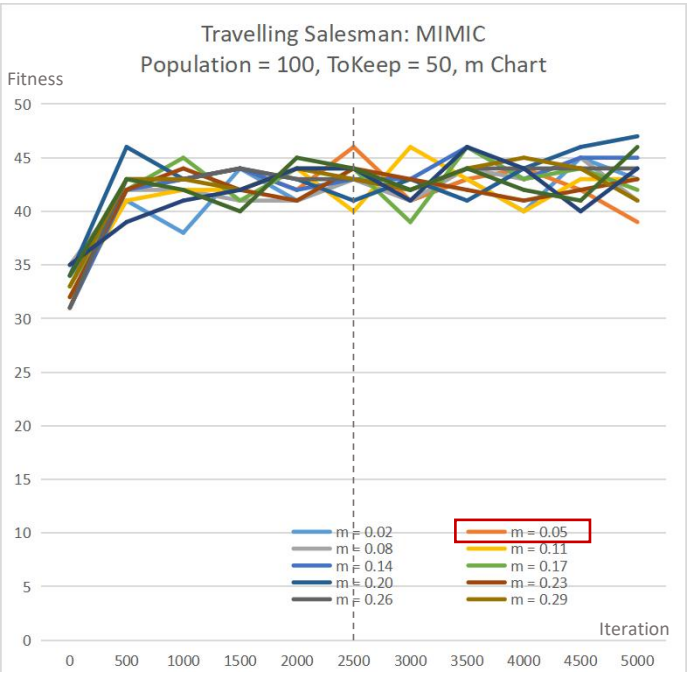


To visualize the result better, I count the average fitness of each combination at iteration 1000~2500. The bar chart shows the best parameters are Mate = 80, Mutate = 20.

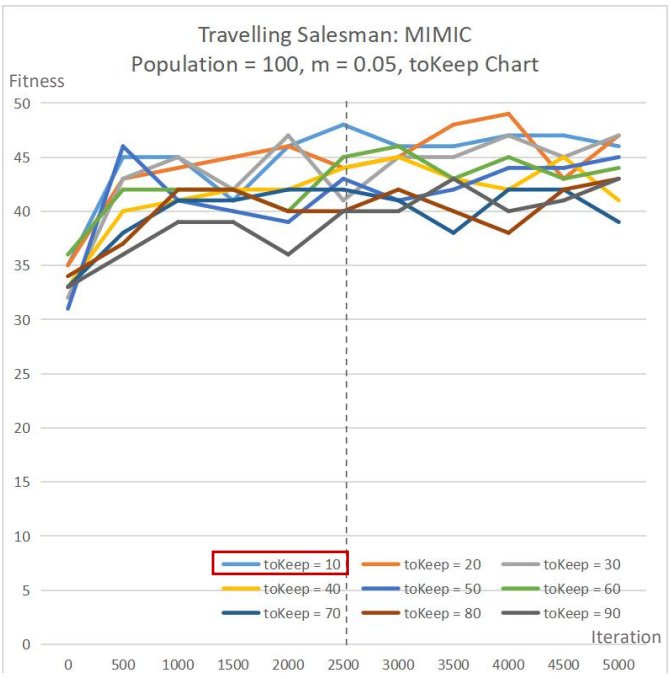
(3) MIMIC

As in the previous optimization problem, I set the population size to be 100 and tested the positive number m to add when making dependency trees, and the toKeep parameter which determines how many samples get selected to join the next population.

In the first experiment, toKeep set to 50, and test different m values(0.02, 0.05, 0.08,…0.29). In the second experiment, the m value was set to 0.05, and test different toKeep values(10, 20, 30 …90). The results are shown below.



