Experimenting Simulated Annealing on the Traveling Salesman Problem

Due Tuesday, 2023 May 2, 11:59 p.m. EST

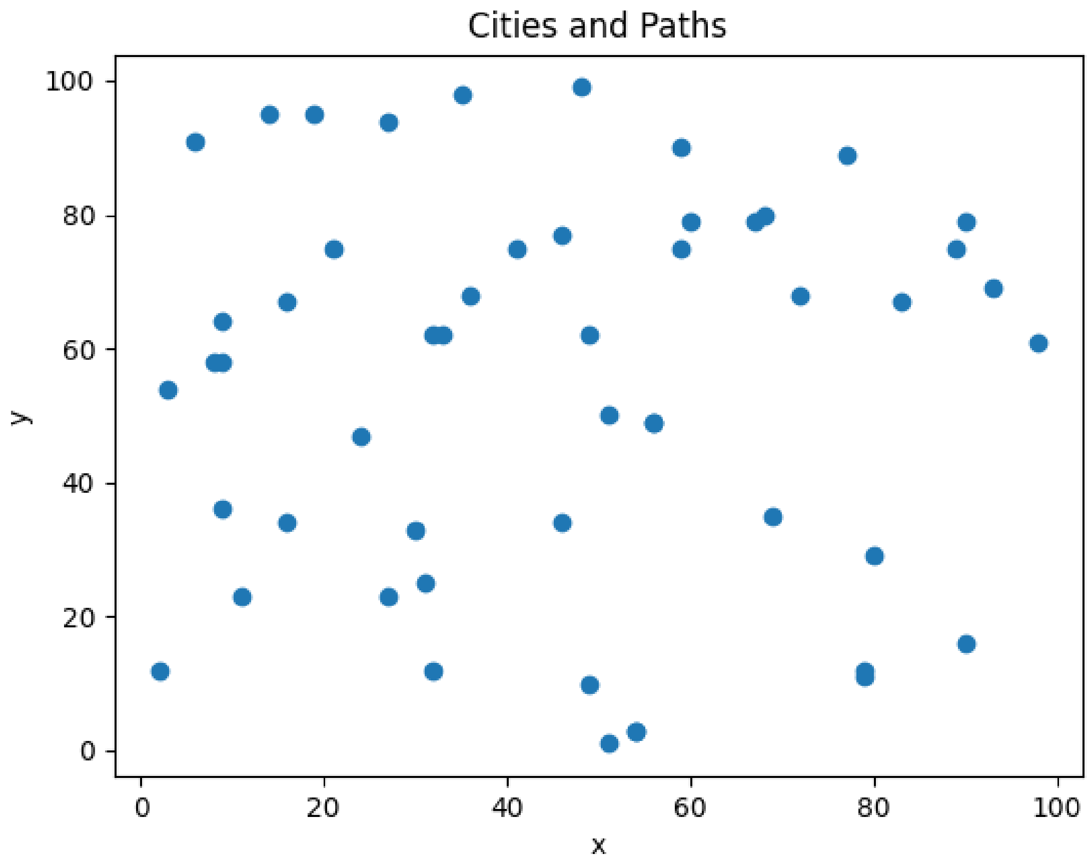
Introduction

Traveling Salesman problem(TSP) asks:"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?" This question is known as the traveling salesman problem, which is an NP-hard combinatorial optimization problem.[1]

In math, we can formulate it as:"Given a list of n cities $c_1, c_2, ..., c_n$, and given the distance between each pair of cities i and j as d_ij, find a permutation of the list 1,2,...,n in the form of $p_1, p_2, p_3, ..., p_n$, where $p_k \in$ list of n cities for all k, such that the cost $\Sigma_{i=1}^{n} d_{p_i p_{i+1}} + d_{p_n 1}$ is minimized."

