POZNAN UNIVERSITY OF TECHNOLOGY

Assignment 2: Greedy Regret Heuristics

Mateusz Tabaszewski 151945 Bartłomiej Pukacki 151942

Source code:
https://github.com/MatTheTab/Evolutionary-Computation/blob/main/Assignment%202%20Regret%20heuristics/Assignment_2_b.ipynb

1. Description of the Problem

The report describes the steps undertaken to solve the problem for assignment 2. The problem solved in this assignment is analogous to the one present in assignment 1 with the exception of the methods used in this assignment. Instead of utilizing the greedy algorithms like previously, in this version, we are using greedy algorithms with regret and a weighted greedy algorithm with an equal split of weights for the greedy and regret algorithm. All described algorithms are based on the greedy cycle method. Just as in the previous assignment, the NP-hard problem described in this assignment is a slight redefinition of the original TSP, as in this version the cost is defined as the sum of the total length of the cycle and the sum of the weights of all the nodes included in the cycle. The cost should be minimized. Furthermore, in this version of the problem, only 50% of all nodes need to be visited in total. The formal definition of this new problem is shown below and remains constant between assignments:

Decision Variables:

$x_{ij} \in \{0,\ 1\}$ - included edges

$y_i \in \{0,\ 1\}$ - visited nodes

Objective Function:

$$min(\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} + \sum_{i=1}^{n} w_i y_i)$$

sb. t.

$\sum_{j\in V} x_{ij} = 2y_i \ , \forall i\in V$; *where V is a set of all vertices*

$$\sum_{i=1}^{n} y_i \geq \frac{n}{2}$$

$x_{ij} \in \{0,\ 1\}, \forall i, j \in V$

$y_i \in \{0,\ 1\},\ \forall i \in V$

As mentioned in the previous report, two instances of the problem - TSPA.csv and TSPB.csv - were presented for testing and visualization of the results. All algorithms described in the further sections were tested on both algorithms. Figures 1 and 2 show the nodes' x and y locations from the TSPA and TSPB instances.

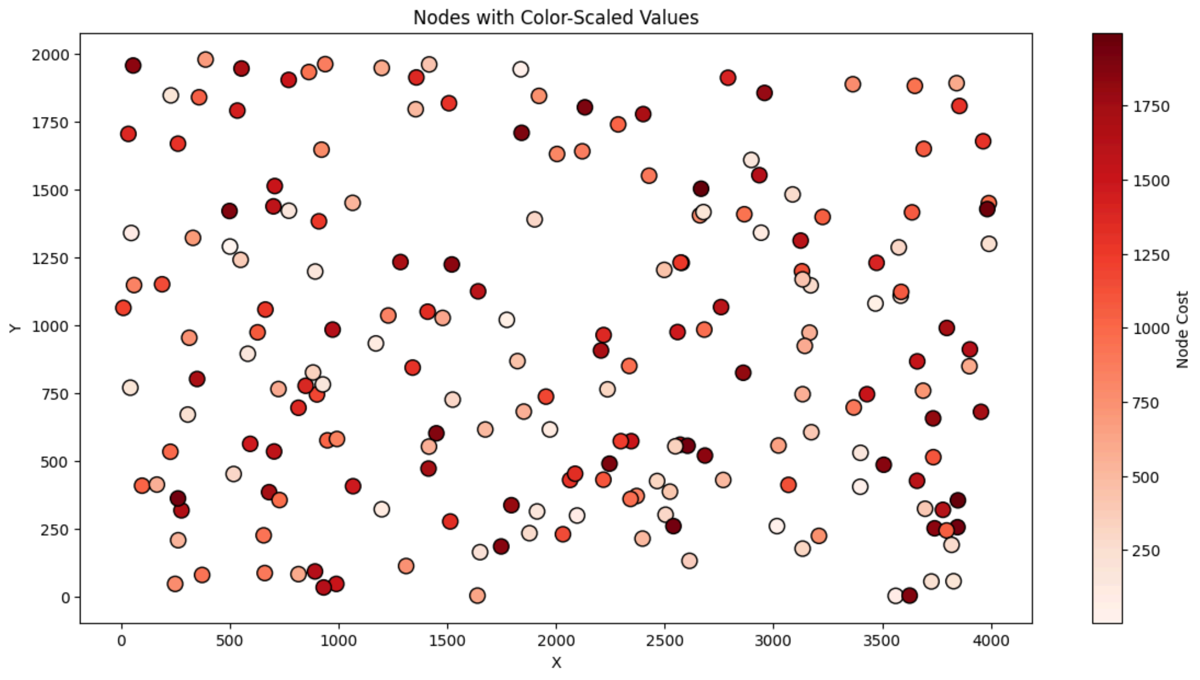

Fig 1. Visualization of the TSPA problem instance, each node's x and y locations on the plot correspond to their given x and y locations and the color intensity signifies the weight/cost of each node. The total length of the cycle and the sum of node weights should be minimized.