Best parameter when iteration = 2500, toKeep = 50:
m = 0.05



Best parameter when iteration = 2500, m = 0.05:
toKeep = 10

(4) Comparison

Obtained the best-performance parameters for each algorithm, I compared the four algorithms together. The experiment setting:

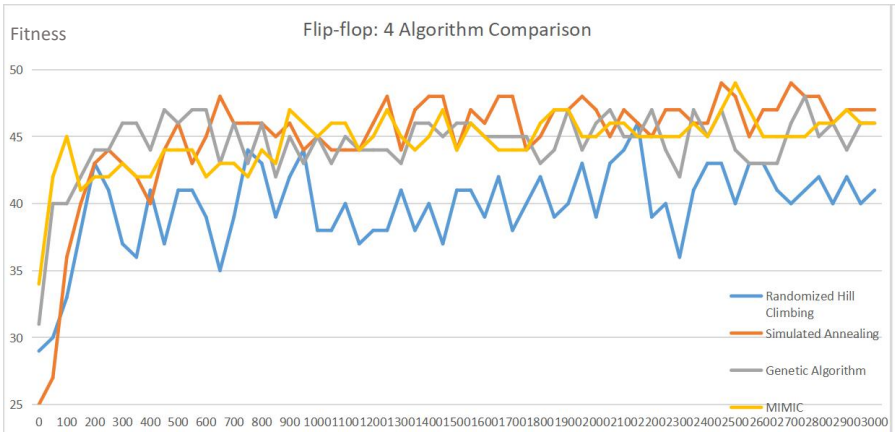
Test iteration from 0 to 3000, increment 100

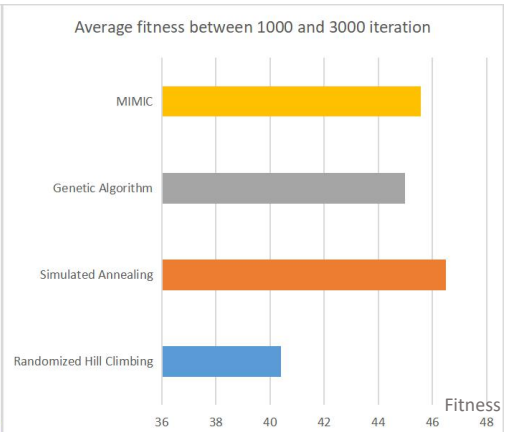
Each iteration:

*Simulated Annealing: starting temperature=100, cooling 35%(T = T*0.65)*

Genetic Algorithm: population size=100, mate=80, mutate=20

MIMIC: sample size=100, toKeep=10, m=0.05





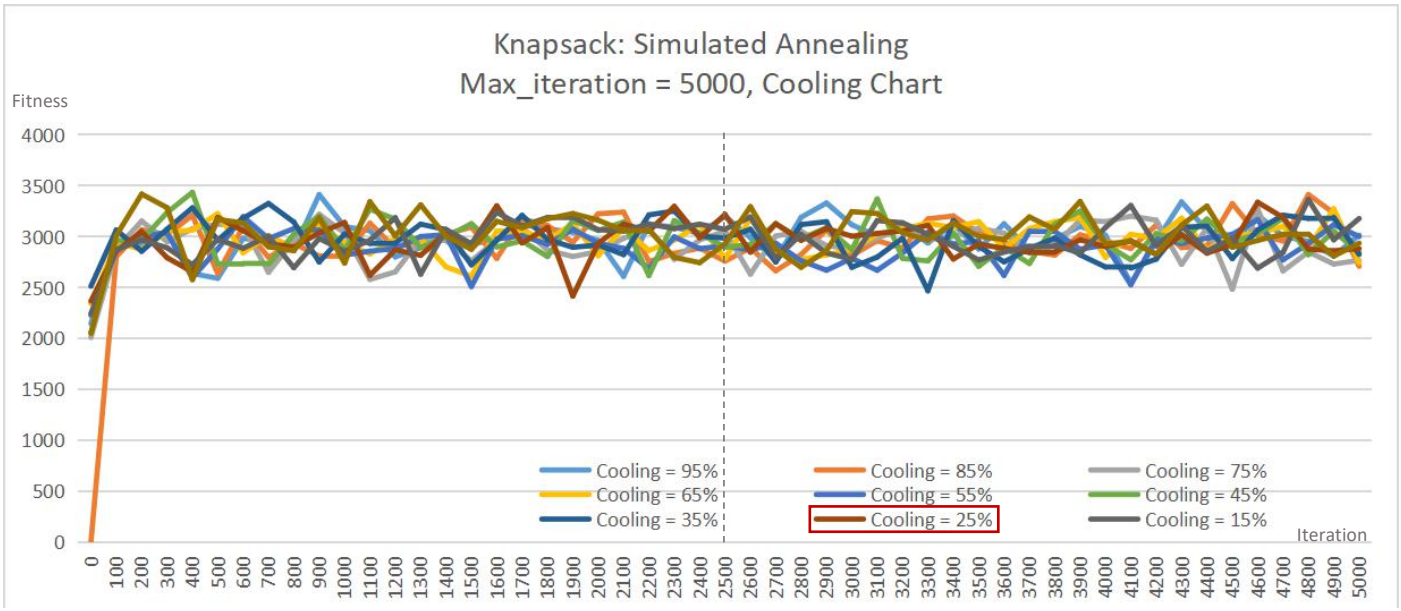
Based on the chart, the Simulated Annealing has the best average fitness value, followed by MIMIC, Genetic Algorithm, the randomized hill climbing is the lowest.

