Evolutionary Computation - lab assignment 2

Szymon Bujowski, 148050, source: https://github.com/bujowskis/put-evolutionary-computation

Problem description

We are given three columns of integers with a row for each node. The first two columns contain x and y coordinates of the node positions in a plane. The third column contains node costs. The goal is to select exactly 50% of the nodes (if the number of nodes is odd we round the number of nodes to be selected up) and form a Hamiltonian cycle (closed path) through this set of nodes such that the sum of the total length of the path plus the total cost of the selected nodes is minimized.

The distances between nodes are calculated as Euclidean distances rounded mathematically to integer values. The distance matrix should be calculated just after reading an instance and then only the distance matrix (no nodes coordinates) should be accessed by optimization methods to allow instances defined only by distance matrices.

The task is to implement 2 methods based on greedy cycle heuristic, adapted to the problem:

- Greedy 2-regret heuristics
- Greedy heuristics with a weighted sum criterion – 2-regret + best change of the objective function. By default use equal weights but you can also experiment with other values

For each greedy method generate 200 solutions starting from each node. The results will be compared to the ones obtained during assignment 1.

Implemented algorithms and pseudocodes

Notes

- 50% was the required number of nodes in a solution, and “required number of nodes” will be referred to this way in the pseudocode
- “**Total move cost**” is the total change in objective function after adding a node, that is, its edge distance and additional cost

Greedy 2-regret heursitics

1. Choose a starting node
2. Keep track of chosen and remaining nodes. Chosen nodes form the cycle
3. WHILE number of chosen nodes is not 50% AND there are still remaining nodes
3.1. Determine total move costs of inserting any of remaining nodes between pairs of chosen nodes
3.2. FOR EACH such pair:
3.2.1. Choose a node with the smallest total move cost
3.2.2. Form a cycle resulting from inserting the node - remove
3.2.3. Determine total move costs of inserting any of remaining nodes between pairs of chosen nodes within formed cycle
3.2.4. Choose a node with the smallest total move cost - save the sum of total move cost obtained in 3.2.1. And 3.2.4.
3.3. Choose a node with the smallest sum of total move cost obtained in 3.2.4.
3.4. Insert the node between its respective pair, remove it from remaining nodes
4. Form the cycle from the chosen nodes

Greedy 2-regret weighted objective heursitics

1. Choose a starting node
2. Keep track of chosen and remaining nodes. Chosen nodes form the cycle
3. WHILE number of chosen nodes is not 50% AND there are still remaining nodes
3.1. Determine total move costs of inserting any of remaining nodes between pairs of chosen nodes
3.2. FOR EACH such pair:
3.2.1. Choose a node with the smallest total move cost
3.2.2. Form a cycle resulting from inserting the node - remove
3.2.3. Determine total move costs of inserting any of remaining nodes between pairs of chosen nodes within formed cycle
3.2.4. Choose a node with the smallest total move cost - save the weighted sum of (total move cost obtained in 3.2.1. And 3.2.4.) and (total move cost obtained in 3.2.1.)
3.3. Choose a node with the smallest sum of the weighted sum obtained in 3.2.4.
3.4. Insert the node between its respective pair, remove it from remaining nodes
4. Form the cycle from the chosen nodes

Results

Note: All best solutions were checked with the solution checker

Statistics

TSPA

Statistic	NN at any	Greedy cycle	Greedy 2-regret	Greedy 2-regret weighted obj
Min obj fun	71179	71488	70978	70930
Max obj fun	75450	74350	73327	72970
Avg obj fun	73172.74	72589.82	71729.00	71740.89