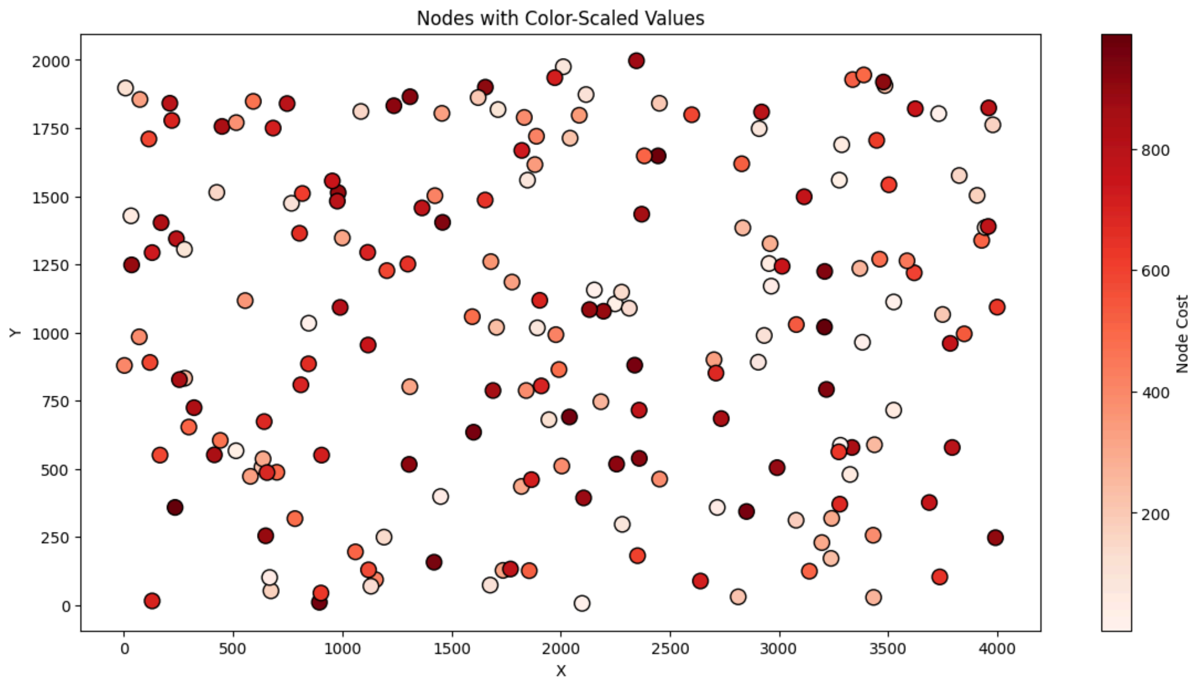

Fig 2. Visualization of the TSPB problem instance, each node's x and y locations on the plot correspond to their given x and y locations and the color intensity signifies the weight/cost of each node. The total length of the cycle and the sum of node weights should be minimized.

2. Algorithms

The assignment was completed with the use of the following two algorithms: a greedy cycle algorithm with 2-regret and a weighted greedy cycle algorithm with 2-regret. Both algorithms were presented in the form of pseudocode for the sake of clarity. The means of calculating the scores of solutions is the same as before, so there is no need to repeat it as such the pseudocode for that has been omitted.

The greedy cycle algorithm with 2-regret also called the greedy 2-regret cycle algorithm from now it will just be referred to by us as **the greedy cycle regret algorithm**. The algorithm allows the user to define the starting node. Then the second node must be chosen. By default, it is the closest node to the starting one, as is defined using both distance and node cost, i.e. the node that is the closest while also having the smallest weight/cost. Then, the third node must be chosen. Since currently switching the placement of the nodes does not allow for regret calculating we simply choose the node with the smallest distance to the remaining nodes which also has the smallest weight. Now, with three chosen nodes, the proper algorithm starts with the loop which goes over all nodes trying to find the best and second-best placements for each node. The score for both node placements is calculated and based on that, so is regret. After going through all the nodes and positions, the node with the highest regret is chosen for its best position. The algorithm continues until half of all nodes are selected. The complexity of this algorithm is O(n^3).

```
FUNCTION greedy_regret_cycle(distance_matrix, weights, start_node):
    INPUT:
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node
        start_node - starting node for the cycle

    second_node ← node with smallest distance to start_node and smallest weight
    third_node ← node closest to start_node and second_node with the smallest weight
    solution ← [start_node, second_node, third_node]
    non_cycle_nodes ← list of all nodes except start_node, second_node, third_node
    num_iterations ← ((number of all nodes)//2 - 3)
    FOR iteration in num_iterations:
        best_regret ← -INFINITY
        FOR node in non_cycle_nodes:
            best_score ← INFINITY
            second_best_score ← INFINITY
            FOR insert_location in RANGE(length of solution):
                temp_score ← UPDATE score if we INSERT node in solution AT insert_location
                IF temp_score is better than best_score:
                    second_best_score ← best_score
                    second_best_node ← best_node
                    best_score ← temp_score
                    best_node ← node
                    best_position ← insert_location
                ELSE IF temp_score is better than second_best_score:
                    second_best_score ← temp_score
                    second_best_node ← node
            regret ← best_score - second_best_score
            regret_location ← best_position
            IF regret is better than best_regret:
                best_regret ← regret
                best_node ← node
                best_location ← regret_location
        INSERT into solution best_node into best_location
        UPDATE score
        REMOVE from non_cycle_nodes best_node
    RETURN solution, score
```