3. Knapsack optimization

The knapsack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item included in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items. The problem often arises in resource allocation where the decision-makers have to choose from a set of non-divisible projects or tasks under a fixed budget or time constraint, respectively.

(1) Simulated Annealing

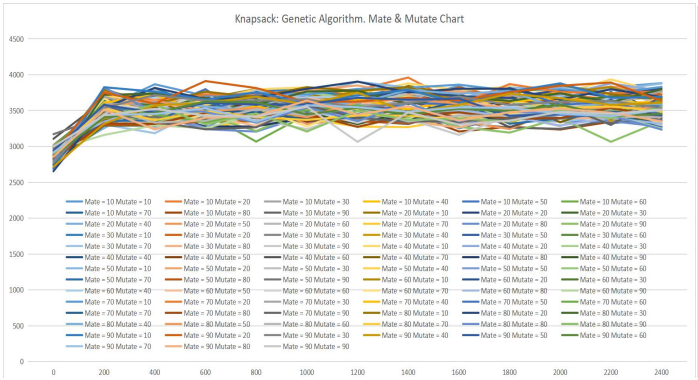
As other optimization problems, for simulated annealing, I tested different cooling parameter. The starting temperature is set to 10, and iteration from 0 to 5000, increment by 100. The result is shown below.

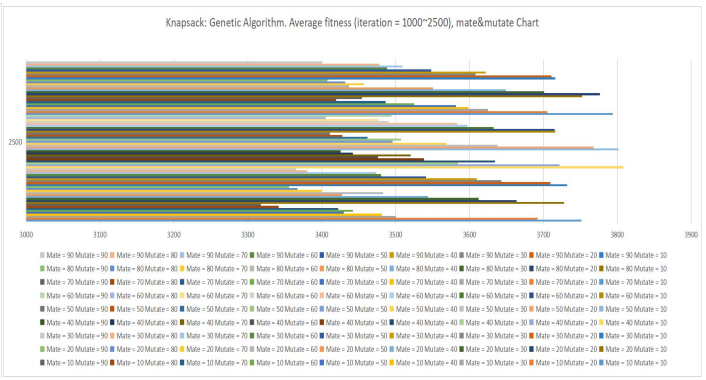


The result shows at 2500 iteration, the best cooling parameter is 25%(T = T*0.75).

(2) Genetic Algorithm

As in the previous optimization problem, for simulated annealing, I tested different cooling parameters. The starting temperature is set to 10, and iteration from 0 to 5000, increment by 100. The result is shown below.



