Assignment #1: Simulated Annealing and Sampling

Nazyrov Amir

Github:

https://github.com/sentenced210/Deep_Learning/tree/master/assigment_1

Part 1:

For running simulated annealing on Iris Dataset I create simple neural network which consist of 4 layers:

Input layer: 4 neurons
1st hidden layer: 6 neurons
2nd hidden layer: 4 neurons
Output layer: 3 neurons

This structure of the neural network is the most optimal for my algorithm, the greater number of runs converge to 97% accuracy. I tried to substitute a different number of neurons, but of their large number, the algorithm converge accuracy to 60-75%.

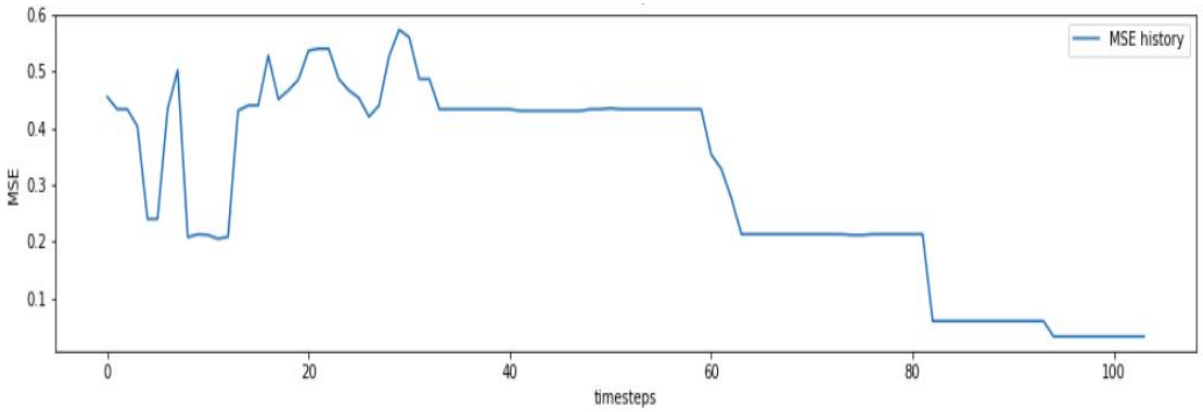
Also I want say that the algorithm doesn't always converge to a good result, often it gets stuck in the local minimum and can't get out of there

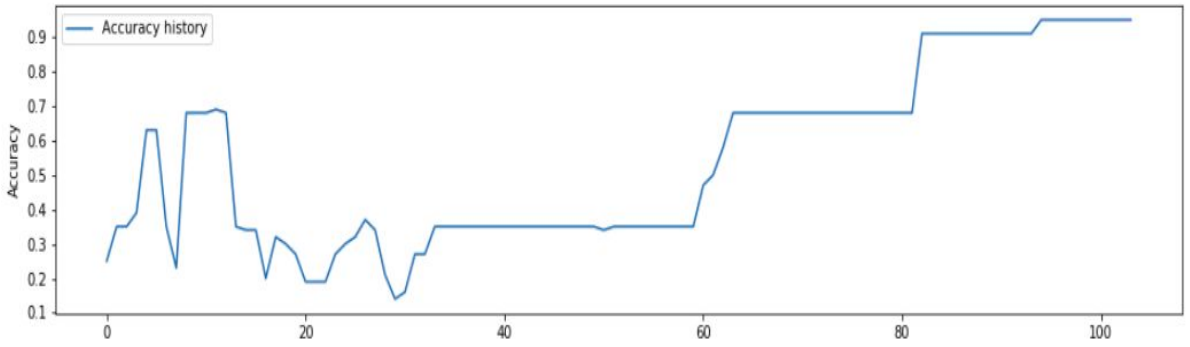
Results of simulated annealing on different annealing rates (T = 2, T min = 0.00001):

