Our problem to solve was a modified version of the Travel Salesman Problem: additionally, we have costs associated with each node, and we had to select exactly 50% of the total nodes.

Pseudo codes:

Greedy 2-regret heuristics & Greedy heuristics with a weighted sum criterion

1. Calculate distance matrix between each pair of nodes, add to each distance avg cost of both nodes.
2. For the initial node find NN, and add edges between them(in both directions) to the output cycle.
3. While output cycle length is less than half of length of input:
   3.1. For each **node**(N) not yet in output cycle:
      3.1.1. For **edge**(N_1,N_2) in output cycle:
         3.1.1.1. Calculate **change** in distance(**edge**(N_1,N) + **edge**(N_2,N) - **edge**(N_1,N_2)
      3.1.2. Sort all distance change in ascending order
      3.1.3. Calculate regret value (difference between the first and second change)
      3.1.4. Calculate **value = weight_regret * regret + weight_cost * first change**
   3.2. Select the best edge, for which value is minimal
   3.3. Remove "useless" edge from cycle, and add 2 newones

Code: https://github.com/Imrauviel/Evolutionary-computation

Conclusion:

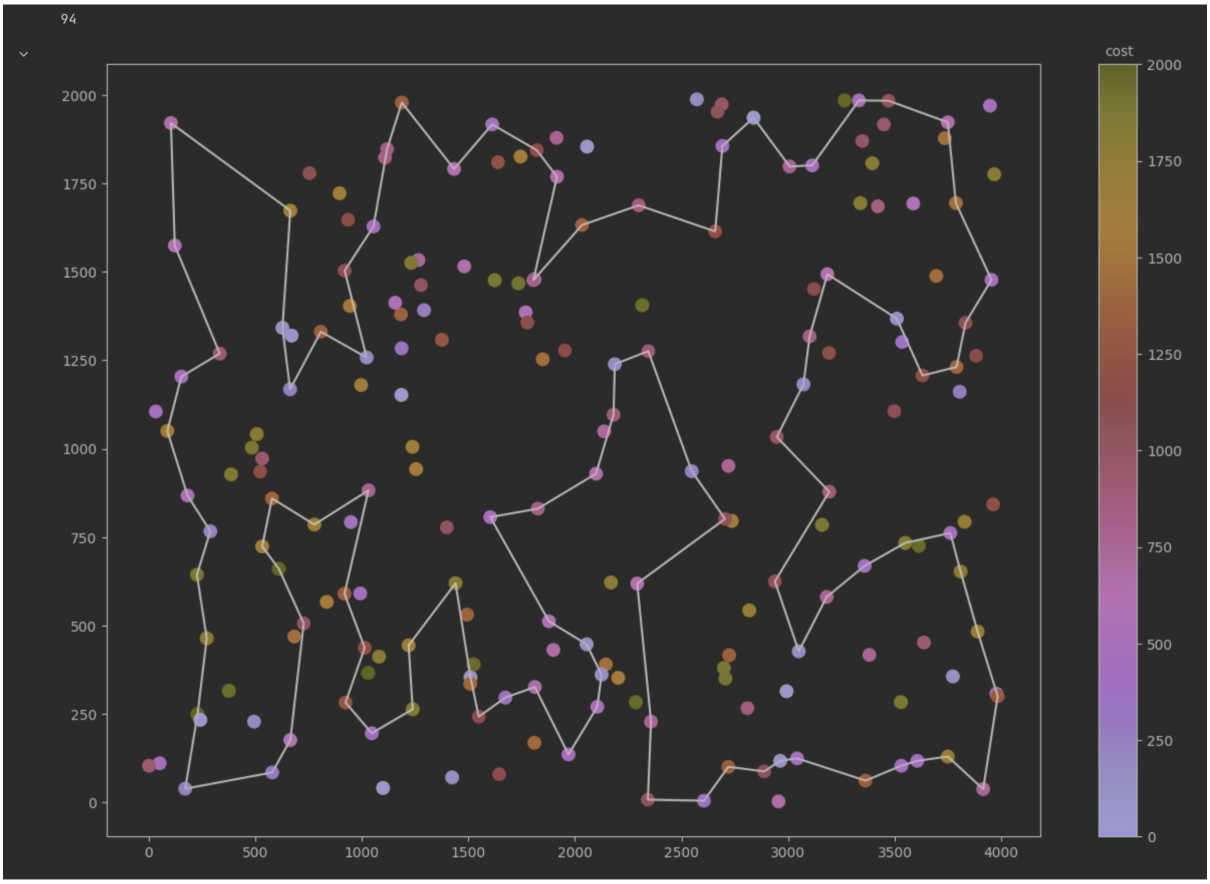
Problem with 2-regret: 1st choice is aribtrary(when we have only 2 nodes in cycle, regretes for all other nodes are the same)

