Assignment 6: Multiple Start Local Search and Iterated Local Search

Mateusz Tabaszewski 151945 Bartłomiej Pukacki 151942

Source code:
https://github.com/MatTheTab/Evolutionary-Computation/blob/main/Assignment%206%20MSLS%20and%20ILS/Assignment_6_MSLS_ILS.ipynb

1. Description of the Problem

Similar to the previous experiments we will be considering the redefined TSP this time in the context of the Multi Start Local Search (MSLS) and Iterated Local Search (ILS) algorithms. The TSP problem's objective is to minimize an objective function defined as an Euclidean distance of a cycle connecting the available nodes plus the weight/cost of each node in the solution. Furthermore, only half of all nodes need to be included in the final cycle. This time the focus will be on improving the scores achieved by the algorithm rather than improving the computation time. Furthermore, we will focus on creating a perturbation mechanism that will allow us to outperform the MSLS algorithm given the same time frame to solve the problem. Below are the visualizations of the TSPA and TSPB problem instances consistent with the previous definitions.

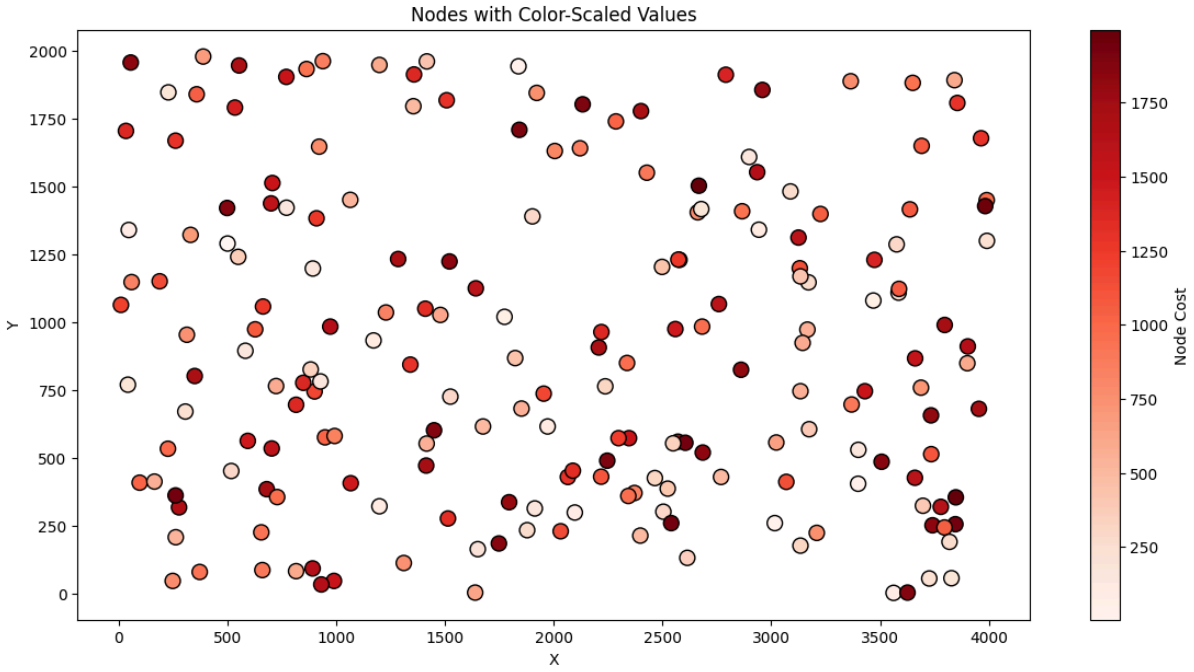

Fig 1. Visualization of the TSPA problem instance, each node's x and y locations on the plot correspond to their given x and y locations and the color intensity signifies the weight/cost of each node. The total length of the cycle and the sum of node weights should be minimized.