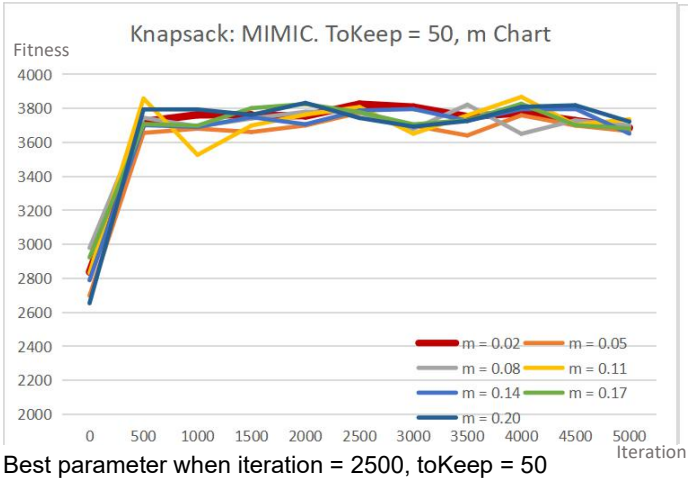


From the charts we can see the best combination for genetic algorithm is Mate = 40, mutate = 10.

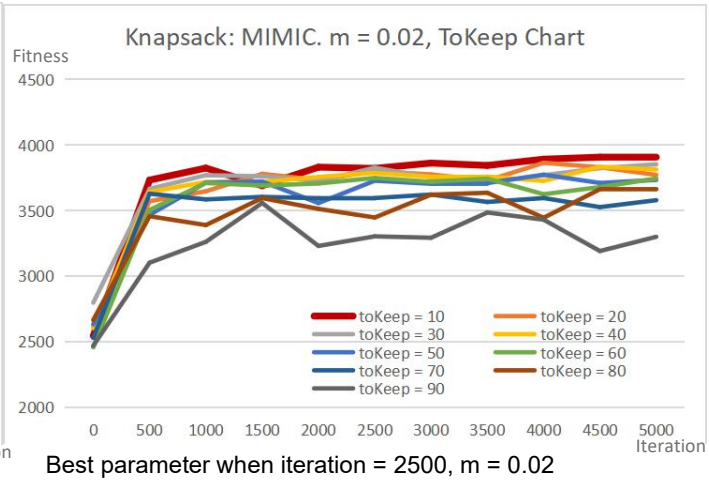
(3) MIMIC

As in the previous optimization problem, I set the population size to be 100 and tested the positive number m to add when making dependency trees, and the toKeep parameter which determines how many samples get selected to join the next population.

In the first experiment, toKeep set to 50, and tested different m values(0.02, 0.05, 0.08,…0.20).
In the second experiment, the m value was set to 0.05, and tested different toKeep values(10, 20, 30 …90).
The results are shown below..



