Evolutionary Computation - Assignment 2

Sebastian Chwilczyński 148248, Karol Cyganik 148250

October 22, 2023

1 Description of the problem

We are given a graph with n n nodes, each of which is described by it's x and y coordinates and a node cost. The goal is to select exactly $ceil(n/2)$ nodes that form a Hamiltonian cycle (closed path) and minimize

$$\sum_{e \in E} cost(e) + \sum_{v \in V} cost(v)$$

Where E is a set of selected edges, $cost(e)$ is Euclidean distance between two nodes rounded mathematically to integer value, V is set of selected nodes and $cost(v)$ is node cost.

2 Pseudocode

2.1 Greedy 2-regret - $O(n^3)$

We provide just a single algorithm, as by setting α to 1 we obtain classic unweighted 2-regret.

Algorithm 1 Algorithm that starts from given vertex and iteratively adds edge and node with lowest weighted 2-regret and cost always creating a cycle

Input: D: Distance matrix; C: List of costs; n: Starting node; α weight
Output: $Cycle$: List of $ceil(n/2)$ nodes selected with weighted 2-regret

$chaepest \leftarrow \arg\min_{i \neq n} D[n,i] + C[i]$
$cycle \leftarrow [n, cheapest]$
for $i = 1$ to $ceil(n/2)$ **do**
    $scores \leftarrow []$
    $bestPositions \leftarrow []$
    for each $v \notin cycle$ **do**
        $insertCosts \leftarrow []$
        for $pos = 0$ to $length(cycle)$ **do**
            $insertCosts[pos] \leftarrow D[cycle[pos], v] + D[v, cycle[pos+1]] - D[cycle[pos], cycle[pos+1]] + C[v]$
        end for
        $bestPos, secondBestPos \leftarrow$ indices of two smallest elements in the insertCosts array
        $regret \leftarrow insertCosts[secondBestPos] - insertCosts[bestPos]$
        $scores[v] \leftarrow \alpha * regret - (1-\alpha) * insertCosts[bestPos]$
        $bestPositions[v] \leftarrow bestPos$
    end for
    $bestVertex \leftarrow \arg\min scores$
    insert $bestVertex$ at position $bestPositions[bestVertex] + 1$ to the $cycle$
end for
return $cycle$

3 Results

Table 1: TSPA Results

Metric	random	nn	greedy	2 regret	W_2 regret
best	2338179.0	84471.0	75666.0	104829.0	74563.0
worst	289219.0	95013.0	80321.0	124764.0	78976.0
average	265672.095	87679.135	77064.415	116240.25	76341.56

Table 2: TSPB Results

Metric	random	nn	greedy	2 regret	W_2 regret
best	2338697.0	77448.0	68743.0	109774.0	70153.0
worst	299450.0	82631.0	76324.0	128550.0	77676.0
average	267823.05	79282.58	70735.35	118806.91	71801.35

Table 3: TSPC Results

Metric	random	nn	greedy	2 regret	W_2 regret
best	195689.0	56304.0	53226.0	65095.0	54126.0
worst	236233.0	63697.0	58876.0	73090.0	58288.0
average	215498.38	58872.68	55842.07	69013.725	55946.205

Table 4: TSPD Results

Metric	random	nn	greedy	2 regret	W_2 regret
best	196249.0	50335.0	50409.0	64682.0	49165.0
worst	241897.0	59846.0	60964.0	74903.0	59416.0
average	220321.22	54290.68	54838.01	70442.125	53691.48

4 Code

Implementation of algorithms and visualizations is available here

5 Conclusions

Not weighted 2-regret is worse then every non-random algorithm on all TSP instances. It is particularly weak for the first two. Intuitively this is caused by three facts:

- In this problem we don't need to add every node to the cycle.
- Big regret doesn't mean small cost.
- Costs and regrets changes dynamically with addition of every node to the cycle.