TSPB

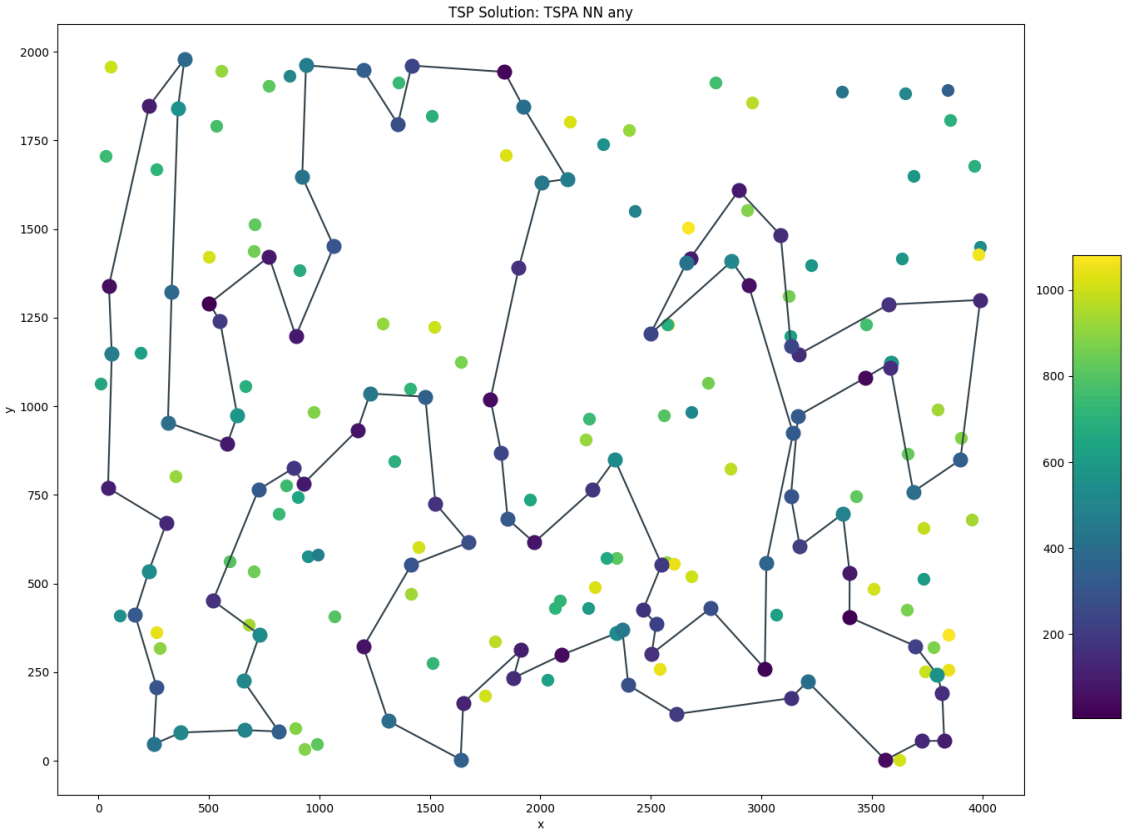
Statistic	NN at any	Greedy cycle	Greedy 2-regret	Greedy 2-regret weighted obj
Min obj fun	44417	48765	46018	46648
Max obj fun	53438	57262	51362	49364
Avg obj fun	45870.26	51389.38	48104.96	47978.02

Best solutions

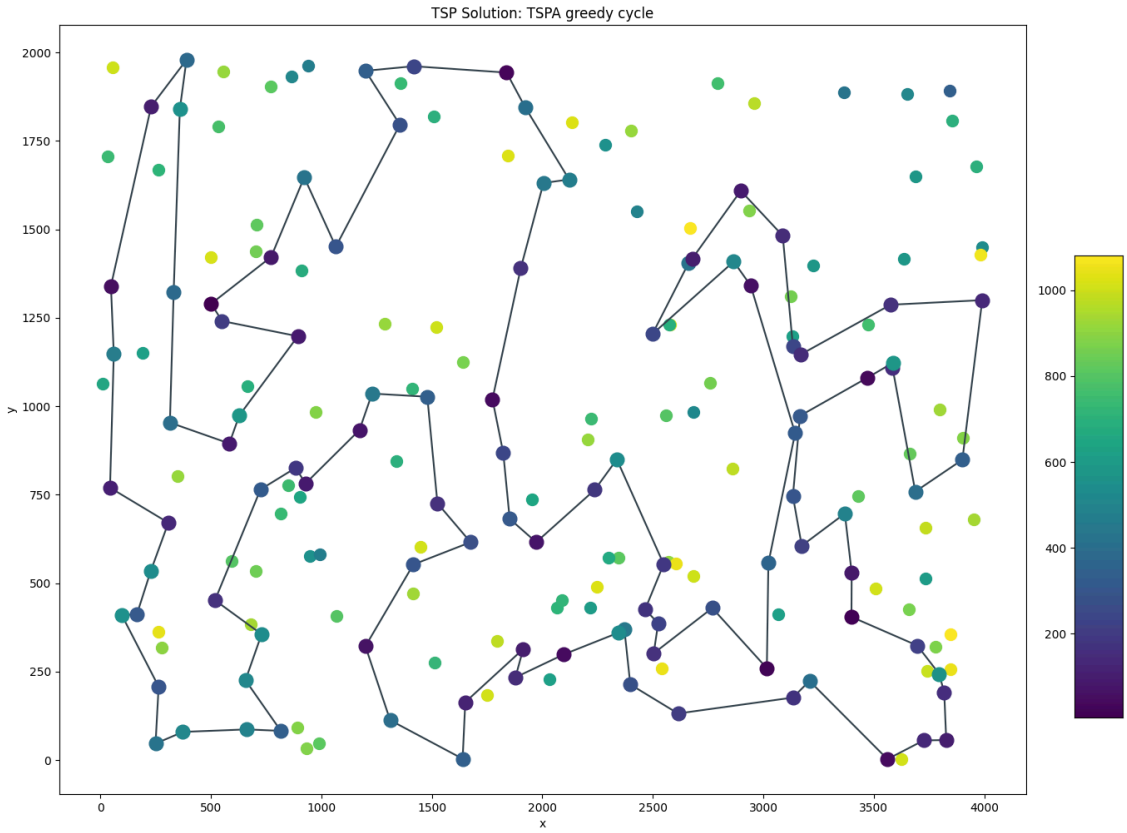
Note: additional cost is depicted using a color scale