Another implemented algorithm was the greedy heuristics with a weighted sum criterion – 2-regret + best change of the objective function which for the sake of simplicity will be referred to as **the weighted greedy regret cycle algorithm**. This method is similar to the previous one with the exception that both regret and distance are taken into account when calculating the best node to be selected. Similarly to before, the starting node is given, and the second one is chosen based on the distance and the regret, which is calculated as a score obtained by choosing this node now and the second closest distance to this node in the future. The absolute values of weights in the algorithm are equal and equal to 0.5, but the regret's weight is negative, so we end up minimizing the total summed score. Each subsequent node is selected by choosing the node with the smallest summed score between the regret of putting the node in a particular split in the cycle and the total score increase related to choosing the node. The complexity of this algorithm is O(n^3).

```
FUNCTION greedy_regret_cycle_weighted(distance_matrix, weights, start_node,
regret_weight):
    INPUT:
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node
        start_node - starting node for the cycle
        regret_weight - weight of the regret in the algorithm, the value should
be smaller than 1 and negative, for example, the default is -0.5

    second_node ← node with smallest distance to start_node and smallest weight
    third_node ← node closest to start_node and second_node with the smallest
weight
    solution ← [start_node, second_node, third_node]
    non_cycle_nodes ← list of all nodes except start_node, second_node,
third_node
    num_iterations ← ((number of all nodes)//2 - 3)
    FOR iteration in num_iterations:
        best_weighted_score ← INFINITY
        FOR node in non_cycle_nodes:
            best_score ← INFINITY
            second_best_score ← INFINITY
            FOR insert_location in RANGE(length of solution):
                temp_score ← UPDATE score if we INSERT node in solution AT
insert_location
                IF temp_score is better than best_score:
                    second_best_score ← best_score
                    second_best_node ← best_node
                    best_score ← temp_score
                    best_node ← node
                    best_position ← insert_location
                ELSE IF temp_score is better than second_best_score:
                    second_best_score ← temp_score
                    second_best_node ← node
            regret ← best_score - second_best_score
            regret_location ← best_position
            score_difference ← best_score - score
            weighted_score ← ((1 + regret_weight) * score_difference) +
(regret_weight * regret)
            IF weighted_score is better than best_weighted_score :
                best_weighted_score ← weighted_score
                best_node ← node
                best_location ← regret_location
        INSERT into solution best_node into best_location
        UPDATE score
        REMOVE from non_cycle_nodes best_node
    RETURN solution, score
```

3. Experiments

To quantify the performance of greedy methods, each algorithm was run 200 times on each of the 200 nodes as starting points on both instances available and the solutions and scores generated were collected. Additionally, the random algorithm was run 200 times to compare the results.

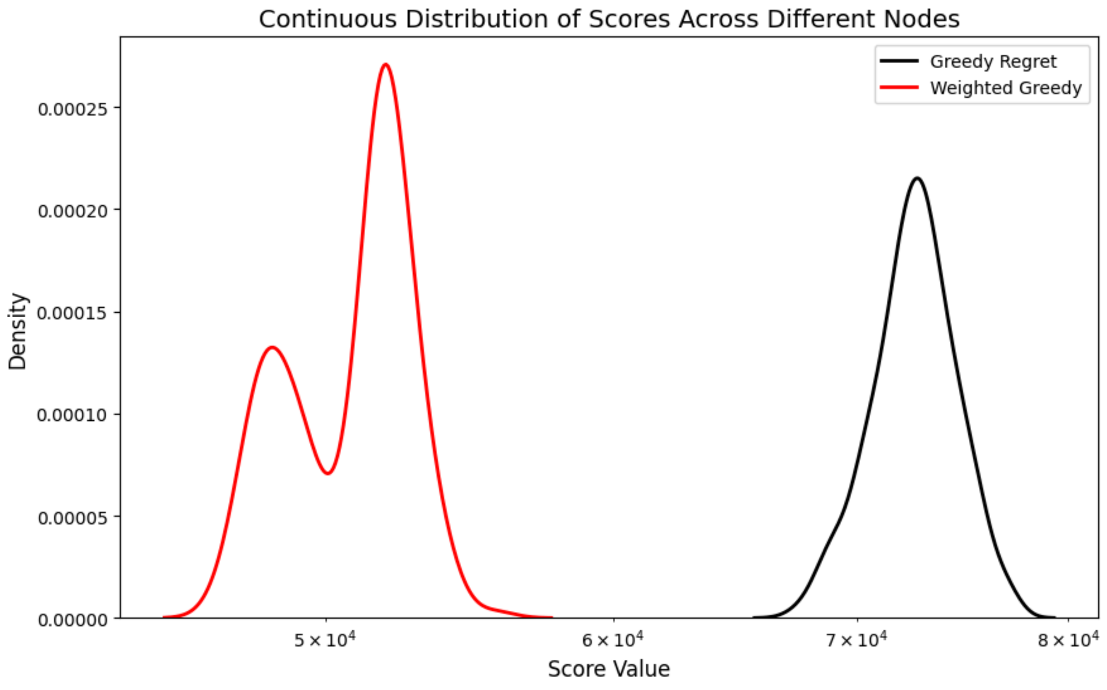

Fig 3. Visualization of the distribution of scores achieved by each algorithm, run 200 on each node as a starting point, on the TSPB problem instance.