Question Specification

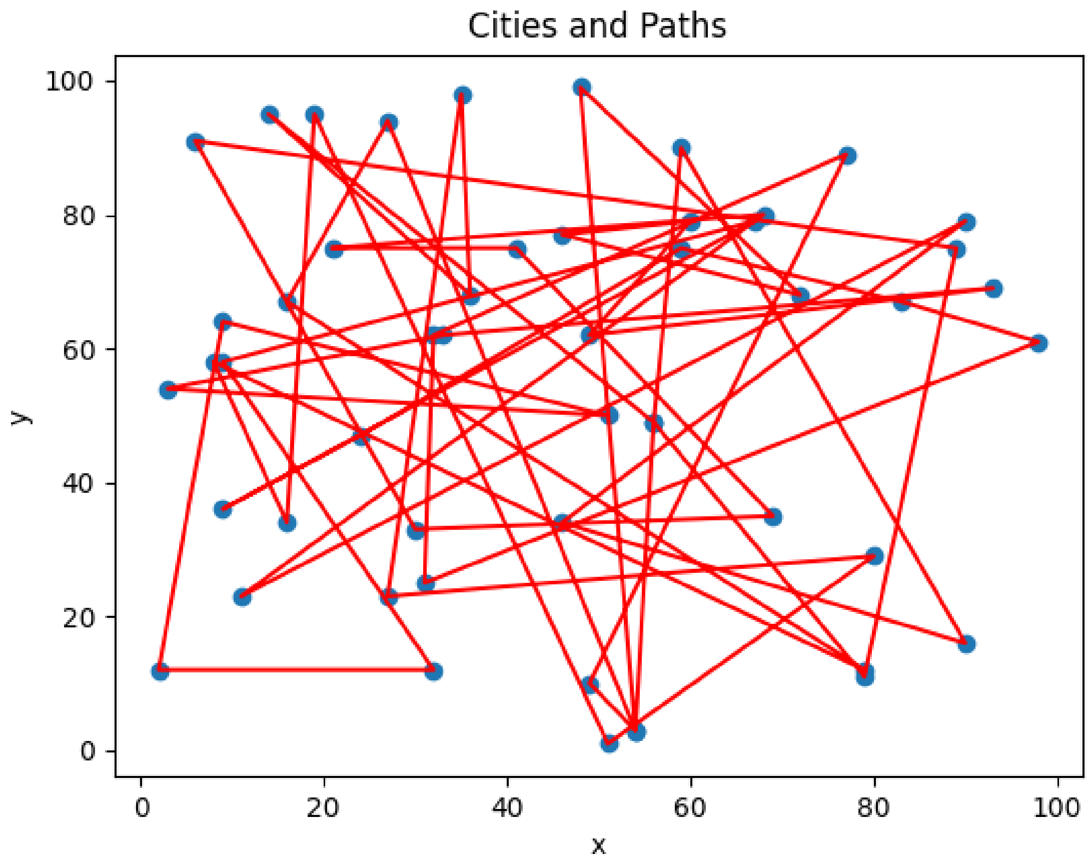
I randomly generated a list of 52 cities and want to find the path(a sequence of cities from the list that has no repetition and length 52) that has the minimal cost using Simulated Annealing Algorithm[2](SA). The list can be found in data/cities.data. Below is a plot of the coordinates of the cities.

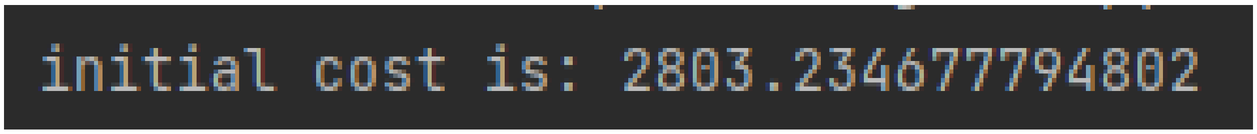


The cost is defined same as above, Cost $= \Sigma_{i=1}^{n} d_{p_i p_{i+1}} + d_{p_n 1}$
The acceptance probability is defined as:
P(accept w' as new solution) $= exp((-w' + w)/t_k)$, where w is the current solution and t_k is the current temperature.