TSPA	Random	NN	Greedy Cycle	Greedy 2-Regret	Greedy with weighted sum
Min	244344	84149	75566	109024	74514
Max	291673	97583	79799	123981	78926
Mean	267442	82446	70656	116162	76292

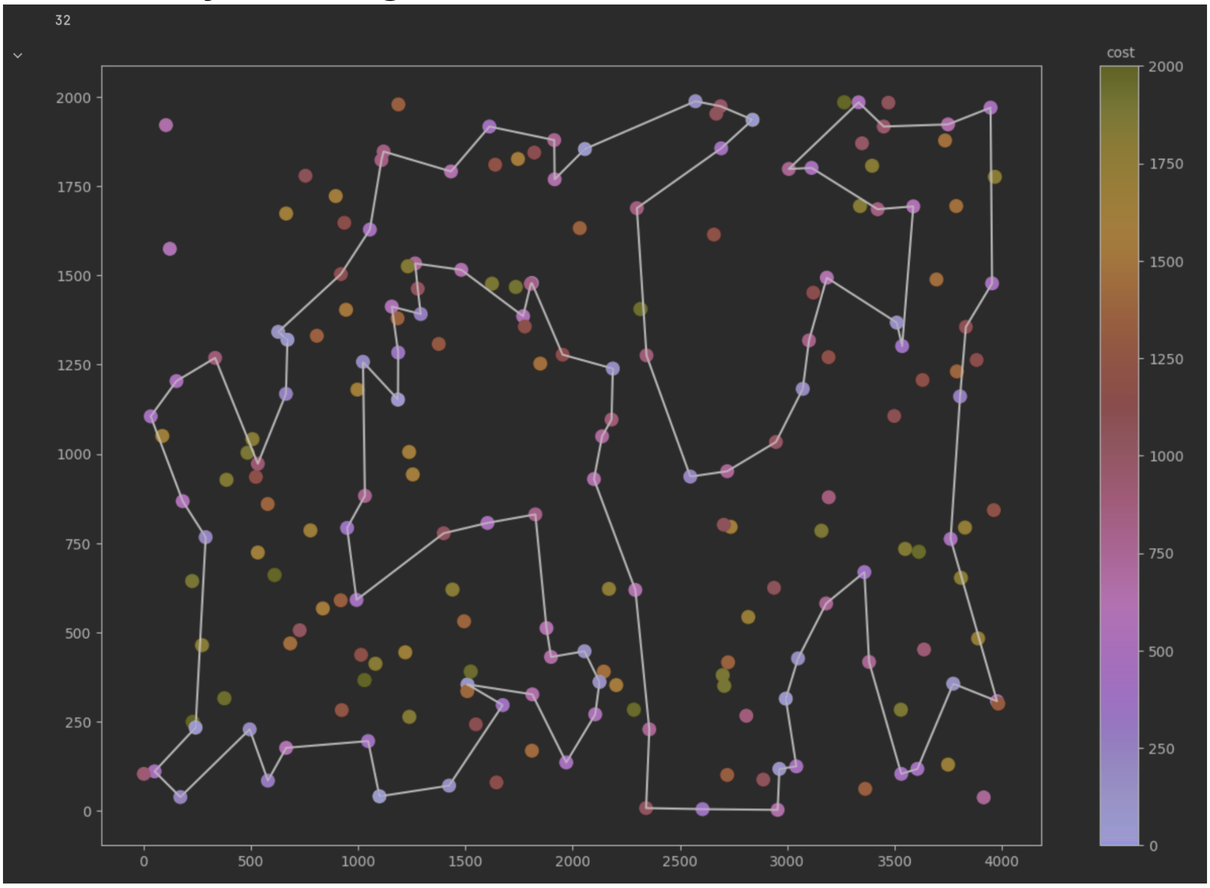
TSPB	Random	NN	Greedy Cycle	Greedy 2-Regret	Greedy with weighted sum
Min	242446	74413	68655	107990	69686
Max	294115	93154	76230	127540	78503
Mean	267442	82446	70656	118615	71736