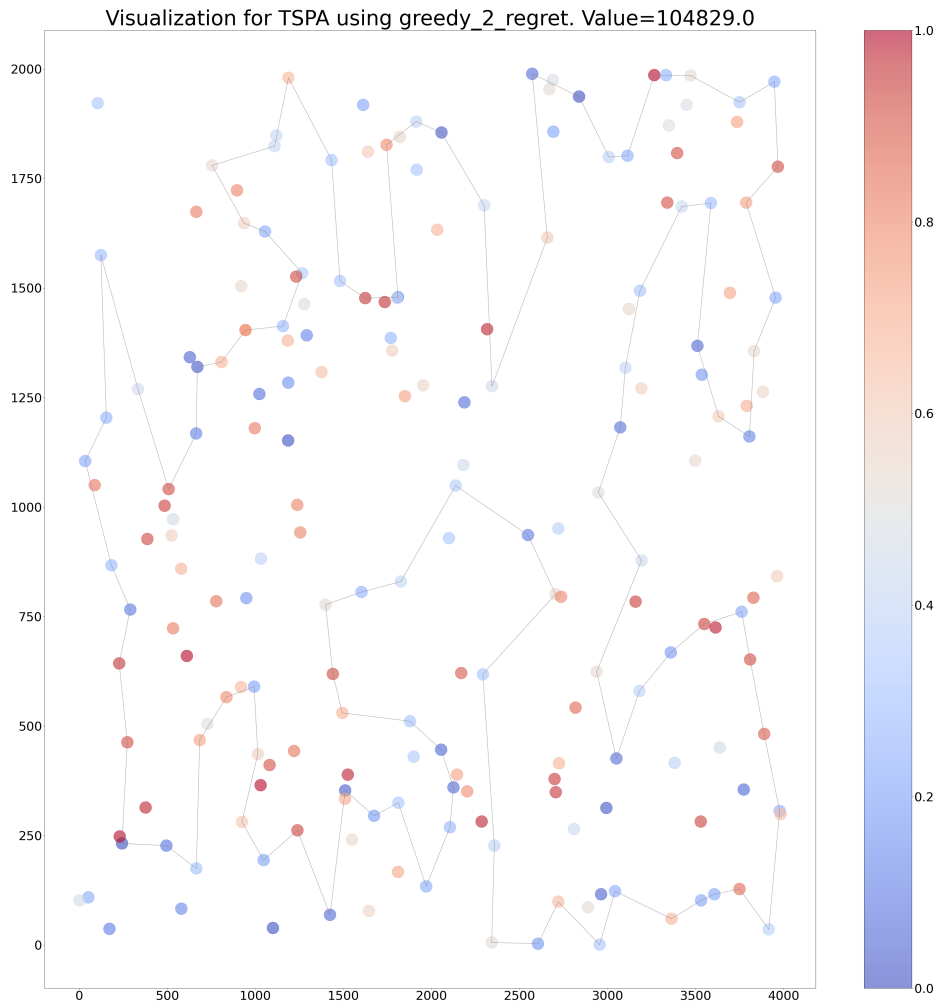

Figure 1: TSPA: 2-regret

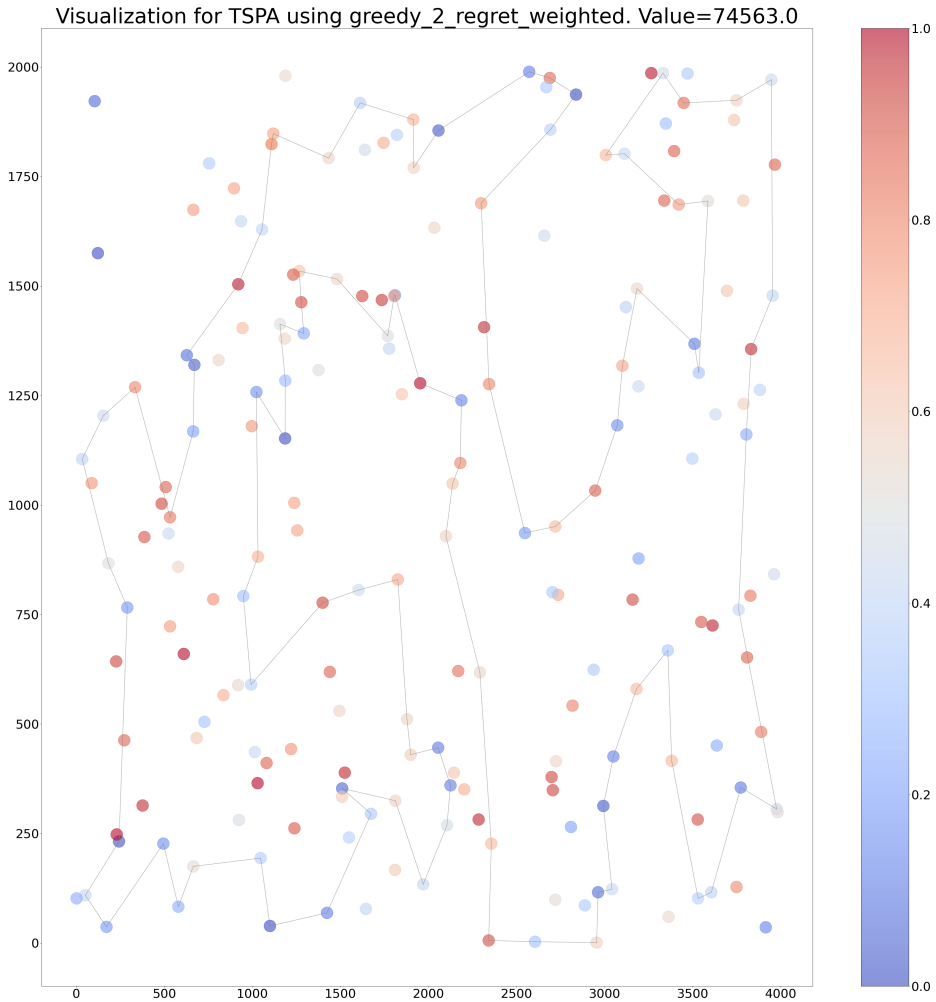

Figure 2: TSPA: Weighted 2-regret

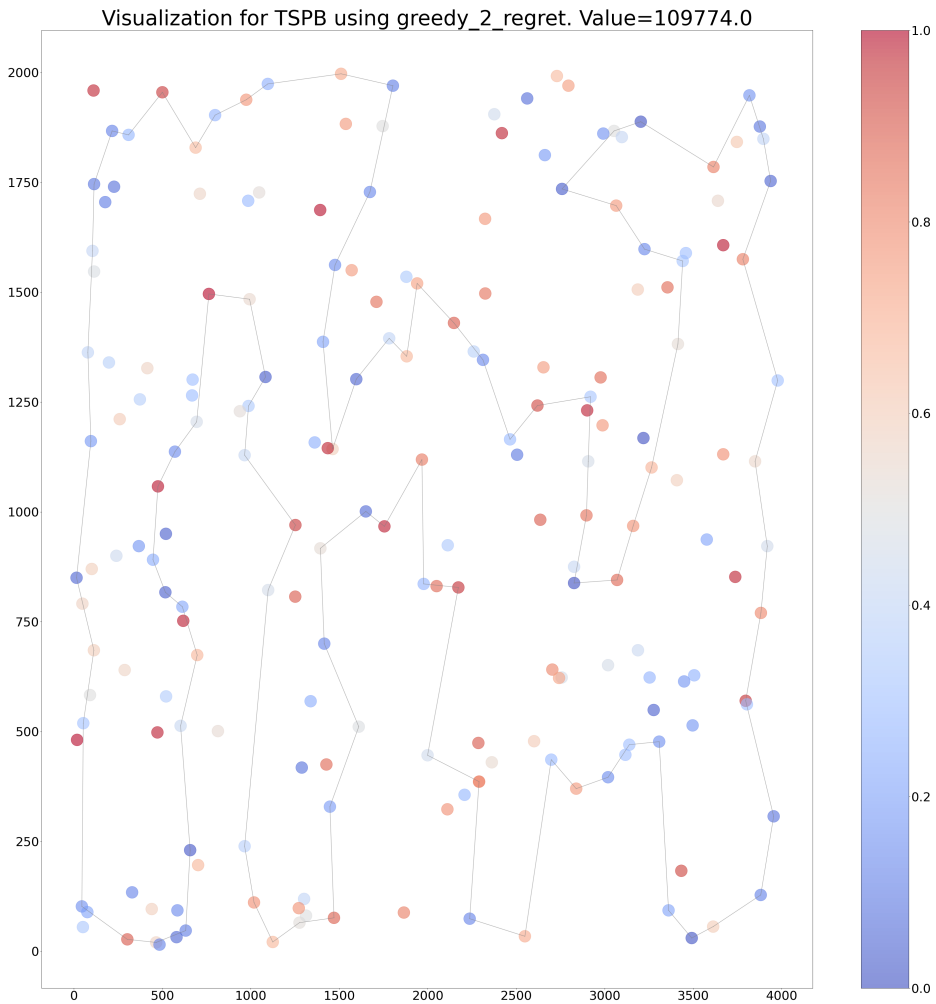