Problem instance	Algorithm	Min Score	Average Score	Max Score
TSPA	Random	223539	264301.22	308435
	Nearest Neighbor Closest	83182	85108.51	89433
	Nearest Neighbor All	78896	80974.365	82368
	Greedy Cycle	71488	72605.865	74350
	Greedy Regret Cycle	105852	115630.16	123171
	Weighted Greedy Regret Cycle	71108	72132.785	73395

TSPB	Random	179796	213397.41	253866
	Nearest Neighbor Closest	52319	54390.43	59030
	Nearest Neighbor All	52992	55015.845	57460
	Greedy Cycle	48765	51344.64	57262
	Greedy Regret Cycle	67568	72656.19	77329
	Weighted Greedy Regret Cycle	47144	50882.28	55700

Table 1. Minimum, average and maximum scores achieved by each method on both problem instances.

The best scores achieved are visualized below.

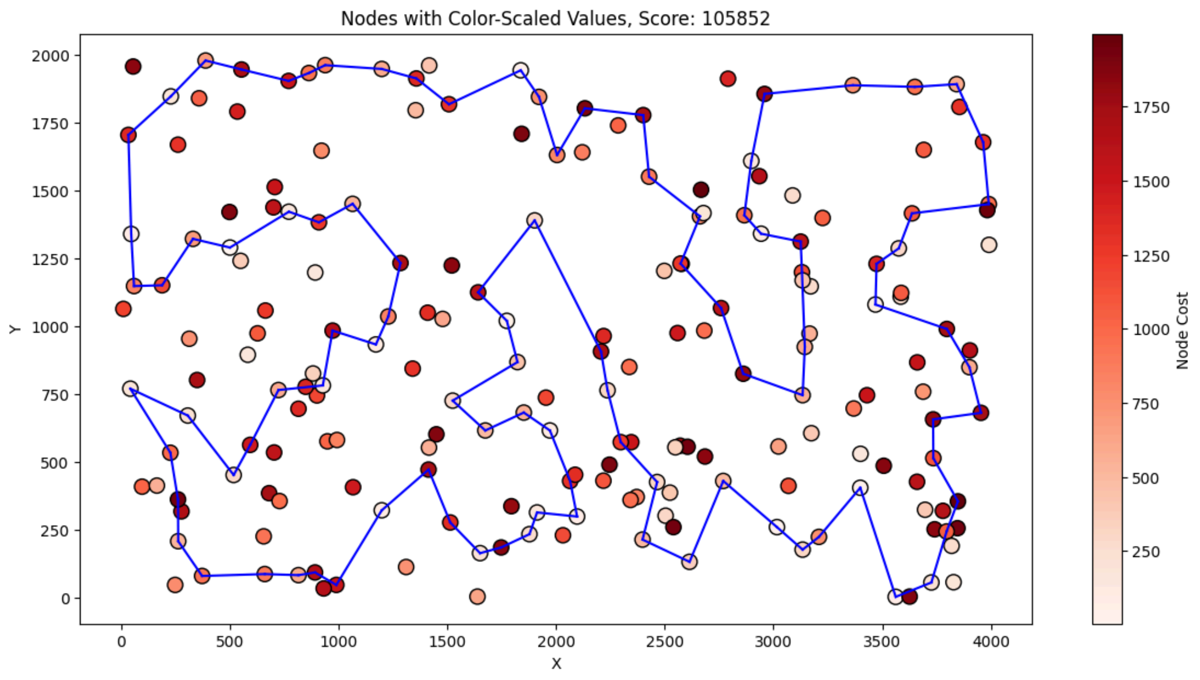

Fig 4. Visualization of the best solution found by the greedy regret cycle algorithm on the TSPA problem instance. Starting node: 93