Annealing rate	Total steps	Total time (seconds)	Training loss (MSE) (end)	Training accuracy (end)	Validation loss (MSE) (end)	Validation accuracy (end)
0.9 (Fast)	103	10,76	0,03	0.95	0.16	0.76
0.99 (Medium)	1118	89.33	0.02	0.97	0.06	0.9
0.999 (Slow)*	5296	459,92	0.18	0.61	0.25	0.46
Using gradient descent	100 epochs	3.7	0.009	0.99	0.03	0.94

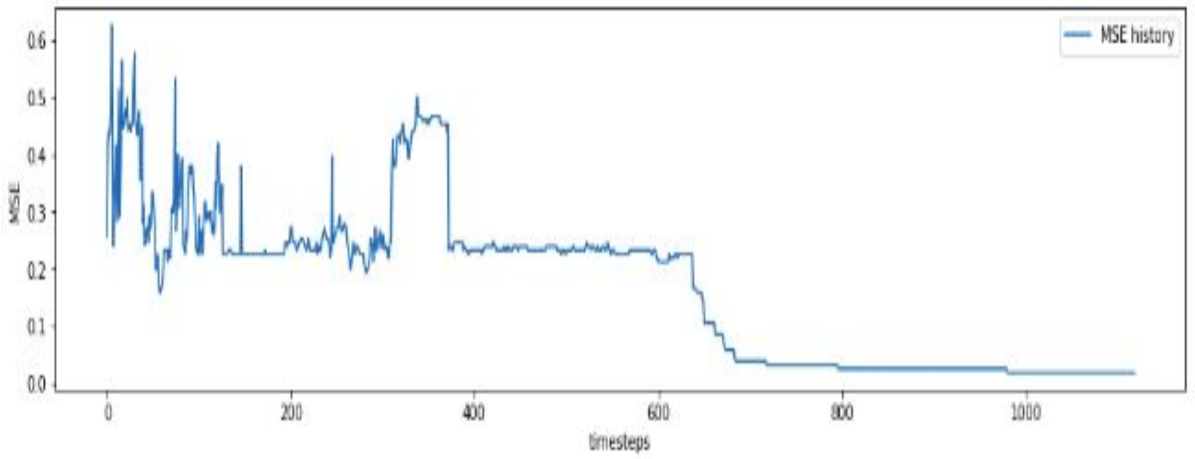
* - Not exact result

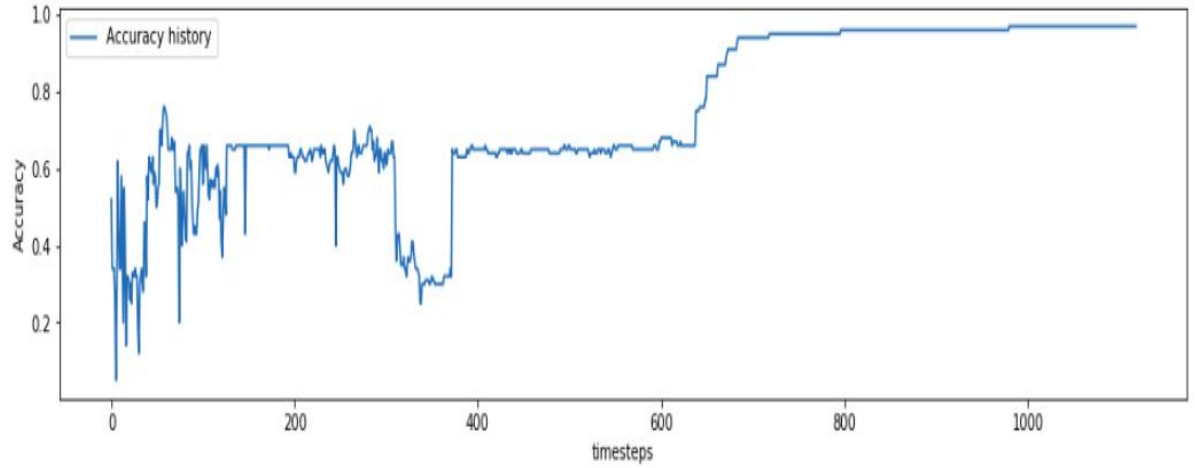
For annealing rate = 0.9:



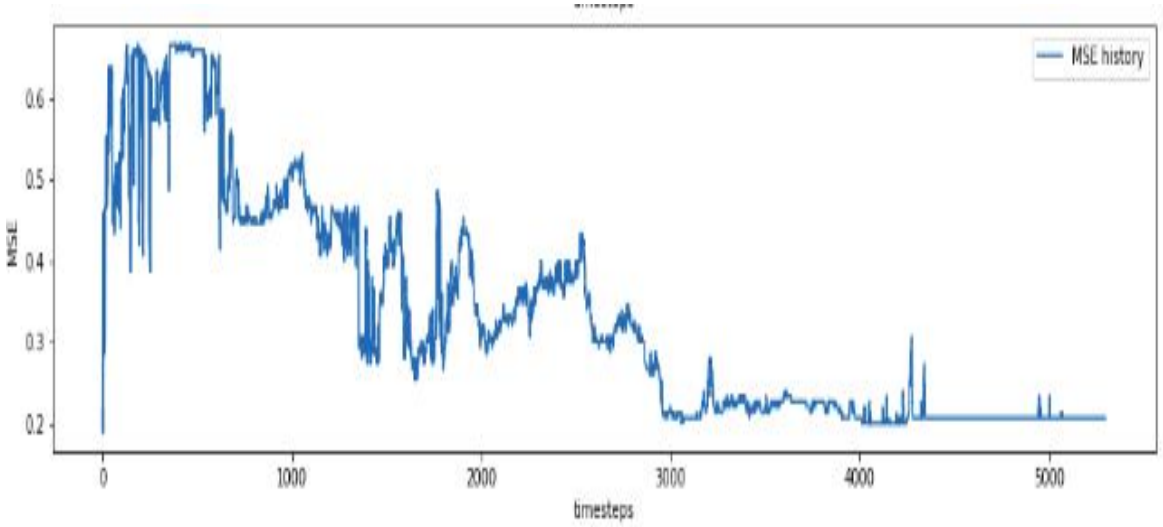


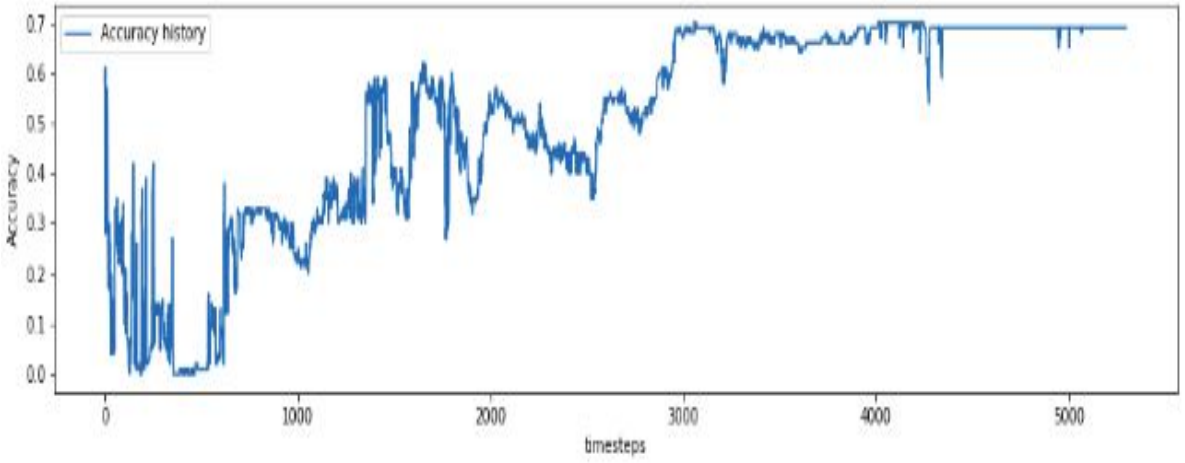
For annealing rate = 0.99:





For annealing rate = 0.999:





Conclusion

In my opinion, for learning parameters of NN we should use gradient descent, because it work faster than annealing. If we will compare only annealing technique, we should use medium annealing rate (0.99), because it performs better results:

fast annealing rate underfit, slow annealing rate slowly learn and take much time.