Figure 3: TSPB: 2-regret

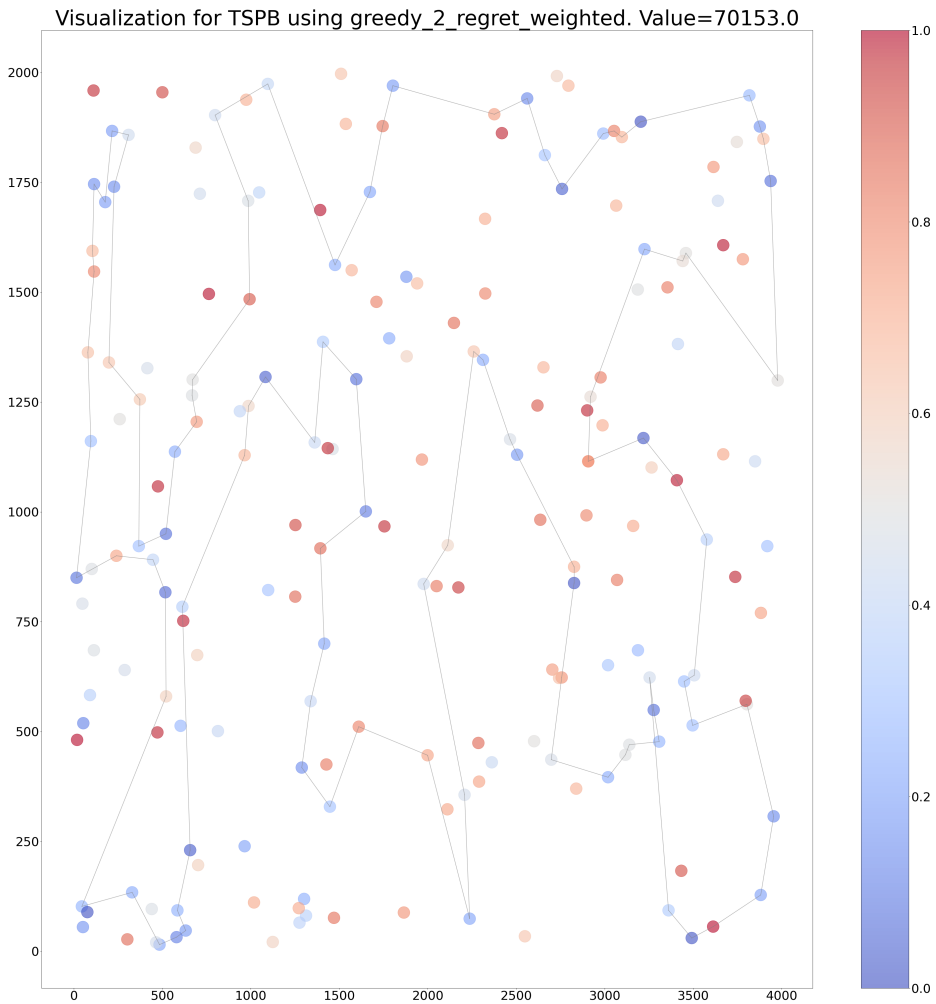

Figure 4: TSPB: Weighted 2-regret

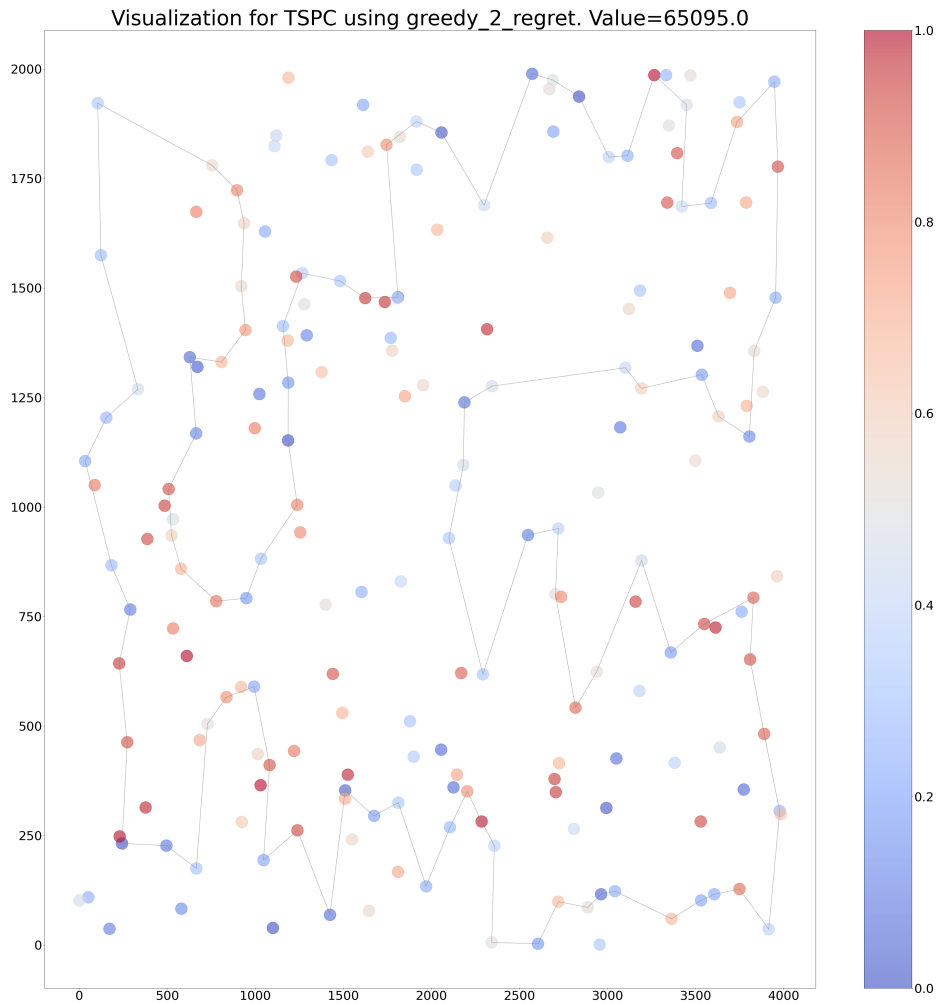