Numerical Experiments

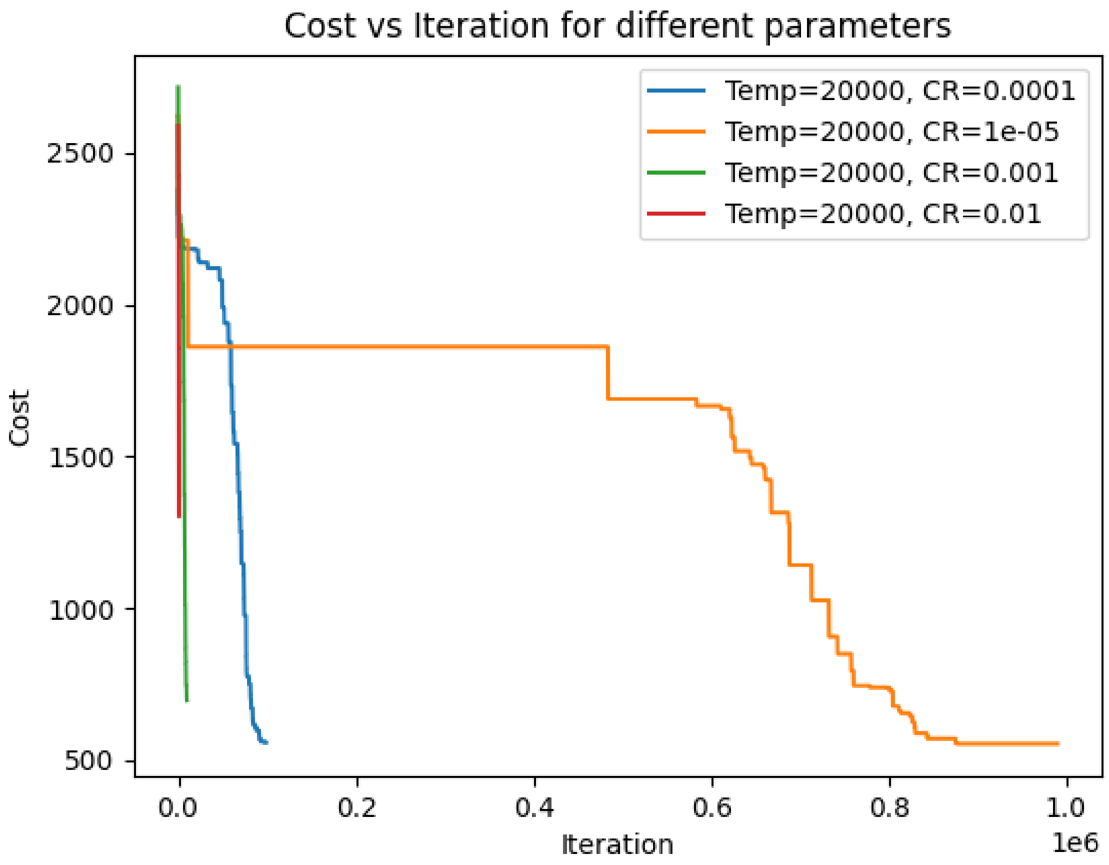
Here is a plot representation of a randomly generated solution using shuffle on the original list with random.seed(42). As we can see, the initial cost is 2803.23.

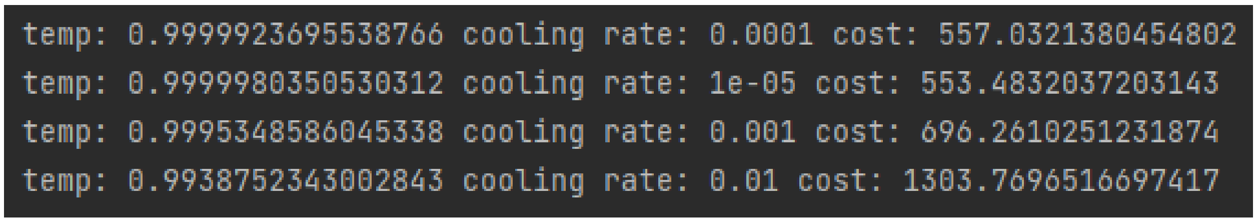




```
initial cost is: 2803.234677794802
```

We first try running SA using a list parameters [(20000,1e-4),(20000,1e-5),(20000,1e-3),(20000,1e-2)]. That is fixing temperature = 20000, but use cooling rate from 1e-5 to 1e-2. Below are the plot of Total Cost vs the number of iterations and the numerical results.

