Assignment 1: Greedy heuristics

Authors:
Szymon Skwarek, 145461
Antoni Klorek, 145454

Problem description

A set of nodes is described by position on a plane (x and y coordinates) and a cost. The goal is to form a Hamiltonian cycle with 50% of the nodes selected in such a way that minimises the total cost. The total cost is calculated as a sum of distances needed to be traversed in order to move from one node to the other and the cost associated with a visited node. The distance between two nodes is calculated as Euclidean distance rounded mathematically to an integer.
The approach to solving the problem will consist of three greedy heuristics methods:

- random solution,
- nearest neighbour method,
- greedy cycle method.

Data is given as a 3-column table where the first column indicates the x coordinate, the second column indicates the y coordinate and the third column indicates the node's cost. Each table row corresponds to a single node. In this task, there are two sources of data - two CSV files, each consisting of 200 rows (200 nodes).

For each method, the solution will be calculated starting from each node, so in all for each data source and for each method, 200 solutions will be calculated.

Algorithms

Random solution

In a random solution, the next node is chosen randomly from the remaining nodes. When 50% of nodes will be chosen, the path from the last node to the first will be generated to close the cycle. 200 solutions were generated starting from each of the 200 given nodes.

Pseudocode

The presented below pseudocode describes a function that finds a single random solution to a TSP problem starting from a given node.

```
FUNCTION random_solution(starting_node, data, distances)
    Input: index of a starting node, dictionary with data of all nodes,
           array of distances between each node
    Output: total cost of a found cycle, nodes chosen to the cycle

    SET cost = 0
    SET chosen_nodes = []
    SET lst = list of numbers from 0 to 200 excluded
    pop starting node index from lst
    shuffle lst
    SET i = 0

    FOR each number in first 99 numbers from lst
        SET start_node = ith node index from chosen_nodes
        add cost of start_node to cost
        SET dist = distance between start_node and numberth node
        add dist to cost
        add number to chosen_nodes
        increment i by 1
    ENDFOR

    add cost of the last chosen node to the cost
    add to the cost a distance between the last chosen node and the
        first chosen node

    RETURN cost, chosen_nodes
```