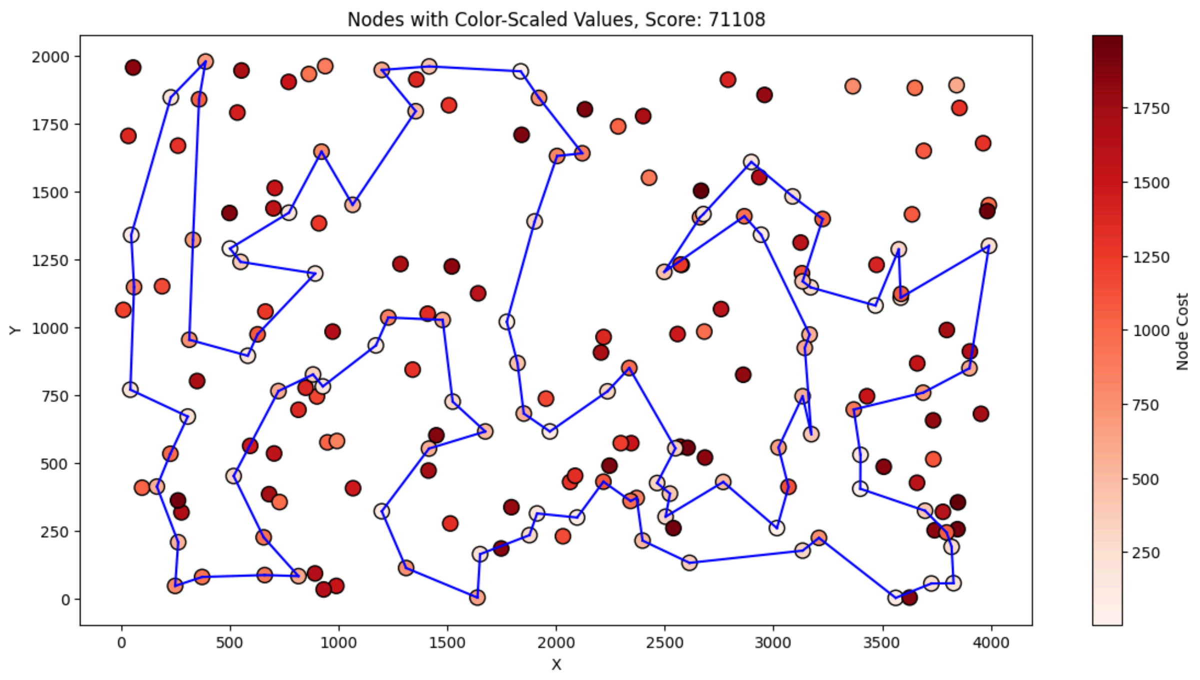

Fig 5. Visualization of the best solution found by the weighted greedy regret cycle algorithm on the TSPA problem instance. Starting node: 0

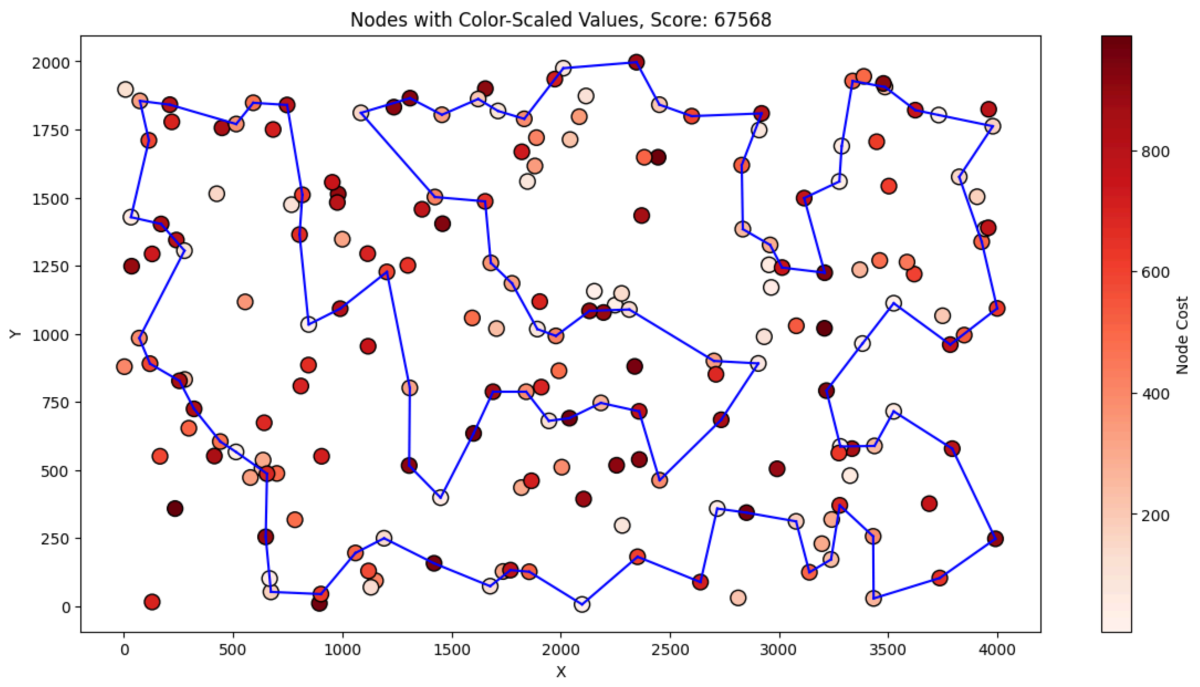

Fig 6. Visualization of the best solution found by the greedy regret cycle algorithm on the TSPB problem instance. Starting node: 60