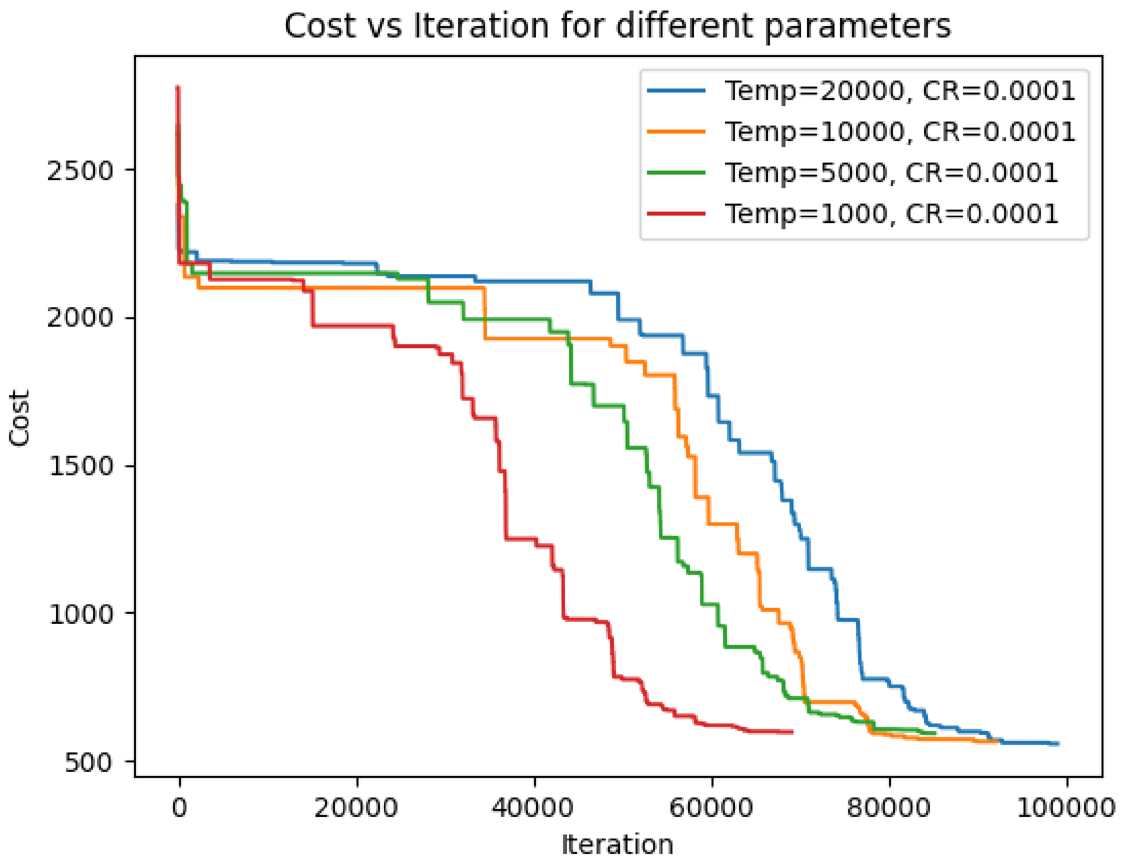


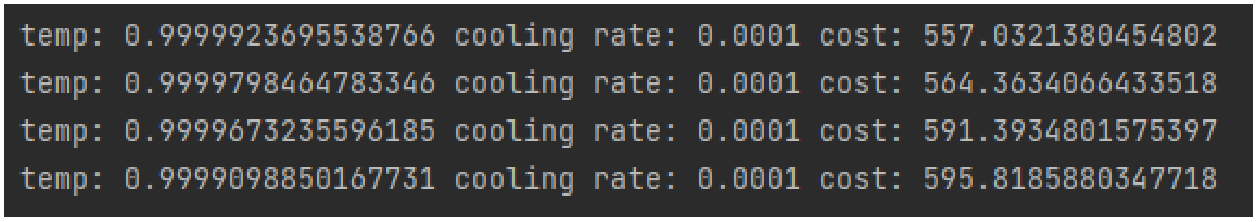


```
temp: 0.9999923695538766 cooling rate: 0.0001 cost: 557.0321380454802
temp: 0.9999980350530312 cooling rate: 1e-05 cost: 553.4832037203143
temp: 0.9995348586045338 cooling rate: 0.001 cost: 696.2610251231874
temp: 0.9938752343002843 cooling rate: 0.01 cost: 1303.7696516697417
```

With the same initial temperature, the smaller the cooling rate, the longer it take to get the final solution, but the cost value is more optimized.

Then I try running SA using another list of parameters = [(20000,1e-4),(10000,1e-4),(5000,1e-4),(1000,1e-4)]. That is fixing cooling rate = 1e-4, but use temperature 20000,10000,5000,1000. Below are the plot of cost change over iterations and the numerical results.