Best parameter when iteration = 2500, toKeep = 50
m = 0.02

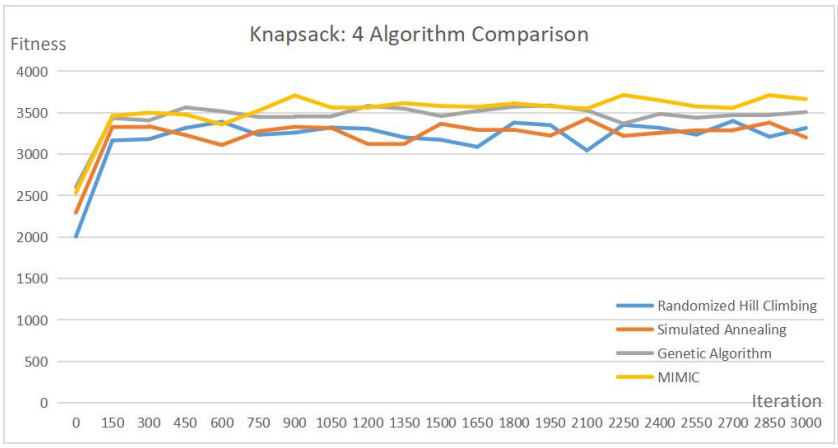


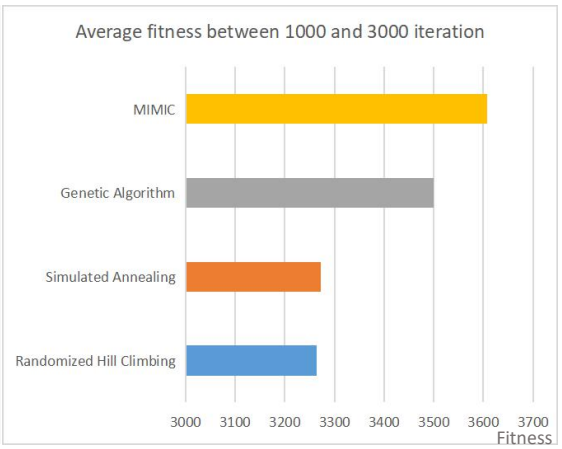
Best parameter when iteration = 2500, m = 0.02
toKeep = 10

(4) Comparison

Like for other optimization problems, I did a comparison among the 4 search algorithms to evaluate their performance. Below is the test setting:

Test iteration from 0 to 3000, increment 150
Each iteration:
*Simulated Annealing: starting temperature=100, cooling 25%(T = T*0.75)*
Genetic Algorithm: population size=100, mate=40, mutate=10
MIMIC: sample size=100, toKeep=10, m=0.02





The chart above is the performance(Fitness) curve of the four algorithms within 3000 times iteration. At the first, all 4 algorithms increased their performance as iteration increased. At around 150 iterations, the four algorithms increasing slows down. We can see the MIMIC curve is above the other algorithms. From the chart at right, the overall performance is MIMIC > Genetic Algorithm > Simulated Annealing > Randomized Hill Climbing.

Part 2 : Optimization Algorithms & Neural Network

In Part 2, I choose one of the classification problems from assignment 1, which is Speech Emotion Recognition. I first experimented with different neural network structures on the problem and selected the most suitable structure to apply the optimization algorithms to. The consideration is both performance and time. Then I tried different parameter combinations for each algorithm and find the ones that achieve the best performance. Finally, I apply the chosen parameters to the algorithms and compare their performance with the backpropagation algorithm.

Speech Emotion Recognition Problem

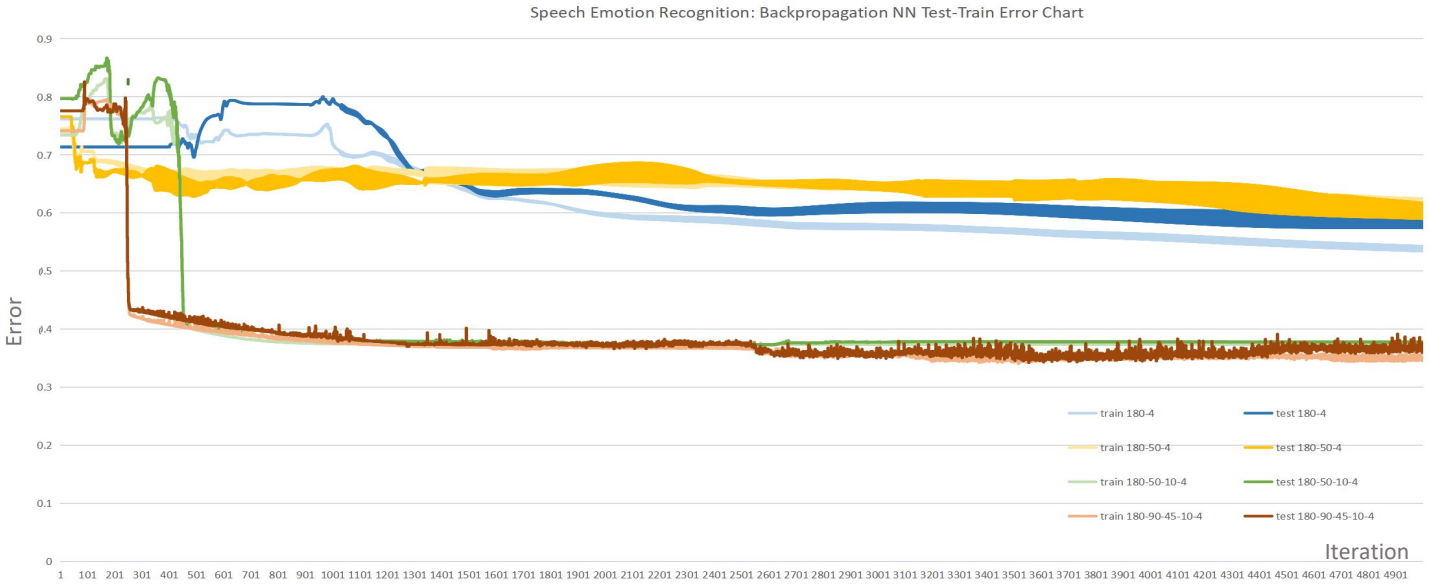
Speech Emotion Recognition is a task of recognizing human emotion from speech regardless of the speech contents. We have been always wanting to achieve natural communication between humans and machines just like the communication between