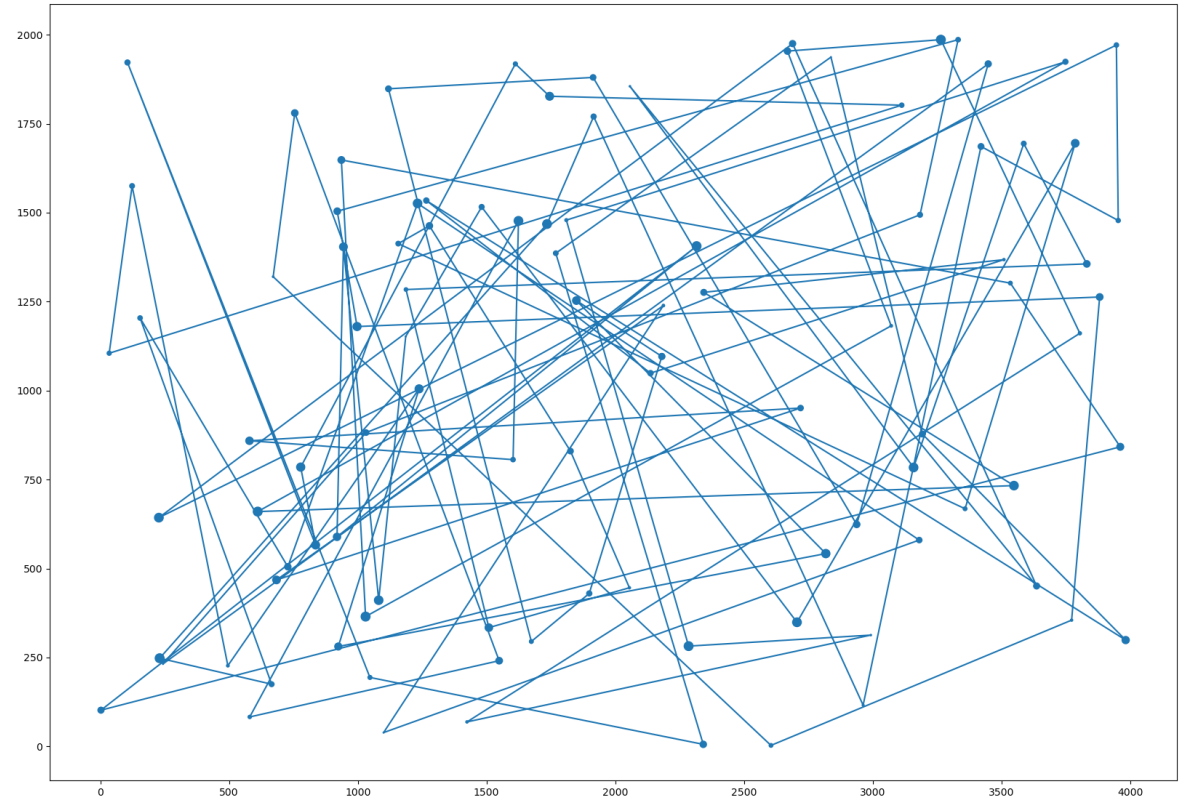
TSPA.csv

Results

The below table presents the results for the random method on data from the TSPA.csv file. The minimum cost is the best one obtained from 200 solutions, the maximum cost is the worst one from 200 solutions and the average cost is the average cost from all 200 resulting cycles.