Part 2:

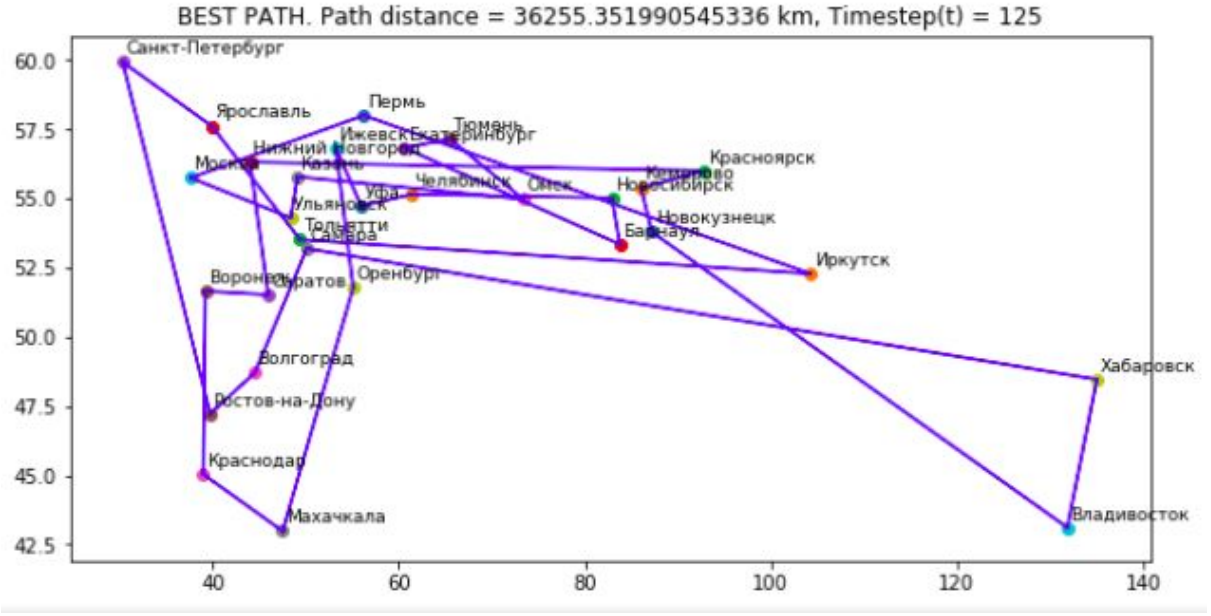
In this part I try to use different start temperatures, but with T > 100000 it work slowly and with T < 1000 it fastly crash, so I use T which equals firstly Energy function value: 50000