humans. After years of research, we are able to convert human speech into a sequence of words. However, despite the great progress made in speech recognition, we are still far from having a natural interaction between man and machine because the machine does not understand the emotional state of the speaker. That's why I want to pick the problem of speech emotion recognition(SER). It's important to let the machine have enough intelligence to understand human emotion by voice.

1. Find the most suitable network structure by backpropagation

To apply optimization algorithms to the learning problem, my design is to use the 3 algorithms to optimize the input weight for each unit in the neural network. So the first step is to find the most suitable network structure.

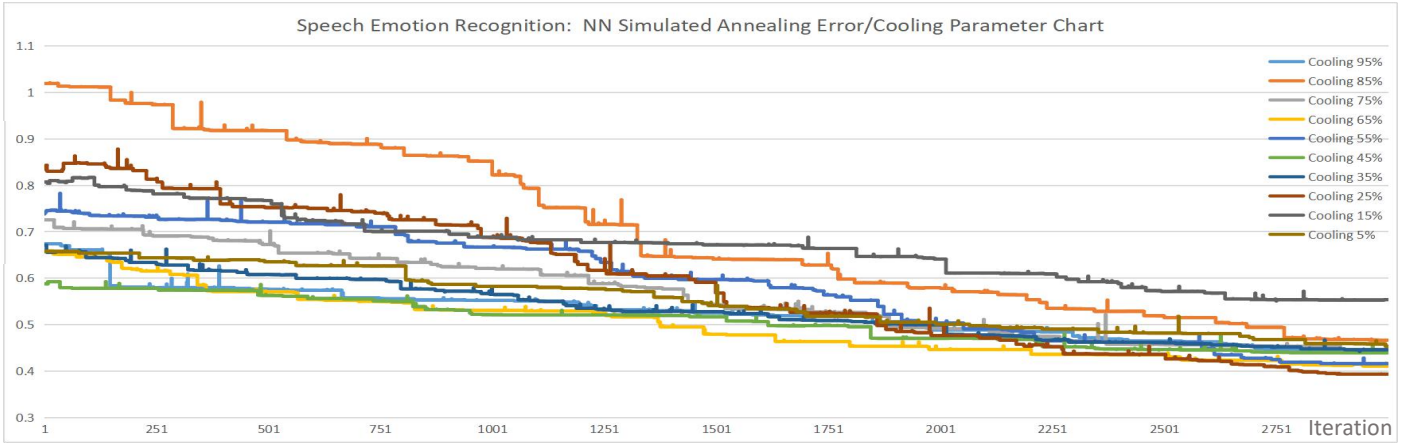
I applied the backpropagation algorithm to the problem and tested a few different structures. Based on the experience in assignment 1. I pick the 180-4, 180-50-4, 180-50-10-4, 180-90-45-10-4 structures to conduct the test. (e.g. 180-50-4 means 180 input units, 4 output units, 1 hidden layer of 50 units). The experiment result is shown below.



The chart above shows the training and testing error curve of 4 network structures. We can easily tell the best performance one is 180-90-45-10-4, the second-best structure is 180-50-10-4 which is really to the best performance one. As you can imagine, the more hidden layers the structure has, the more computing time is needed. Considering both time and performance, I choose the 180-50-10-4 network structure to perform the following experiments, which has 180 input units, 4 output units, and 2 hidden layers.