Minimum cost	235828
Maximum cost	286386
Average cost	263332

Visualisation



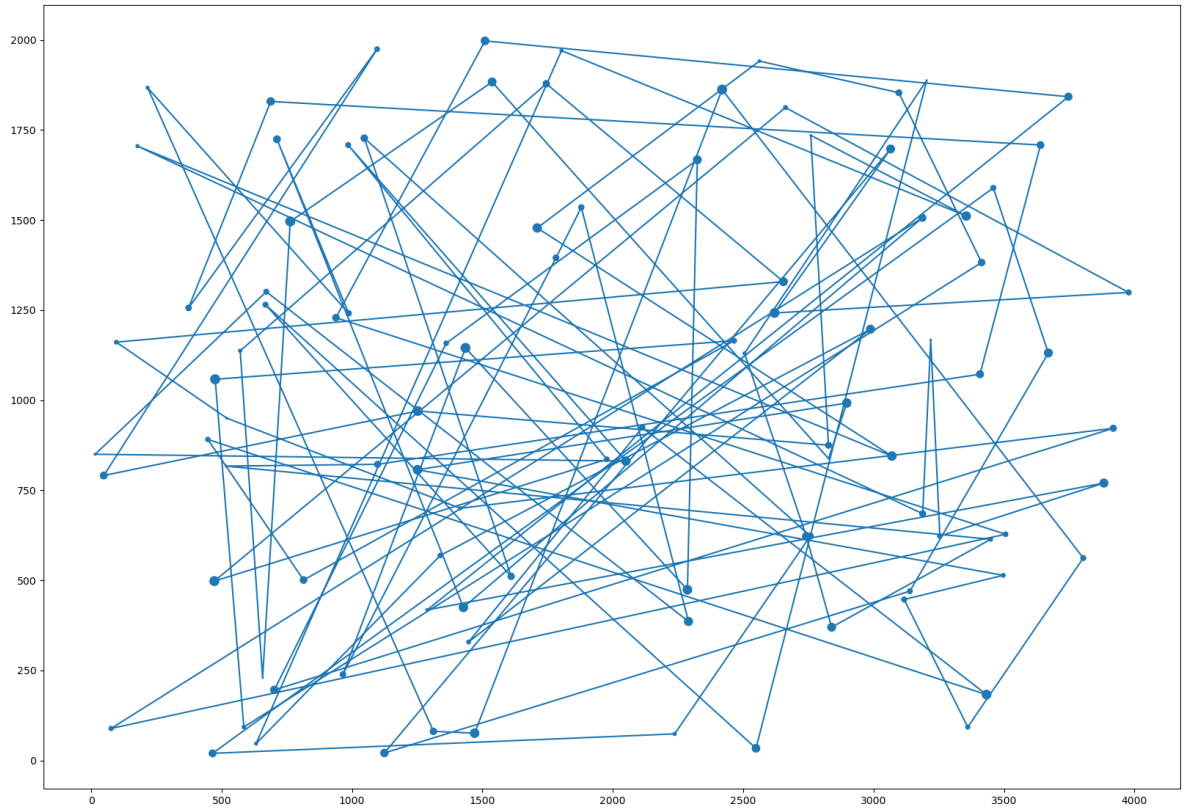
TSPB.csv

Results

The below table presents the results for the random method on data from the TSPB.csv file. The minimum cost is the best one obtained from 200 solutions, the maximum cost is the worst one from 200 solutions and the average cost is the average cost from all 200 resulting cycles.

Minimum cost	241057
Maximum cost	291425
Average cost	265937

Visualisation



Nearest neighbor solution

In the nearest neighbor solution, the process starts from some chosen starting node. Then, the next node is picked as the one with the smallest distance + cost from the last chosen node already in the cycle. The procedure repeats until 50% of the nodes are chosen.

Pseudocode

The presented below pseudocode describes a function that finds a single nearest neighbor solution to a TSP problem starting from a given node.

```
FUNCTION nearest_neighbor(starting_node, data, all_costs)
    Input: index of a starting node, dictionary with data of all nodes,
        array of distances plus end node cost between each node
    Output: total cost of a found cycle, nodes chosen to the cycle

    change distances in all_costs array from all nodes to starting_node
        to infinities
    SET cycle = []
    add starting_node to a cycle
    SET cost = cost of the starting_node
    SET curr_id = starting_node

    FOR _ in range(99)
        SET min_index = index of a node with the smallest distance plus
            its cost to the current node
        SET min_value = value of the distance between min_index node
            and curr_id node plus cost of min_index node
        change entries in all_costs array from all nodes to the
            chosen node to infinities
        add value of the distance to the cost
        append min_index node to the cycle
        SET curr_id = min_index
    ENDFOR

    RETURN cost, cycle
```