TSPA

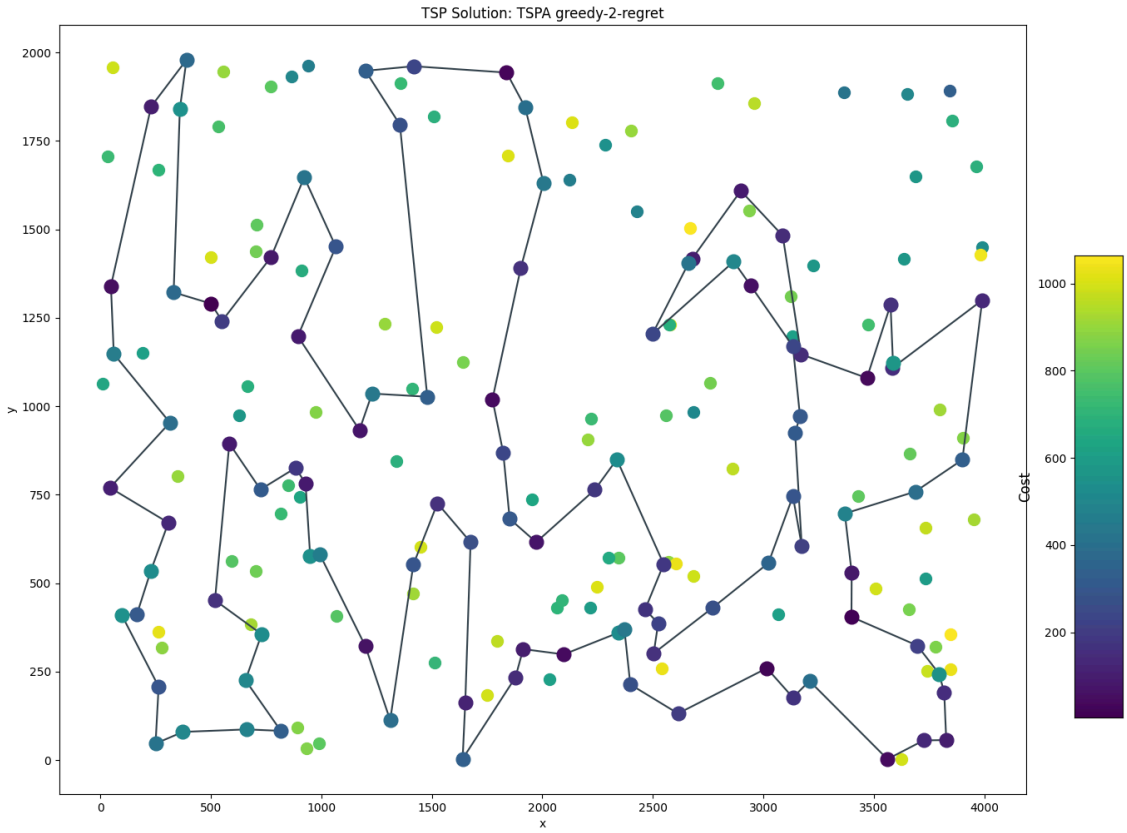
NN at any - [TSPA-nn-any-best.json]