2. Simulated Annealing Parameter Choosing

Based on the network structure, I applied simulated annealing algorithm to find the best weight for input at each unit. I set the starting temperature to be 10000, and tested different cooling parameters(cooling 95%, 85%, 75%, …, 5%). The result is shown below.



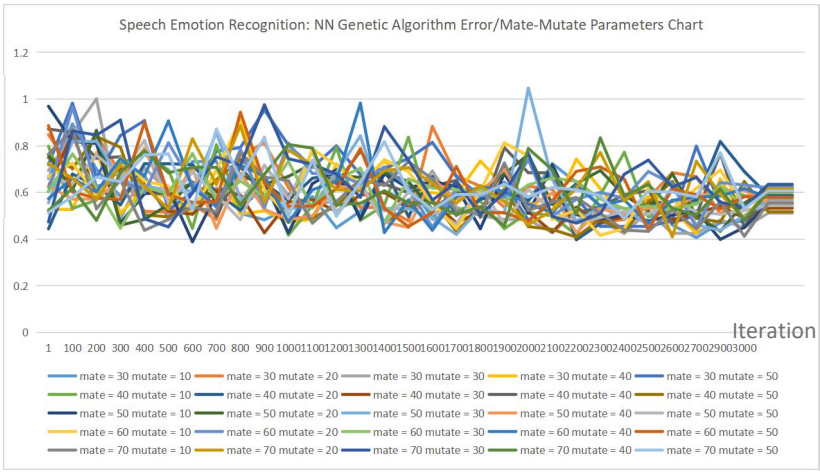
Based on the chart, we can find that Cooling = 25% curve reached the lowest error when iteration is approaching 3000. Thus, Cooling 25% (T = T*0.75) is the best parameter when the starting temperature is 10000.

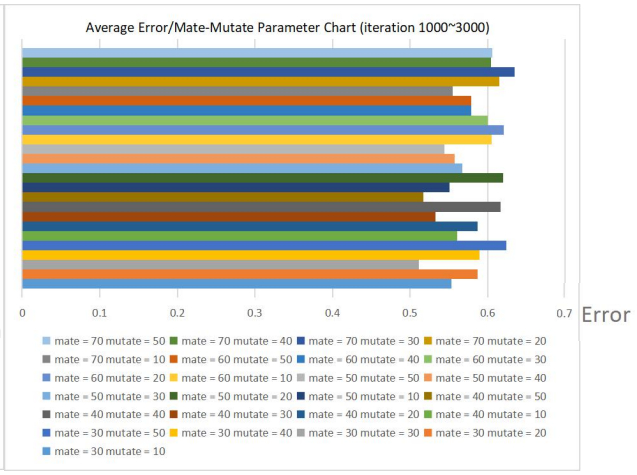
3. Genetic Algorithm Parameter Choosing

In the genetic algorithm, the most important parameter is the mutation rate and mate rate. As always, I set the population size to be 100, and test a lot of combinations of mate and mutate size. (Mate = 30, 40, 50, 60, 70. Mutate = 10, 20, 30, 40, 50)

Because the genetic algorithm is costly in time, I set the iteration interval to be 100, which means only test when max iteration is 1, 100, 200…3000, so the resulting curve is not very smooth. To read the result better, I calculated the average error between 1000

iterations and 3000 iterations, and get the chart on the right side.





From the charts, we can tell the parameter combination of ”mate = 30, mutate = 30” has the lowest error. Thus it’s the best parameter for the genetic algorithm.