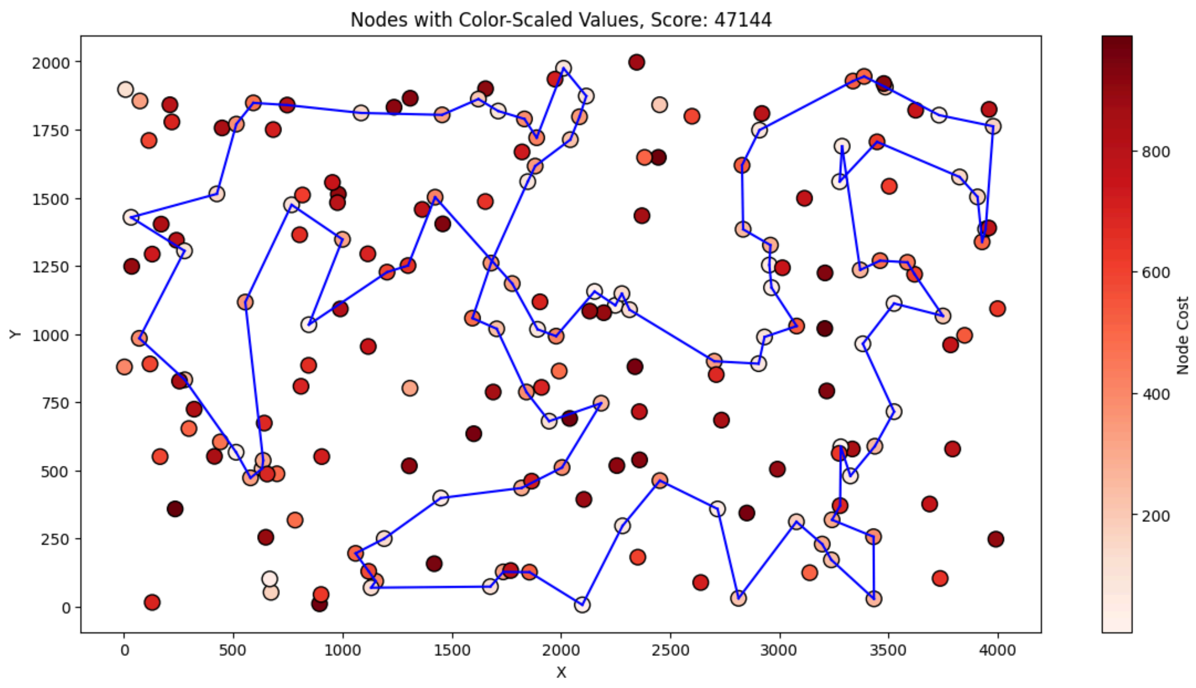

Fig 7. Visualization of the best solution found by the weighted greedy regret cycle algorithm on the TSPB problem instance. Starting node: 199

All best solutions were checked using the solution checker spreadsheet available on eKursy. The lists of node indices in the best solutions and their scores are presented in the table below.

Problem instance	Algorithm	Score	Solution
TSPA	Random	223539	14, 111, 63, 123, 89, 157, 168, 81, 148, 62, 94, 42, 134, 192, 65, 162, 19, 75, 127, 103, 136, 70, 3, 194, 167, 146, 52, 55, 170, 39, 172, 51, 27, 7, 121, 166, 46, 18, 105, 28, 163, 0, 30, 53, 190, 54, 96, 43, 137, 66, 80, 86, 4, 16, 56, 184, 97, 181, 24, 159, 128, 31, 196, 133, 10, 73, 45, 41, 118, 59, 82, 2, 100, 176, 72, 78, 197, 107, 174, 169, 185, 76, 17, 37, 8, 11, 117, 77, 74, 40, 154, 140, 114, 132, 49, 32, 92, 182, 38, 151
	Nearest Neighbor Closest	83182	124, 94, 63, 53, 180, 154, 135, 123, 65, 116, 59, 115, 139, 193, 41, 42, 160, 34, 22, 18, 108, 69, 159, 181, 184, 177, 54, 30, 48, 43, 151, 176, 80, 79, 133, 162, 51, 137, 183, 143, 0, 117, 46, 68, 93, 140, 36, 163, 199, 146, 195, 103, 5, 96, 118, 149, 131, 112, 4, 84, 35, 10, 190, 127, 70, 101, 97, 1, 152, 120, 78, 145, 185, 40, 165, 90, 81, 113, 175, 171, 16, 31, 44, 92, 57, 106, 49, 144, 62, 14, 178, 52, 55, 129, 2, 75, 86, 26, 100, 121
	Nearest Neighbor All	78896	118, 51, 176, 137, 183, 89, 23, 186, 143, 117, 93, 140, 0, 80, 151, 162, 133, 63, 79, 94, 124, 53, 97, 26, 100, 152, 1, 2, 120, 44, 25, 78, 16, 171, 175, 113, 56, 31, 145, 179, 92, 129, 57, 185, 106, 52, 55, 178, 49, 102, 14, 62, 9, 148, 144, 40, 119, 81, 196, 165, 90, 101, 86, 75, 180, 154, 135, 70, 123, 112, 4, 84, 127, 59, 65, 149, 131, 116, 43, 42, 181, 160, 54, 30, 177, 10, 190, 184, 34, 193, 159, 22, 146, 18, 108, 41, 139, 46, 68, 115
	Greedy	71488	117, 0, 46, 68, 139, 193, 41, 115, 5, 42, 181, 159, 69, 108, 18,