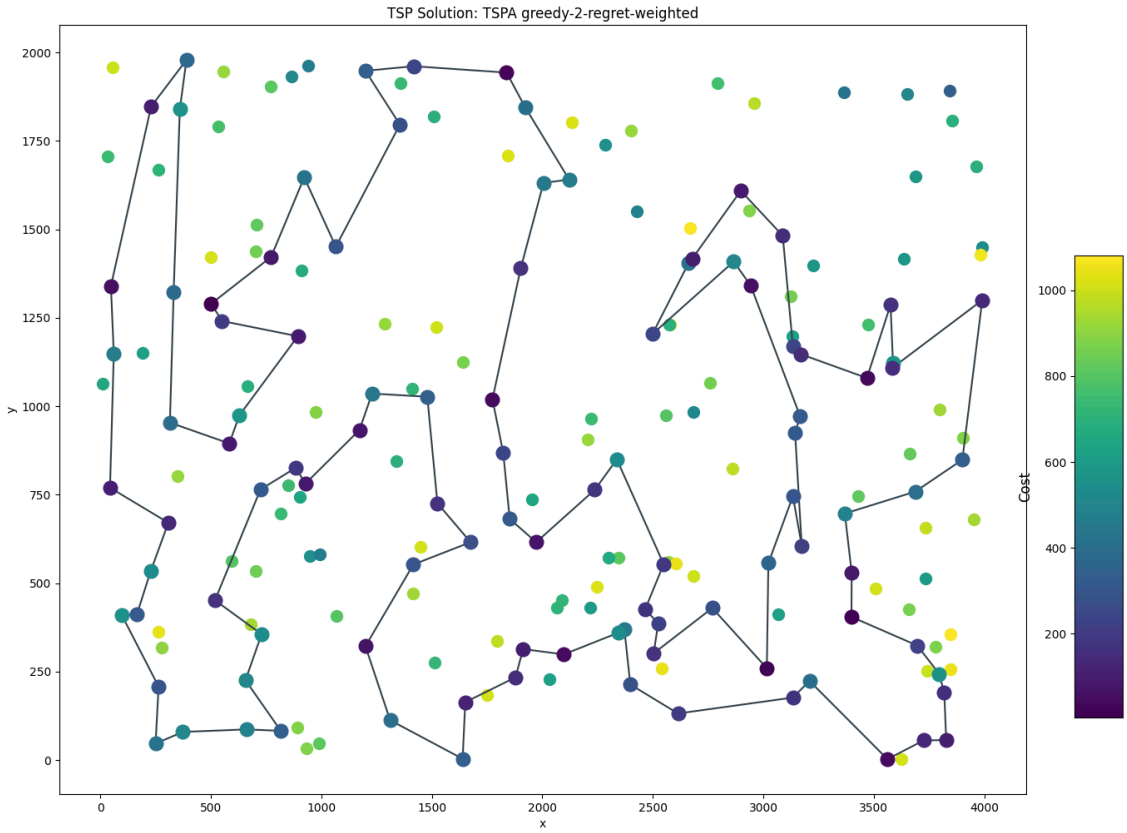
Greedy cycle - [TSPA-greedy-cycle-best.json]



Greedy 2-regret - [TSPA-greedy-2-regret-best.json]