TSPA.csv

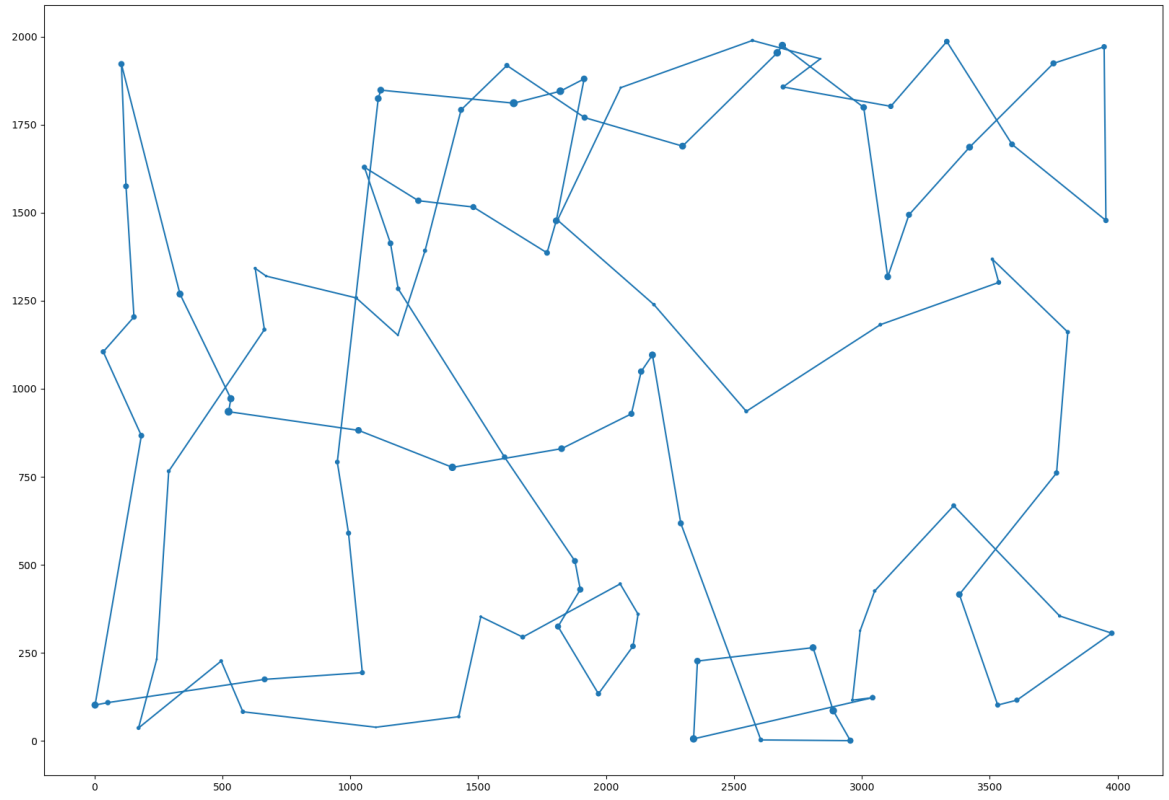
Results

The below table presents the results for the nearest neighbor method on data from the TSPA.csv file. The minimum cost is the best one obtained from 200 solutions, the maximum cost is the worst one from 200 solutions and the average cost is the average cost from all 200 resulting cycles.