	Cycle		22, 146, 34, 160, 48, 54, 30, 177, 10, 190, 4, 112, 84, 35, 184, 43, 116, 65, 59, 118, 51, 151, 133, 162, 123, 127, 70, 135, 180, 154, 53, 100, 26, 86, 75, 44, 25, 16, 171, 175, 113, 56, 31, 78, 145, 179, 92, 57, 52, 185, 119, 40, 196, 81, 90, 165, 106, 178, 14, 144, 62, 9, 148, 102, 49, 55, 129, 120, 2, 101, 1, 97, 152, 124, 94, 63, 79, 80, 176, 137, 23, 186, 89, 183, 143, 117
	Greedy Regret Cycle	105852	159, 195, 146, 22, 20, 18, 108, 67, 36, 140, 93, 117, 170, 153, 183, 89, 23, 83, 64, 15, 9, 37, 128, 172, 57, 55, 3, 32, 49, 102, 144, 132, 21, 7, 164, 71, 27, 39, 165, 8, 185, 174, 81, 98, 17, 157, 188, 56, 171, 16, 78, 25, 44, 120, 2, 75, 86, 97, 189, 94, 130, 137, 66, 176, 80, 151, 133, 79, 63, 136, 53, 180, 154, 6, 135, 194, 161, 123, 29, 126, 112, 4, 190, 177, 147, 48, 34, 160, 184, 28, 43, 65, 197, 59, 118, 60, 46, 198, 139, 193
	Weighted Greedy Regret Cycle	71108	117, 0, 46, 68, 139, 193, 41, 115, 5, 42, 181, 159, 69, 108, 18, 22, 146, 34, 160, 48, 54, 177, 10, 190, 4, 112, 84, 184, 43, 116, 65, 59, 118, 51, 151, 133, 162, 123, 127, 70, 135, 154, 180, 53, 121, 100, 26, 86, 75, 44, 25, 16, 171, 175, 113, 56, 31, 78, 145, 179, 196, 81, 90, 40, 165, 185, 106, 178, 138, 14, 144, 62, 9, 148, 102, 49, 52, 55, 92, 57, 129, 82, 120, 2, 101, 1, 97, 152, 124, 94, 63, 79, 80, 176, 137, 23, 186, 89, 183, 143
TSPB	Random	179796	78, 18, 141, 43, 65, 49, 184, 62, 35, 16, 121, 31, 167, 165, 45, 109, 174, 19, 132, 195, 67, 99, 194, 63, 144, 92, 54, 5, 59, 114, 15, 66, 111, 50, 108, 116, 82, 37, 40, 118, 185, 140, 143, 186, 139, 154, 22, 9, 170, 23, 129, 86, 130, 148, 76, 57, 120, 85, 179, 29, 126, 153, 56, 27, 94, 196, 70, 12, 169, 122, 51, 44, 6, 74, 3, 81, 192, 157, 182, 138, 71, 24, 102, 104, 105, 7, 98, 87, 34, 106, 172, 103, 124, 77, 176, 42, 68, 8, 113, 88
	Nearest Neighbor Closest	52319	16, 1, 117, 31, 54, 193, 190, 80, 175, 5, 177, 36, 61, 141, 77, 153, 163, 176, 113, 166, 86, 185, 179, 94, 47, 148, 20, 60, 28, 140, 183, 152, 18, 62, 124, 106, 143, 0, 29, 109, 35, 33, 138, 11, 168, 169, 188, 70, 3, 145, 15, 155, 189, 34, 55, 95, 130, 99, 22, 66, 154, 57, 172, 194, 103, 127, 89, 137, 114, 165, 187, 146, 81, 111, 8, 104, 21, 82, 144, 160, 139, 182, 25, 121, 90, 122, 135, 63, 40, 107, 100, 133, 10, 147, 6, 134, 51, 98, 118, 74
	Nearest Neighbor All	52992	10, 133, 122, 90, 51, 121, 117, 198, 1, 38, 27, 31, 73, 193, 190, 80, 175, 78, 142, 45, 5, 177, 36, 61, 91, 141, 77, 81, 153, 187, 163, 89, 103, 114, 127, 165, 137, 176, 166, 194, 86, 185, 95, 130, 99, 62, 124, 106, 143, 0, 35, 109, 29, 33, 160, 144, 8, 82, 21, 104, 111, 138, 182, 11, 139, 168, 195, 145, 3, 155, 15, 70, 169, 132, 13, 188, 6, 147, 18, 55, 34, 152, 183, 140, 20, 28, 149, 4, 148, 60, 47, 94, 66, 179, 113, 54, 135, 63, 40, 107
	Greedy Cycle	48765	162, 175, 78, 142, 36, 61, 91, 141, 97, 187, 165, 127, 89, 103, 137, 114, 113, 194, 166, 179, 185, 99, 130, 22, 66, 94, 47, 148, 60, 20, 28, 149, 4, 140, 183, 152, 170, 34, 55, 18, 62, 124, 106, 128, 95, 86, 176, 180, 163, 153, 81, 77, 21, 87, 82, 8, 56, 144, 111, 0, 35, 109, 29, 160, 33, 49, 11, 43, 134, 147, 6, 188, 169, 132, 13, 161, 70, 3, 15, 145, 195, 168, 139, 182, 138, 104, 25, 177, 5, 45, 136, 73, 164, 31, 54, 117, 198, 193, 190, 80, 162
	Greedy Regret Cycle	67568	60, 20, 59, 28, 4, 140, 183, 174, 181, 83, 55, 34, 170, 53, 184, 155, 84, 70, 132, 169, 188, 6, 150, 147, 134, 43, 139, 11, 33, 160, 39, 35, 143, 106, 119, 81, 41, 111, 68, 8, 104, 157, 171, 177, 123, 25, 118, 116, 121, 125, 191, 115, 10, 133, 17, 107,