Figure 5: TSPC: 2-regret

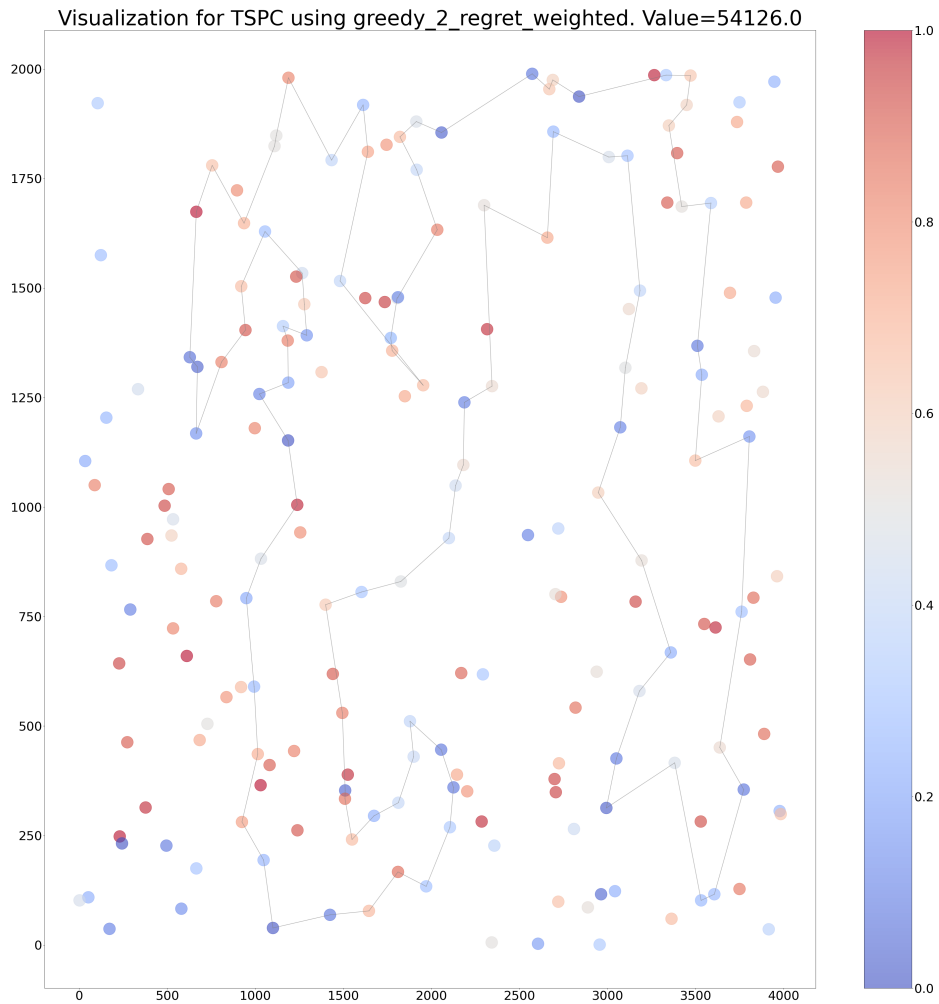

Figure 6: TSPC: Weighted 2-regret

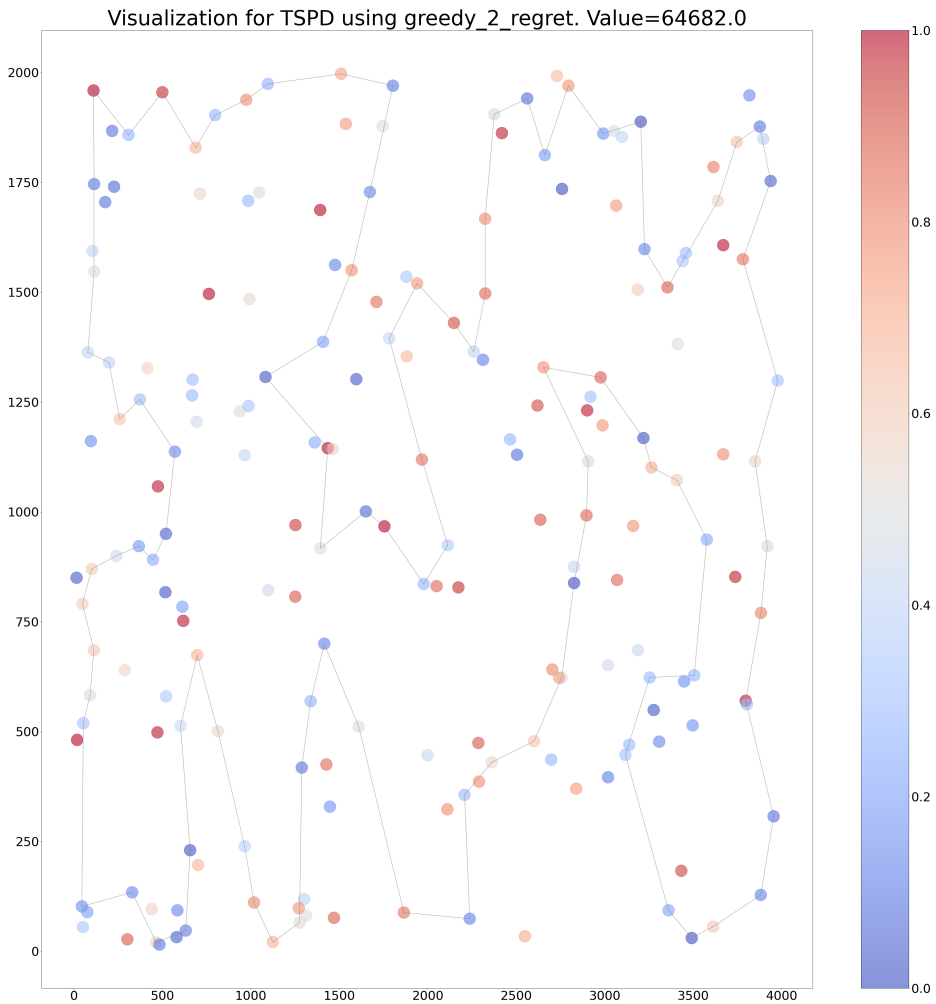

Figure 7: TSPD: 2-regret