Minimum cost	84471
Maximum cost	95013

Average cost	87679

Visualisation



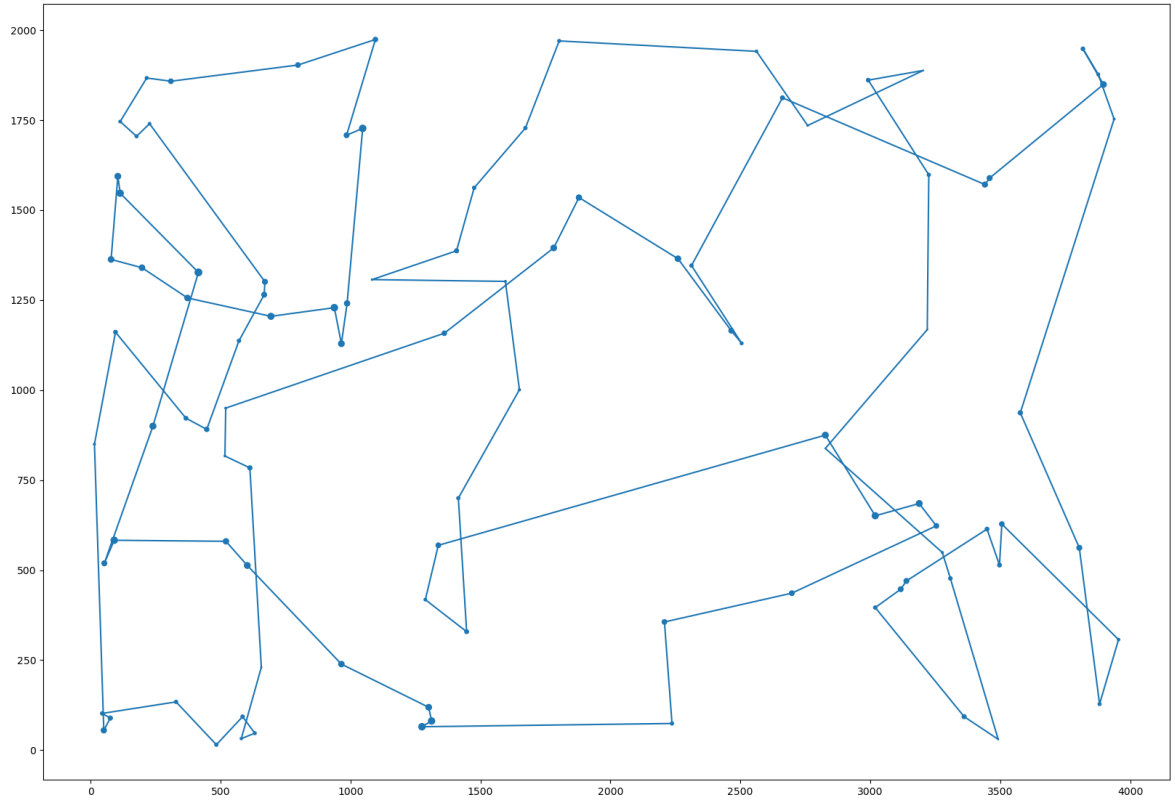
TSPB.csv

Results

The below table presents the results for the nearest neighbor method on data from the TSPB.csv file. The minimum cost is the best one obtained from 200 solutions, the maximum cost is the worst one from 200 solutions and the average cost is the average cost from all 200 resulting cycles.

Minimum cost	77448
Maximum cost	82631
Average cost	79283

Visualisation



Cycle greedy solution

The greedy cycle process starts from some starting node. The second node is chosen as the one placed the nearest from the starting node and an incomplete cycle from these two nodes is constructed. Then, the next node is chosen as the one which will cause the smallest increase in the cycle cost. The procedure repeats until 50% of nodes are chosen.

Pseudocode

The presented below pseudocode describes a function that finds a single cycle greedy solution to a TSP problem starting from a given node.

```
FUNCTION cycle_greedy(first_node, nearest_node, data,
all_distances_with_costs)
    input: starting node, second node - the nearest node to the
        starting node, dictionary with data of all nodes,
        array of distances between each node plus end node cost
    output: total cost of a found cycle, nodes chosen to the cycle

    cost = 0
    add to cost costs of the first and second node
    add to cost two times the distance between first and second node
    cycle = []
    add first and second node to the cycle
```

```
chosen_nodes = {first_node, nearest_node}

FOR _ in range(98)
    min_insertion_dist = infinity
    min_insert_id = None
    min_new_node = None
    node_candidates = set of available nodes
    FOR EACH edge_id, EACH edge in enumerate(zip(cycle[:-1], cycle[1:]))
        i, j = edge
        FOR EACH new_node in a set of node_candidates:
            edge0dist = distance from i node to new_node
                plus new_node cost
            edge1dist = distance from i node to new_node
                plus new_node cost

            insertion_dist = edge0dist + edge1dist - distance
                between i and j node
            IF insertion_dist is smaller than min_insertion_dist:
                min_insertion_dist = insertion_dist
                min_insert_id = edge_id + 1
                min_new_node = new_node
            ENDIF
        ENDFOR
    add min_new_node to chosen_nodes
    insert min_new_node in cycle at position min_insert_id
    add to cost min_insertion_dist
    add to cost cost of min_new_node
ENDFOR
RETURN cost, cycle
```