			100, 63, 96, 135, 38, 16, 197, 24, 198, 117, 164, 105, 80, 162, 45, 5, 7, 36, 79, 91, 141, 97, 146, 153, 186, 163, 165, 127, 26, 114, 137, 75, 93, 48, 166, 194, 176, 64, 86, 185, 52, 57, 66, 148
	Weighted Greedy Regret Cycle	47144	95, 130, 99, 22, 179, 185, 86, 166, 194, 113, 176, 26, 103, 114, 137, 127, 89, 163, 187, 153, 81, 77, 141, 91, 61, 36, 175, 78, 142, 45, 5, 177, 21, 82, 111, 8, 104, 138, 182, 139, 168, 195, 145, 15, 3, 70, 13, 132, 169, 188, 6, 147, 115, 10, 133, 122, 63, 135, 38, 1, 117, 193, 31, 54, 131, 90, 51, 121, 118, 74, 134, 11, 33, 160, 29, 0, 109, 35, 143, 106, 124, 128, 62, 18, 55, 34, 170, 152, 4, 149, 28, 20, 60, 94, 66, 47, 148, 199, 183, 140

Table 2. Best solutions and their scores found by each algorithm in both instances.

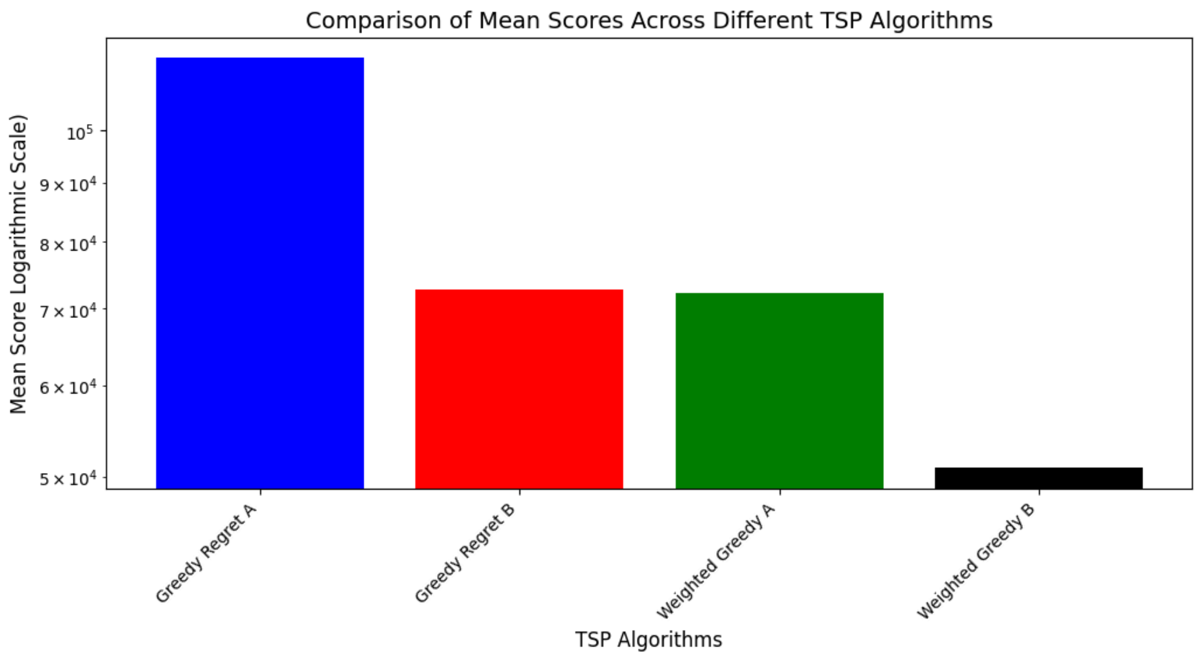

Fig 8. Comparison of different average scores of the algorithms used for instances A and B.

4. Conclusions

The presented experiments revealed the shortcomings of the greedy regret cycle algorithm, as can be seen from the tables comparing the best, average, and worst performances of algorithms, the greedy regret cycle algorithm ends up being outperformed by almost all other methods other than random. This subpar performance might have been caused by the reformulation of the original TSP problem in which we only need to select half of all nodes since the regret algorithm will pick the nodes with the greatest regret which might be good for choosing their position in the cycle relative to the remaining nodes but it might still lead to choosing nodes which in general have too high cost. This might also be the reason for the relatively good performance of the weighted greedy regret cycle algorithm, where both regret and score differences are taken into account. This algorithm generally outperforms all other tested algorithms which might be due to the combination of taking into account the best node placement considering regret and the best nodes to take into the cycle using the score difference calculations. This way, the weighted variant avoids the pitfalls of the greedy regret cycle algorithm by avoiding nodes with too high cost which might have high regret but also increase the score too much. In conclusion, joining regret heuristics with other forms of score calculation can be a good strategy depending on the considered problem and its formulation, as it allows us to consider multiple aspects of the problem.