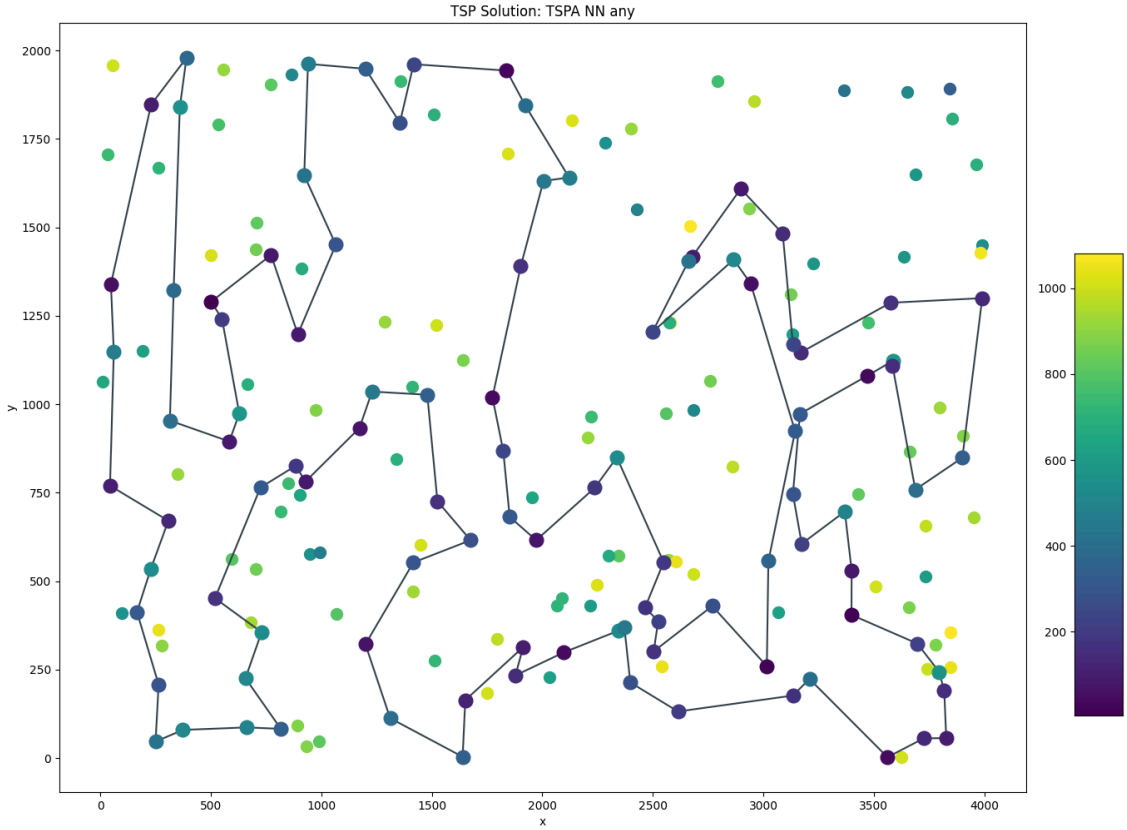


Greedy 2-regret weighted obj - [TSPA-greedy-2-regret-weighted-best.json]