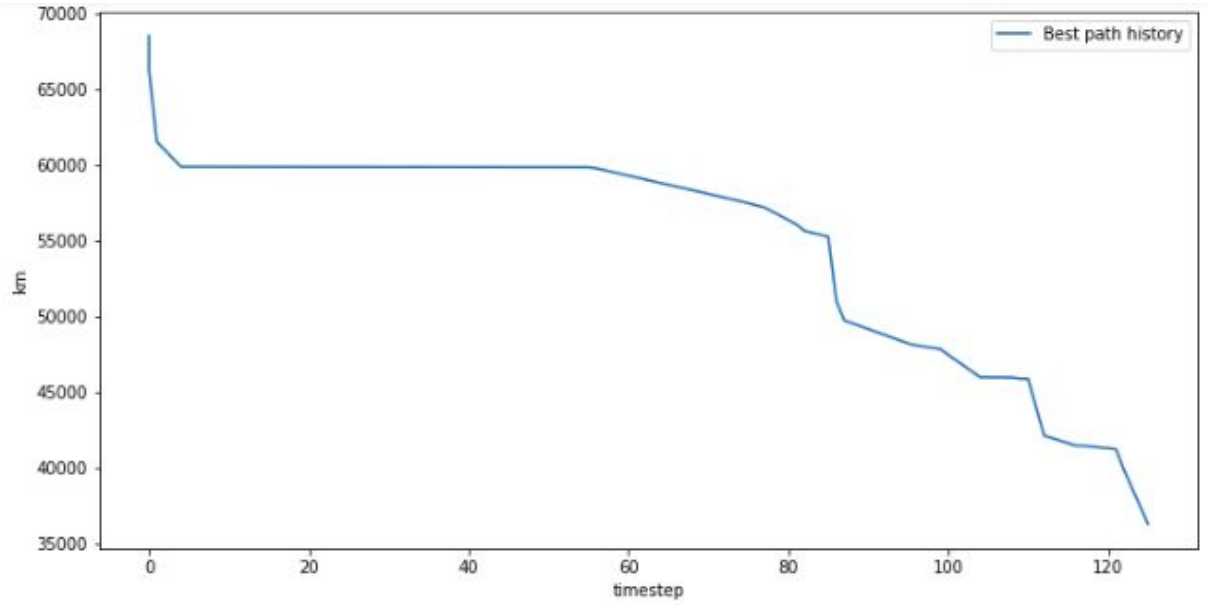
Table with different annealing rate take you can see below:

Annealing rate	Total steps	Total time (seconds)	Best path (km)
0.9 (Fast)	125	1.3	36255
0.99 (Medium)	1451	14.12	25485
0.999 (Slow)	1520	148.24	18547

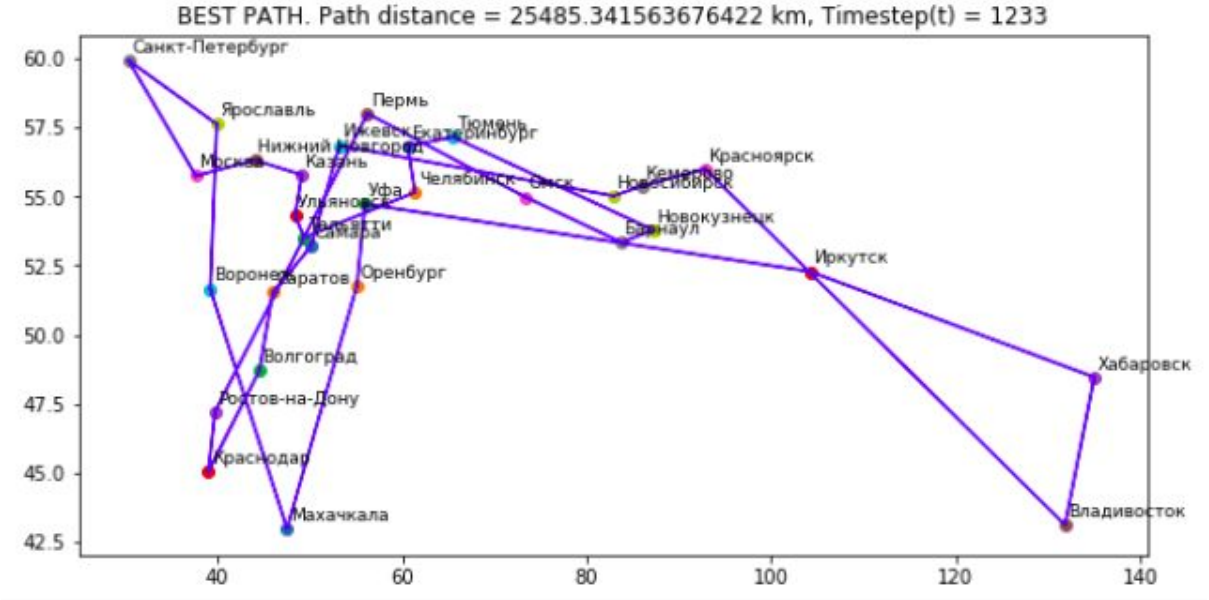
If we use different number of steps (N) before annealing, it only increase time complexity of the algorithm in N times and result will be the same, so I decide to use N = 2