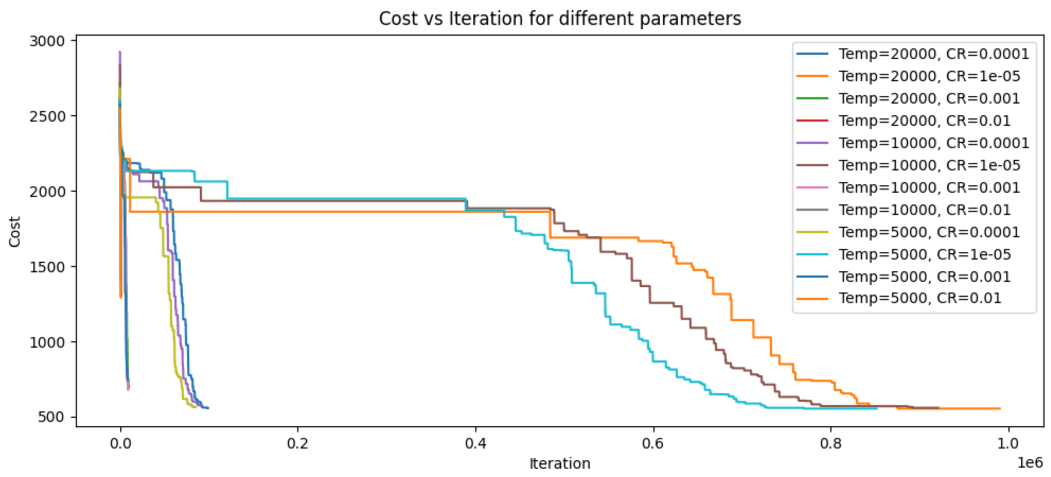


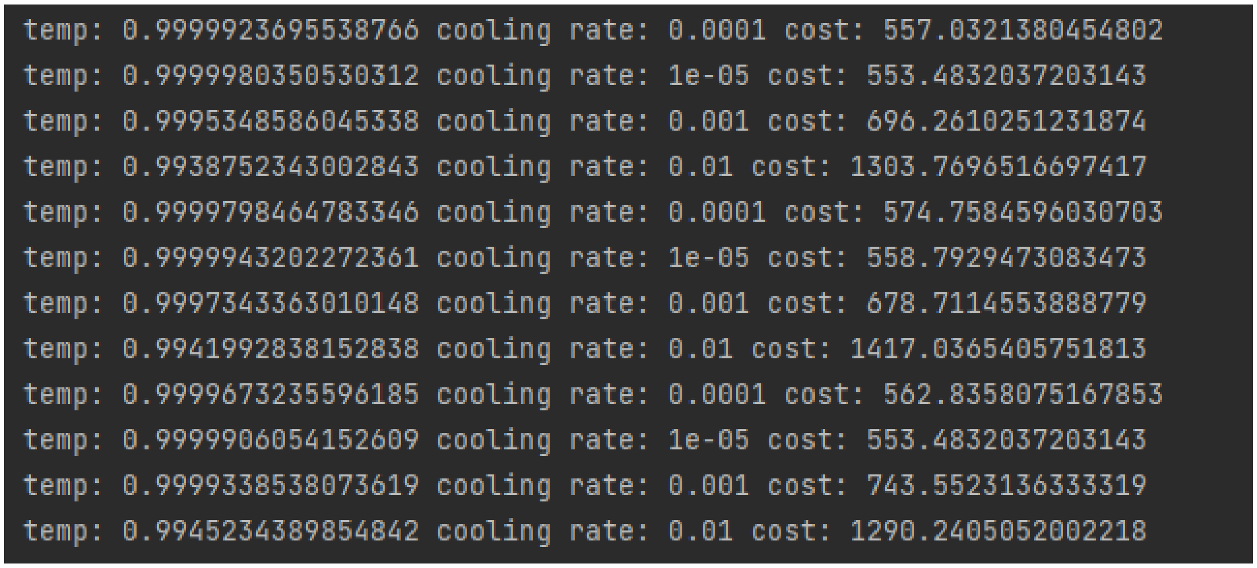


```
temp: 0.9999923695538766 cooling rate: 0.0001 cost: 557.0321380454802
temp: 0.9999798464783346 cooling rate: 0.0001 cost: 564.3634066433518
temp: 0.9999673235596185 cooling rate: 0.0001 cost: 591.3934801575397
temp: 0.9999098850167731 cooling rate: 0.0001 cost: 595.8185880347718
```