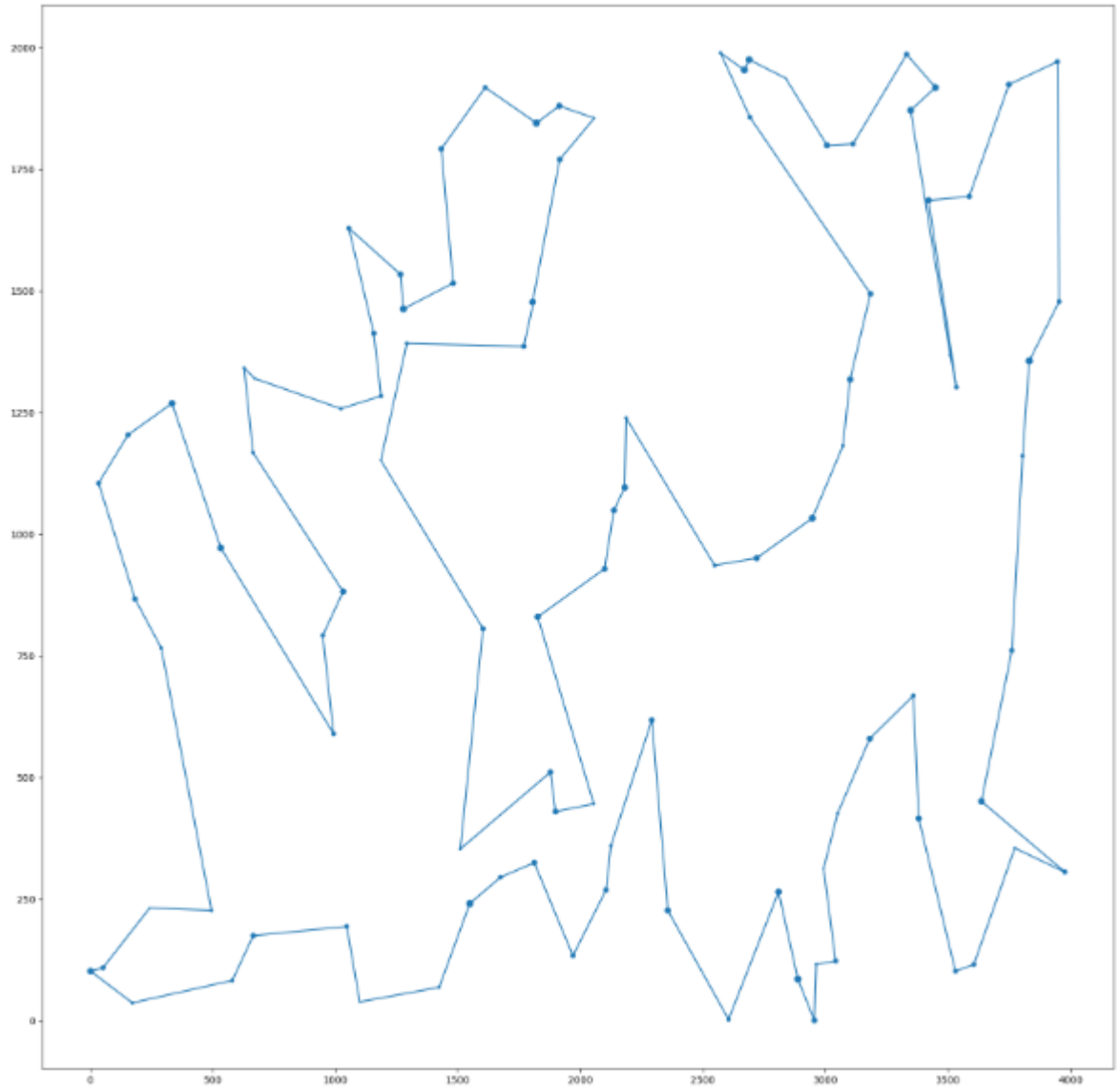
TSPA.csv

Results

The below table presents the results for the nearest cycle greedy method on data from the TSPA.csv file. The minimum cost is the best one obtained from 200 solutions, the maximum cost is the worst one from 200 solutions and the average cost is the average cost from all 200 resulting cycles.