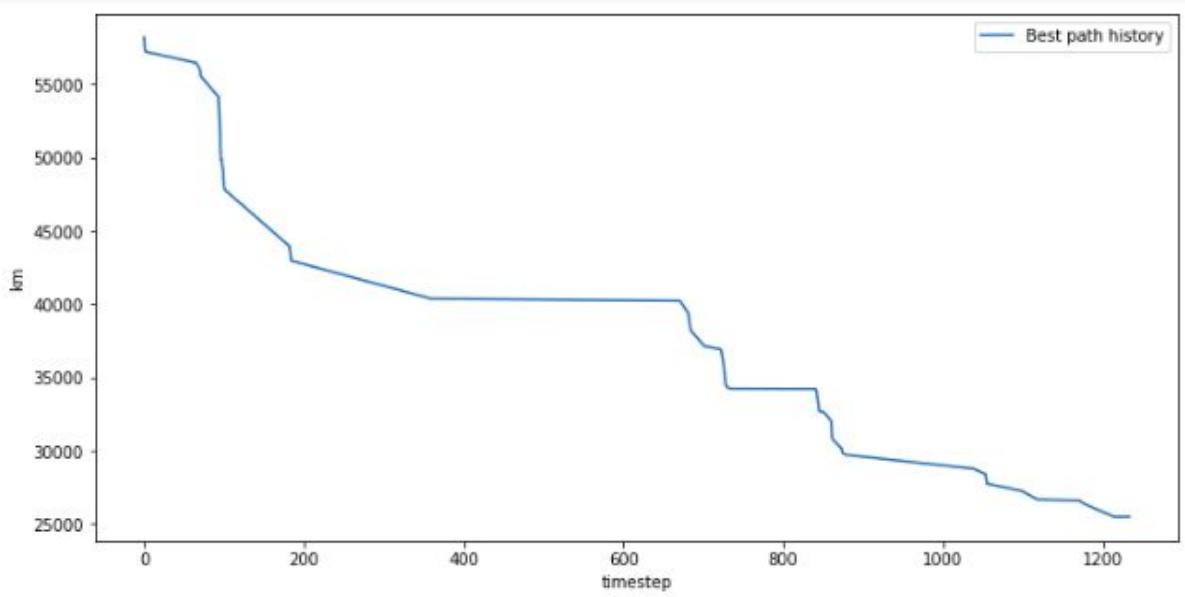
Annealing rate = 0.9



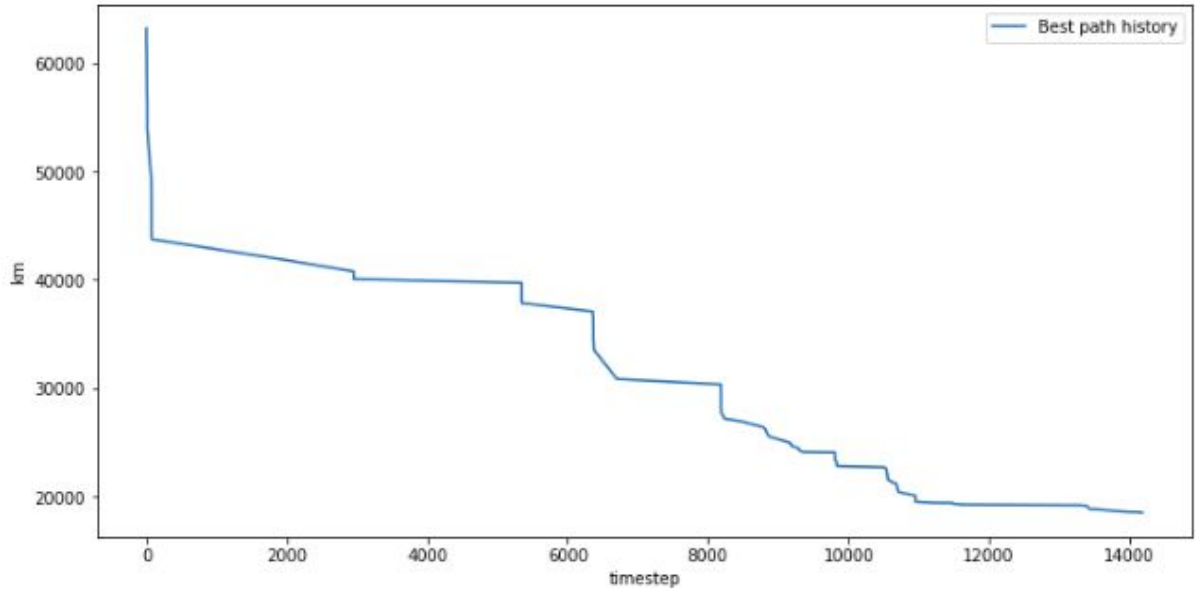


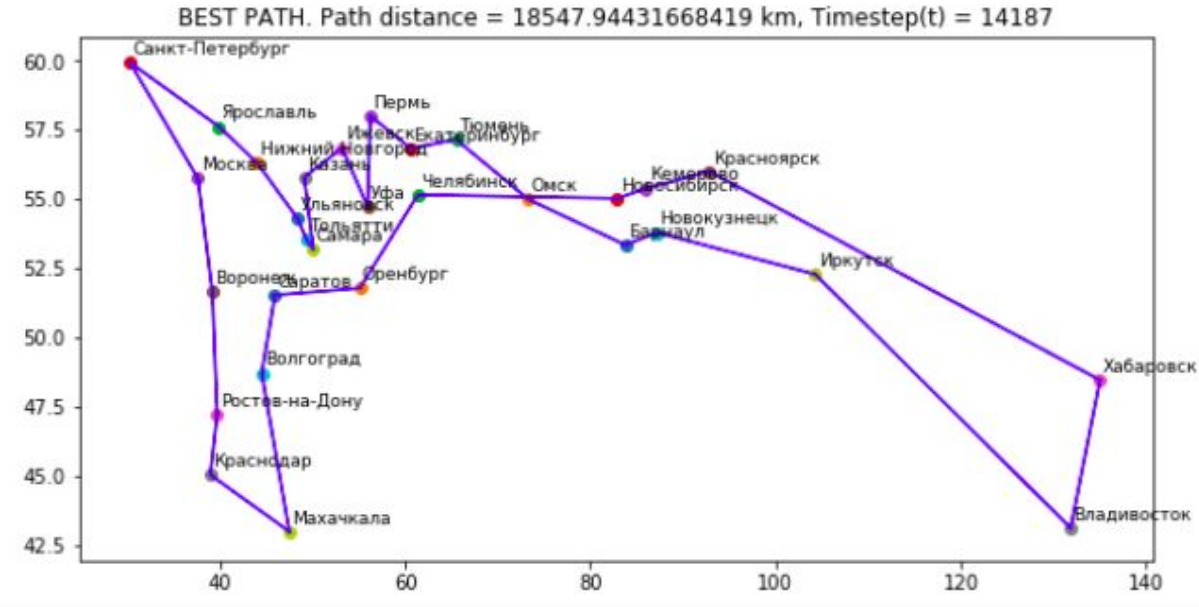
Annealing rate = 0.99




Best path history
55000
50000
45000
40000
35000
30000
25000
km
0
200
400
600
800
1000
1200
timestep

Annealing rate = 0.999





Conclusion: it works. It took 4 hours to build a graph and create an animation.