Fixing cooling rate, the converge speed and the final cost value actually gives very close results. However, we can see a trend of "right shift" with higher initial temperature. It make sense as it takes longer for higher temperature to cool down.

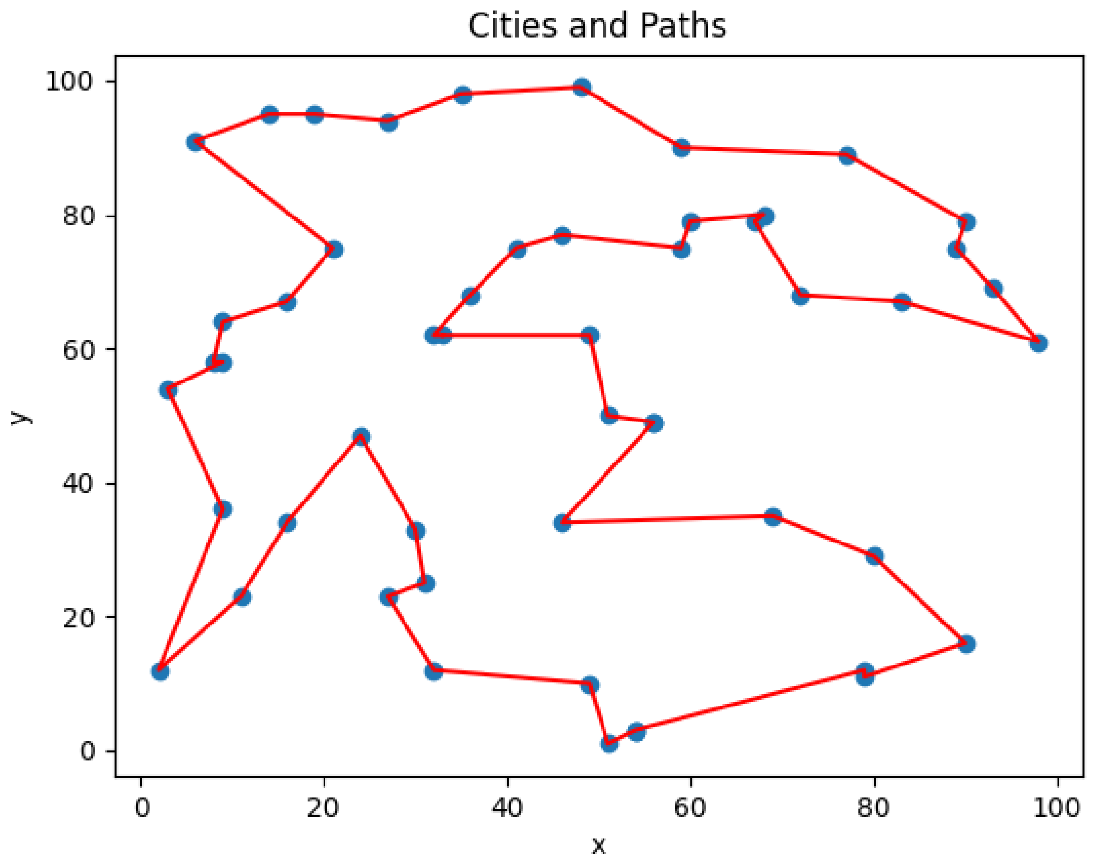
Finally, I try to compare the results with temperature = 20000,10000,5000 and cooling rate = 1e-5,1e-4,1e-3,1e-2. And below are the plot and the numerical results.