Minimum cost	75755
Maximum cost	80116
Average cost	77123

Visualisation



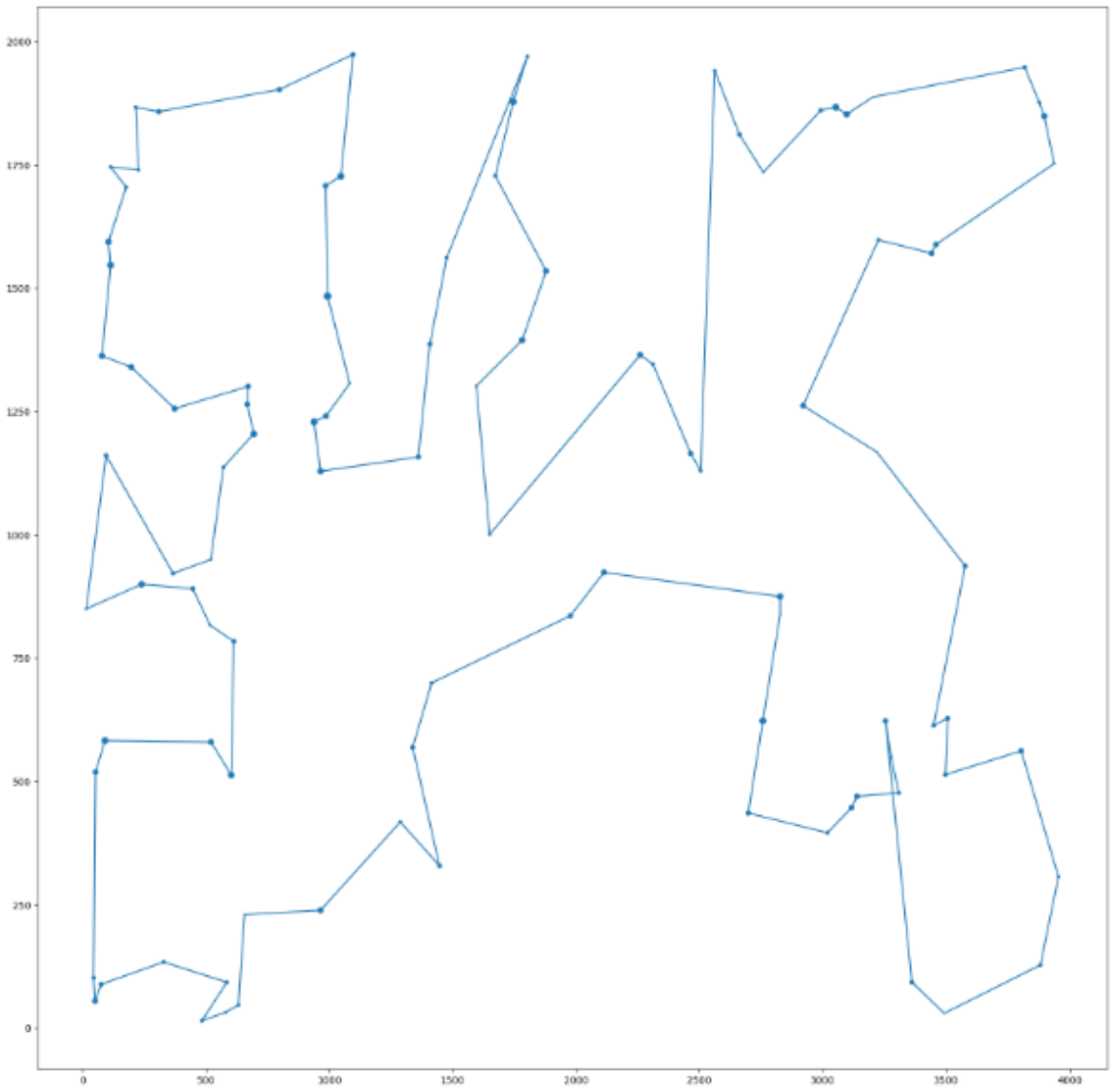
TSPB.csv

Results

The below table presents the results for the nearest cycle greedy method on data from the TSPB.csv file. The minimum cost is the best one obtained from 200 solutions, the maximum cost is the worst one from 200 solutions and the average cost is the average cost from all 200 resulting cycles.

Minimum cost	68651
Maximum cost	76499
Average cost	70783

Visualisation



Source code

The source code for the all presented methods can be found in the github repository: https://github.com/AntKlo/Evolutionary_computation/tree/main/Assignment1-GreedyHeuristics

Conclusions

The random solution performs pretty badly in this problem. Selecting the next nodes at random in almost all trials finds a very long and a costly cycle. The only benefit of a random solution is its very short running time.

The nearest neighbor solution performed much better than a random method. The algorithm finds the next node as the one positioned nearest to the last chosen node already in the cylce. However, there is a place for a lot of improvements because even by looking at a plot one can see that the solution is not optimal. The running time is much longer than a random solution but the resulting cycle is incomparably better.
The greedy cycle method found the shortest cycle of the three analysed methods. However, its processing time is the longest. The method needs to analyse each point and each edge to choose the best option for enlarging the cycle. Inserting a new node and creating edges to it has to be the least costly option from all available choices.