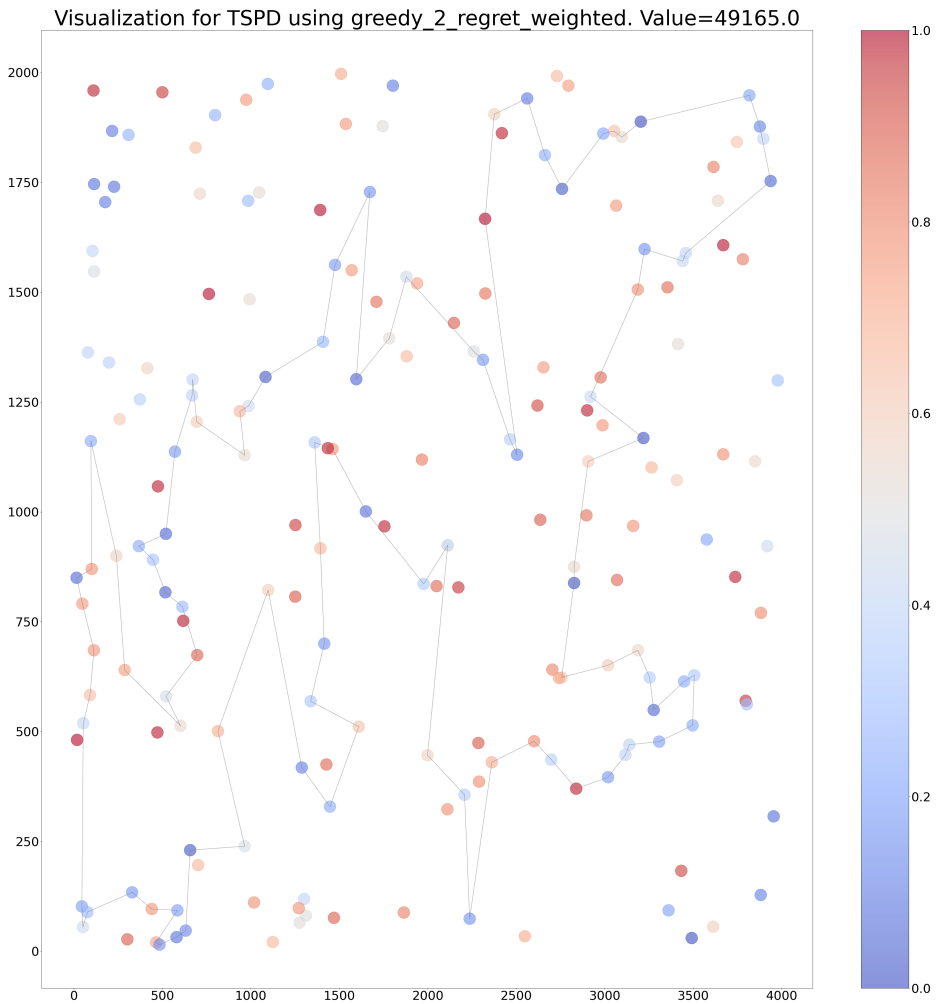

Figure 8: TSPD: Weighted 2-regret

Weighted 2-regret shows comparable performance to greedy cycle method. It is even better for A and D instances. It just allows to take more aspects into consideration which is always a good idea. Moreover, one can perform search over weights space to find optimal ones for every instance. Visually both methods creates cycles similar to greedy cycle method as they all works in a pretty the same way, they just use different score functions.