

Travelling Salesman Problem (TSP)

NAME	STUDENT NUMBER	EMAIL ID
Aishwarya Manapuram	300322316	amana022@uottawa.ca
Ahmed Farooqui	300334347	afaro053@uottawa.ca

Intuition

For this assignment, we tried to experiment and code our own algorithm which resembles the closest to First Ascent Hill Climbing mixed with Random Restart.

When it comes to Hill Climbing or Simulated Annealing, the algorithm remains the same throughout all implementations, however, what differs one from the other is:

1. Number of Iterations to run the algorithm for
2. How to select the neighbors

The simplest way to select the neighbors is using a random swap function. However, our intuition was, can optimize this swap function or at least make it more intelligent than just a random swap every time?

Consider this example:

My cities are in this given order:

1-2-3-4-5-6-7-8-9-10-11-12-13-14-15

In the first iteration, we take this order.

In the second iteration, we can randomly swap any two cities:

Let's say we swap city 5 and city 11.

So old circuit was 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15

The new circuit in this iteration is 1-2-11-4-5-6-7-8-9-10-3-12-13-14-15

Let us believe that the distance to travel this new circuit is lesser than the distance to travel the first circuit.

If so, what path changed?

If the new solution is better than the old one then we can definitely say that

$\text{distance}(2-11-4) + \text{distance}(10-3-12) > \text{distance}(2-3-4) + \text{distance}(10-11-12)$

where $\text{distance}(a-b-c)$ represents (distance to move from a to b + distance to move from b to c)

So, why touch these paths? Let's focus on optimizing further the other branches in this sequence.

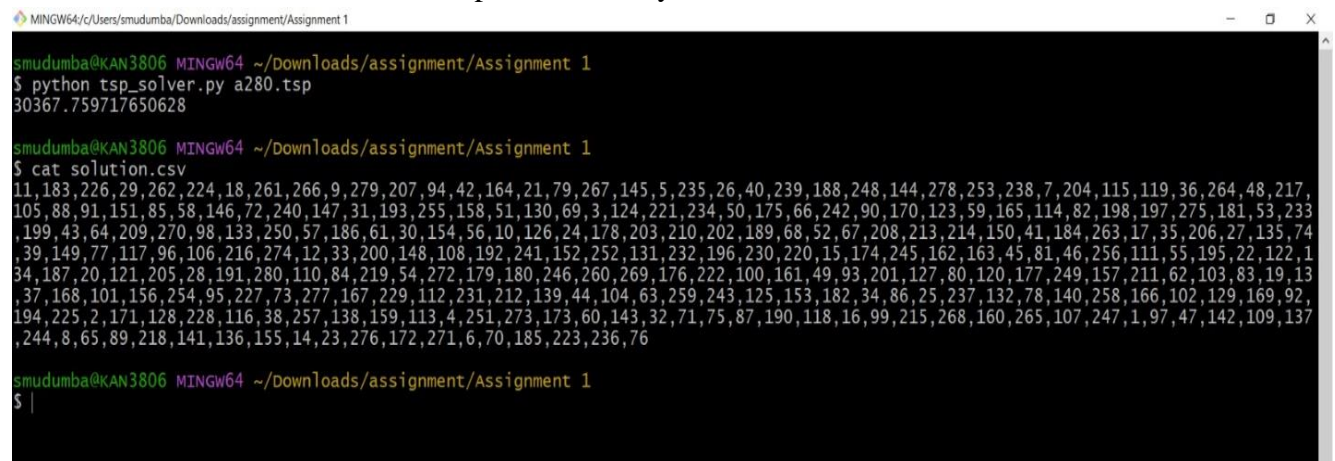
So, the branches are (1), (5-6-7-8-9) and (13-14-15). We choose any one of these branches randomly and try to optimize within this branch by swapping only within it. And we keep doing it like a depth first search.

Once a branch can no longer be branched, we do a random restart to look for other regions to explore and branch.

This algorithm would produce better results with more iterations as we explore more branches.

Result:

The below screenshot shows the output from our system console.



```

MINGW64~/Users/smudumba/Downloads/assignment/Assignment 1
smudumba@KAN3806 MINGW64 ~/Downloads/assignment/Assignment 1
$ python tsp_solver.py a280.tsp
30367.759717650628

smudumba@KAN3806 MINGW64 ~/Downloads/assignment/Assignment 1
$ cat solution.csv
11,183,226,29,262,224,18,261,266,9,279,207,94,42,164,21,79,267,145,5,235,26,40,239,188,248,144,278,253,238,7,204,115,119,36,264,48,217,
105,88,91,151,85,58,146,72,240,147,31,193,255,158,51,130,69,3,124,221,234,50,175,66,242,90,170,123,59,165,114,82,198,197,275,181,53,233
,199,43,64,209,270,98,133,250,57,186,61,30,154,56,10,126,24,178,203,210,202,189,68,52,67,208,213,214,150,41,184,263,17,35,206,27,135,74
,39,149,77,117,96,106,216,274,12,33,200,148,108,192,241,152,252,131,232,196,230,220,15,174,245,162,163,45,81,46,256,111,55,195,22,122,1
34,187,20,121,205,28,191,280,110,84,219,54,272,179,180,246,260,269,176,222,100,161,49,93,201,127,80,120,177,249,157,211,62,103,83,19,13
,37,168,101,156,254,95,227,73,277,167,229,112,231,212,139,44,104,63,259,243,125,153,182,34,86,25,237,132,78,140,258,166,102,129,169,92,
194,225,2,171,128,228,116,38,257,138,159,113,4,251,273,173,60,143,32,71,75,87,190,118,16,99,215,268,160,265,107,247,1,97,47,142,109,137
,244,8,65,89,218,141,136,155,14,23,276,172,271,6,70,185,223,236,76

smudumba@KAN3806 MINGW64 ~/Downloads/assignment/Assignment 1
$

```

The output produced is stored under solution.csv as shown above. You can find the output attached in the folder too.