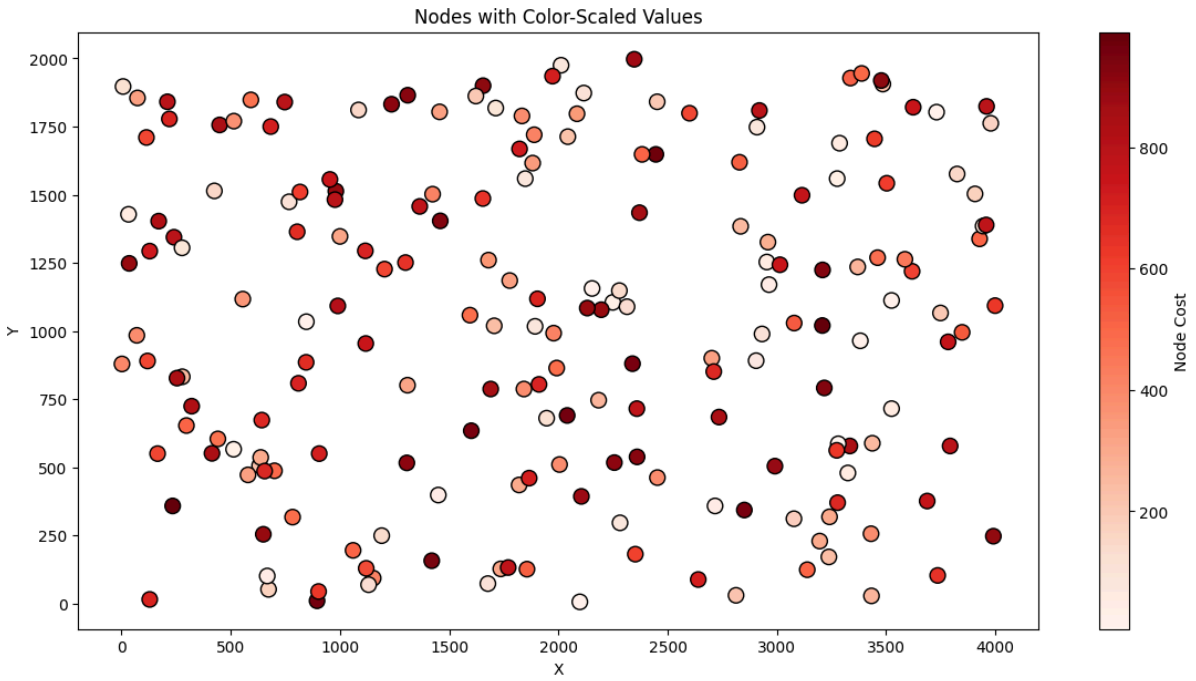

Fig 2. Visualization of the TSPB problem instance, each node's x and y locations on the plot correspond to their given x and y locations and the color intensity signifies the weight/cost of each node. The total length of the cycle and the sum of node weights should be minimized.

2. Algorithms

The implemented algorithms are **Multi Start Local Search (MSLS)** and **Iterated Local Search (ILS)**. Both of these algorithms should introduce performance increase in comparison to the original steepest local search, at the cost of computation time. First, the **MSLS algorithm** is based simply on utilizing the steepest local search algorithm multiple times while starting from different random solutions. In the end, the best solution found from all runs is returned. To implement this algorithm efficiently, we have utilized the previous implementation of the steepest delta search. The random solution is initialized and passed to the steepest delta search 200 times in a single run of MSLS, which is then repeated 20 times for each problem instance: TSPA, TSPB. When it comes to the **ILS algorithm**, the idea is similar, however, this time, the algorithm only initially is given a random solution, which is then perturbed and passed to the steepest delta search. If a better than the previous solution is found then we perturb it and continue, otherwise, the previous solution is used again. This is repeated in such a way that it takes the same amount of time as the 200 MSLS runs for the sake of making a fair comparison. The experiment was also repeated 20 times for each problem instance. It can be noticed that the choice of the perturbation algorithm is crucial to the success of the ILS algorithm. Although we tested multiple different algorithms, and many of them allowed us to outperform MSLS, in the end, we have decided to focus on the following three **perturbation algorithms**: Partial Shuffle, MST Perturbation, and Coordinates Change algorithms.

Partial Shuffle - This perturbation algorithm is centered around choosing a node randomly from the solution and then choosing a node that is num_nodes away from this node, so for example, if we randomly choose the node that is 5th in the solution, and the num_nodes is 10, then the next chosen node will be the 15th one. Then, every node between these two is

shuffled randomly, producing a new solution. Different values of num_nodes were tested but in the end, choosing 10% of all nodes, that is 20, ended up being the best choice and the one visualized in the later sections. This perturbation algorithm should work, since the structure of the solution as a whole does not change a lot but the local change between the selected nodes is big, likely forcing the algorithm to find a new locally-optimal solution.