4. Four algorithms comparison

Finally, I compared the backpropagation algorithm, simulated annealing, genetic algorithm, and randomized hill climbing together on the 180-50-10-4 neural network. The experiment setting is:

Test error of iteration from 0 to 3000:

Backpropagation;

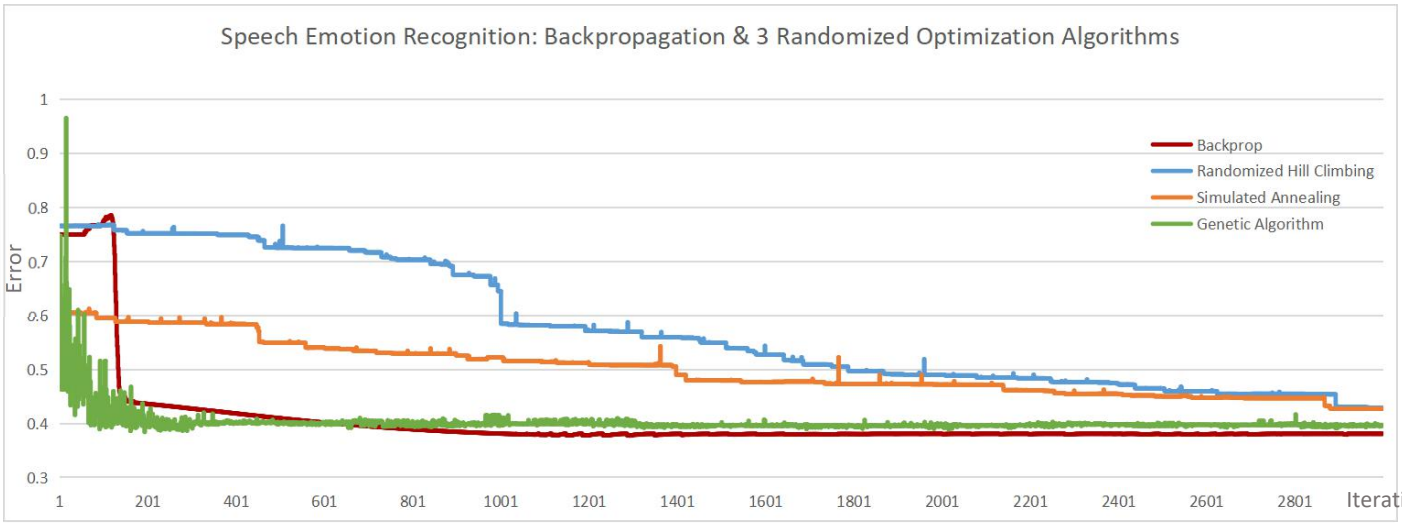
Randomize Hill Climbing

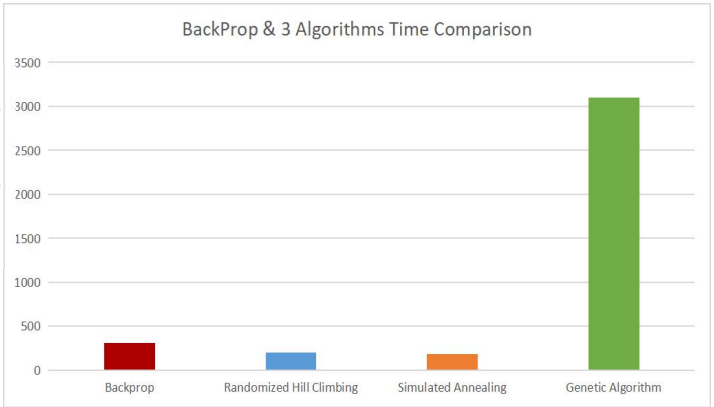
*Simulated Annealing: starting temperature=100, cooling 25%(T = T*0.75);*

Genetic Algorithm: population size=100, mate=30, mutate=30;

Calculate the start and end time of each algorithm.

The results are shown in the charts below.





Overall, the backpropagation algorithm reaches the lowest error rate among all four algorithms, but genetic algorithm curve is very close to it. The worst performed algorithm is randomized hill climbing. The backpropagation and genetic algorithm’s error curve dropped fast between 200 iterations, and the other two algorithms’ curves dropped slowly. Before reaching the 600 iterations, the genetic algorithm has a lower error than the backpropagation. Genetic algorithm can find the optimized weight faster than the other 3 algorithms.

For the time comparison, the genetic algorithm takes the most time to compute, it may be due to the fitness evaluation and crossover operation each time.