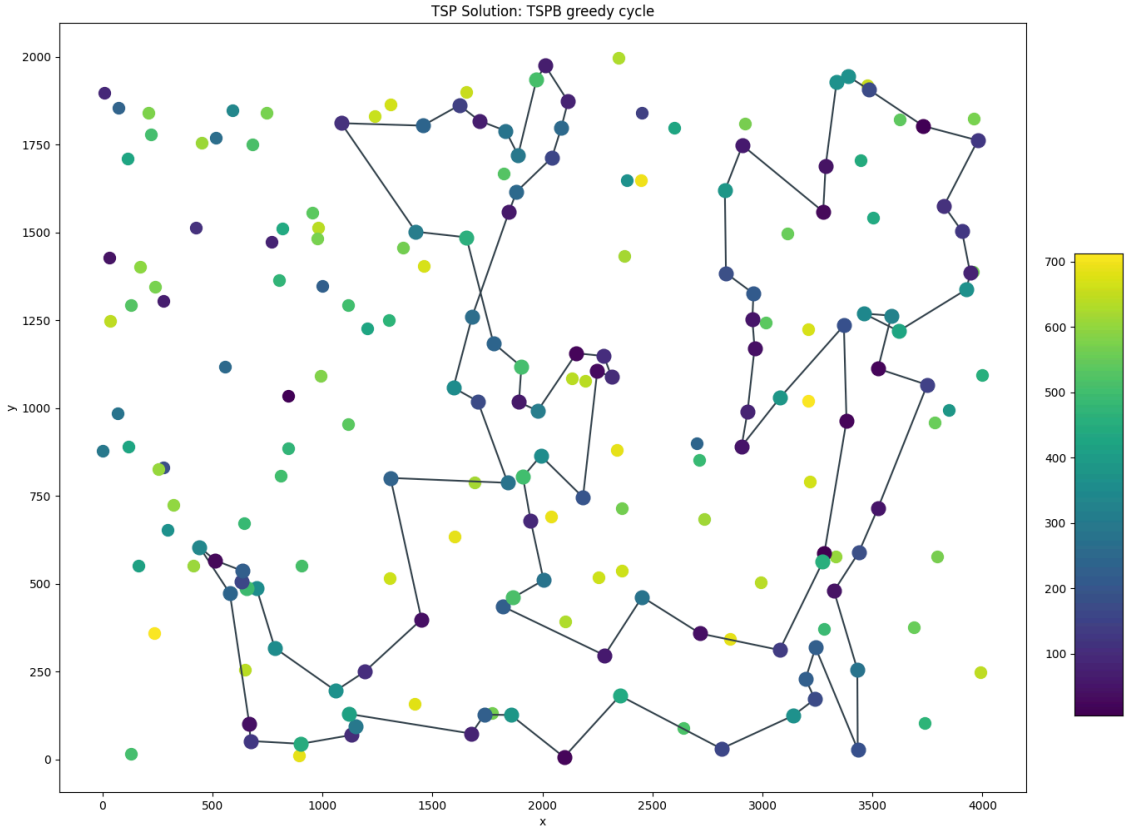


TSPB

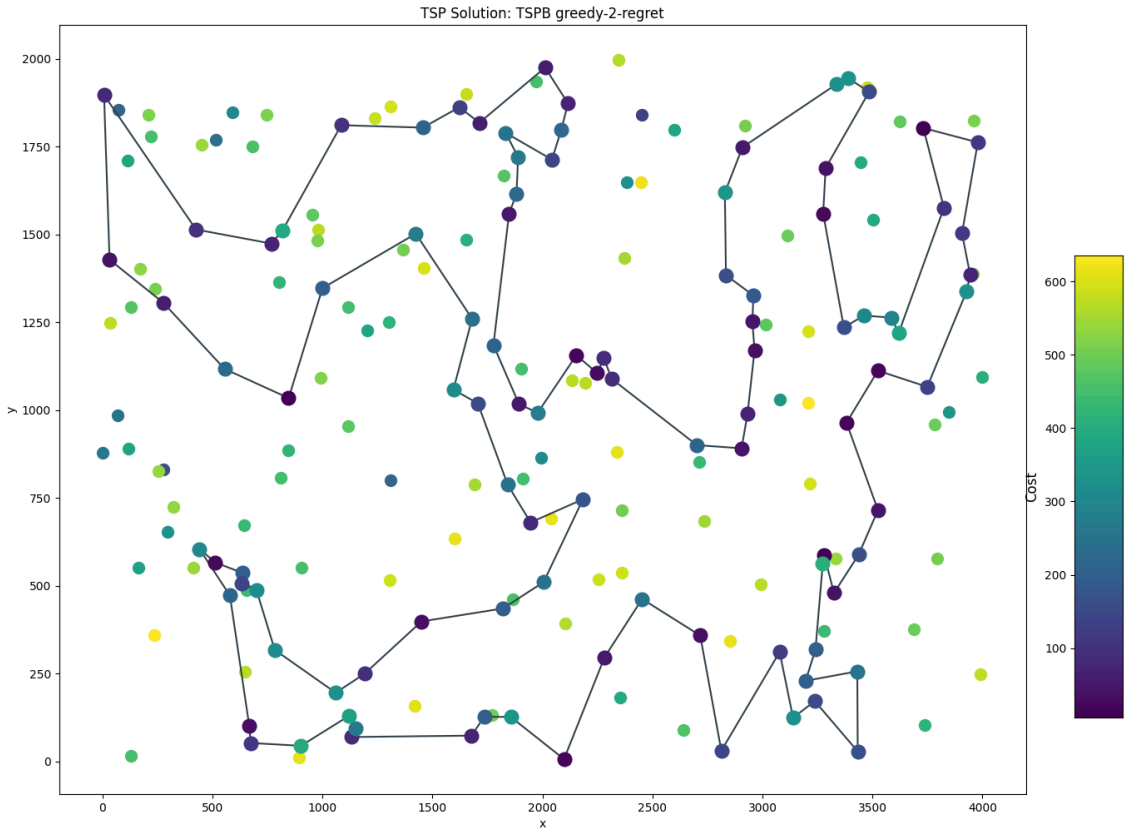
NN at any - [TSPB-nn-any-best.json]