TSP_A

Greedy 2-Regret

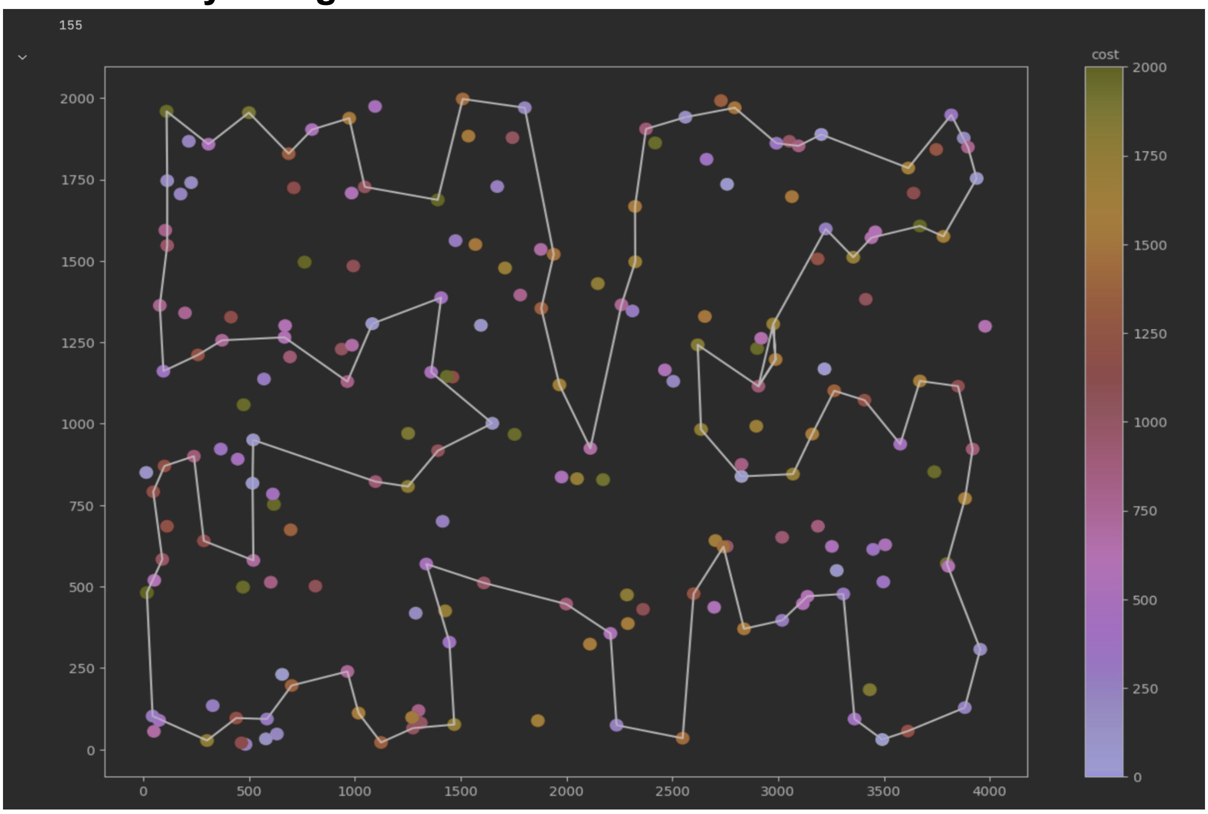


Greedy with weighted sum



TSP_B

Greedy 2-Regret



Greedy with weighted sum