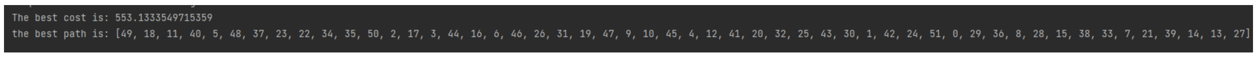




```
temp: 0.9999923695538766 cooling rate: 0.0001 cost: 557.0321380454802
temp: 0.9999980350530312 cooling rate: 1e-05 cost: 553.4832037203143
temp: 0.9995348586045338 cooling rate: 0.001 cost: 696.2610251231874
temp: 0.9938752343002843 cooling rate: 0.01 cost: 1303.7696516697417
temp: 0.9999798464783346 cooling rate: 0.0001 cost: 574.7584596030703
temp: 0.9999943202272361 cooling rate: 1e-05 cost: 558.7929473083473
temp: 0.9997343363010148 cooling rate: 0.001 cost: 678.7114553888779
temp: 0.9941992838152838 cooling rate: 0.01 cost: 1417.0365405751813
temp: 0.9999673235596185 cooling rate: 0.0001 cost: 562.8358075167853
temp: 0.9999906054152609 cooling rate: 1e-05 cost: 553.4832037203143
temp: 0.9999338538073619 cooling rate: 0.001 cost: 743.5523136333319
temp: 0.9945234389854842 cooling rate: 0.01 cost: 1290.2405052002218
```

As we can see, again the cooling rate dominates the shape of the cost function over iteration, and the higher the temperature, the longer it takes to converge(given a fixed cooling rate).
The best path given by the SA algorithm is: [49, 18, 11, 40, 5, 48, 37, 23, 22, 34, 35, 50, 2, 17, 3, 44, 16, 6, 46, 26, 31, 19, 47, 9, 10, 45, 4, 12, 41, 20, 32, 25, 43, 30, 1, 42, 24, 51, 0, 29, 36, 8, 28, 15, 38, 33, 7, 21, 39, 14, 13, 27], with the cost 553.1333.
Below is the plot representation of this result.





```
The best cost is: 553.1333549715359
the best path is: [49, 18, 11, 40, 5, 48, 37, 23, 22, 34, 35, 50, 2, 17, 3, 44, 16, 6, 46, 26, 31, 19, 47, 9, 10, 45, 4, 12, 41, 20, 32, 25, 43, 30, 1, 42, 24, 51, 0, 29, 36, 8, 28, 15, 38, 33, 7, 21, 39, 14, 13, 27]
```

References

[1]https://en.wikipedia.org/wiki/Travelling_salesman_problem
[2]Henderson, Darrall & Jacobson, Sheldon & Johnson, Alan. (2006).The Theory and Practice of Simulated Annealing. 10.1007/0-306-48056-5_10.