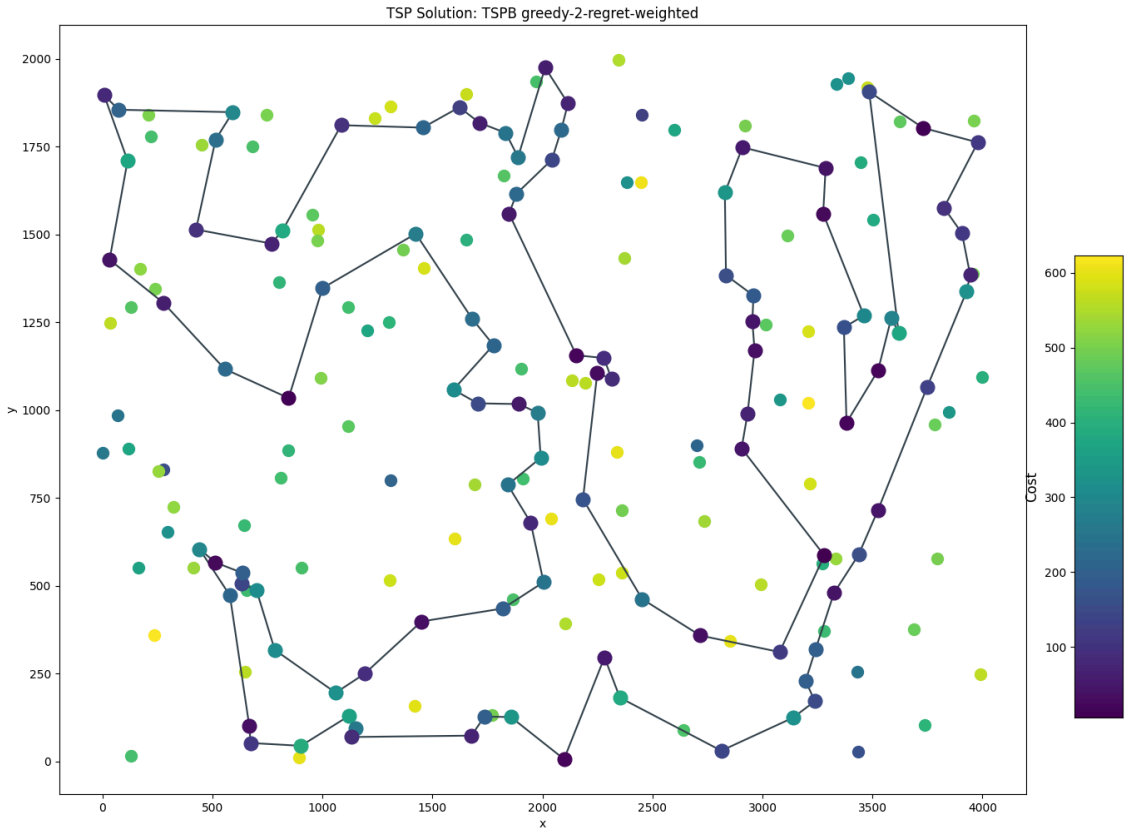
Greedy cycle - [TSPB-greedy-cycle-best.json]



Greedy 2-regret - [TSPB-greedy-2-regret-best.json]



Greedy 2-regret weighted obj - [TSPB-greedy-2-regret-weighted-best.json]



Conclusions

- Both regret heuristics
  - are significantly more complex than those from assignment 1
  - Yielded improvement in solution's quality over heuristic they were based on - greedy cycle
  - Obtained better results than NN at any on TSPA - which is logical, because TSPA has higher differences between node costs, leading to lots of possibilities where regret mechanism may save the algorithm from taking foul route
  - Obtained worse results than NN at any on TSPB - with smaller differences between node costs, exploitation seems more appropriate than worrying about regretting
- Weighted objective helps greedy 2-regret with exploitation, though its effectiveness depends on the particular set of data (noticeably worse solution on TSPB)
  - It's visible from the formed cycles that at times, it prevented the heuristic from taking less optimal route due to "worrying too much"