MST Perturbation - The main idea behind MST Perturbation is centered around choosing two nodes, the starting node and the ending node within the distance of a given number of nodes. The first node is always chosen randomly and the following one is determined by some distance chosen when using the algorithm. Then a Minimum Spanning Tree taking into account weights is constructed between these nodes with the order of the nodes determining the new order of the solution between the starting node and the ending node. This way, the solution between these two points might change significantly enough to cause the algorithm to find a new solution but the perturbation is likely to be more locally optimal than in the case of Partial Shuffle. As such, the algorithm is, in a sense, more exploitative and less explorative than the other perturbations.

Coordinates Change - This algorithm works by redefining the distances between nodes and then determining the new optimum using the steepest delta search before turning the weights back to their original values. In other words, we can change values in some range according to a uniform distribution. Let's assume the distance between node 1 and node 2 is 1000, and that we can change the distance by at most 500, let's also assume that we randomly shifted the distance by 200, so the new distance is now 1200 between node 1 and node 2. Then, we will repeat the process for every position in our distance matrix (but in such a way that the distances are still symmetric) and force the algorithm to recalculate the local optimum. Then we turn the distance matrix back to its original values before resuming the algorithm. This way, we will find a local optimum for a similar, albeit slightly different problem, allowing for the escape from the previous local optimum. Our experiments have shown that the best upper bound for changing the distance is 1000 after testing for values of (200, 400, 1000, 1400).

```
FUNCTION MSLS(distance_matrix, weights, num_iterations):
    INPUT:
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node
        num_iterations - number of runs of the algorithm

    best_solution ← None
    FOR iteration IN num_iterations:
       start_solution, start_score ← FIND RANDOM solution
       end_solution, end_score ← steepest_algorithm(start_solution, start_score,
distance_matrix, weights)
       if end_score IS better best_score:
           best_score ← end_score
           best_solution ← end_solution
    RETURN best_solution, best_score
```

```
FUNCTION ILS(distance_matrix, weights, allowed_time, perturbation_function):
    INPUT:
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node
        allowed_time- maximum allowed time
        perturbation_function - chosen perturbation function

    best_solution ← None
    current_solution, current_score ← FIND RANDOM solution
    While the total running time < allowed_time:
       current_solution, current_score ← perturbation_function(distance_matrix,
weights, current_solution, current_score)
       current_solution, current_score ← steepest_algorithm(current_solution,
current_score, distance_matrix, weights)
       IF current_score IS better than best_score:
            best_score ← current_score
            best_solution ← current_solution
        ELSE:
            current_score ← best_score
            current_solution ← best_solution
    RETURN best_solution, best_score
```

Perturbation Functions:

```
FUNCTION partial_shuffle_perturbation_20(distance_matrix, weights, solution,
score, num_nodes):

    INPUT:
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node
        solution- current solution
        score- current score
        num_nodes - number of nodes to change, by default 20

    start_node ← SELECT random node FROM solution
    end_node ← start_node + num_nodes
    solution ← SHUFFLE solution FROM start_node TO end_node
    score ← UPDATE score
    return solution, score
```

```
FUNCTION mst_perturbation(distance_matrix, weights, solution, score, num_nodes =
20):

    INPUT:
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node
        solution- current solution
        score- current score
        num_nodes - number of nodes to change, by default 20

    start_node ← SELECT random node FROM solution
    end_node ← start_node + num_nodes
    MST_solution ← CONSTRUCT MST ON solution FROM start_node TO end_node
    solution ← ORDER OF nodes ADDED TO MST_solution
    score ← UPDATE score
    return solution, score

FUNCTION change_coordinates_perturbation_medium(distance_matrix, weights,
solution, score, max_perturbation):

    INPUT:
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node
        solution- current solution
        score- current score
        max_perturbation- max change allowed in the distance matrix, by default
                                                                          1000
     update_matrix ← RANDOM values BETWEEN 0 AND max_perturbation IN SHAPE OF
                                                                 distance_matrix
     update_matrix ← update_matrix + update_matrix.T (it is now symmetric)
     new_distance_matrix ← distance_matrix + update_matrix
     solution ← steepest_algorithm(solution, score, new_distance_matrix , weights)
      score ← UPDATE score
      return solution, score
```

3. Experiments

To quantify the performance of the tested algorithms results for 20 runs on each problem instance were presented in the below table:

Method	TSPA av (min - max)	TSPB av (min - max)	Avr. Num Iterations (TSPA/TSPB)
MSLS	**71340 (70919-71756)**	**45952 (45365-46428)**	**200/200**
ILS - Partial Shuffle	**69545 (69141 - 70200)**	**43952 (43448 - 44659)**	**1548/1495**
ILS - MST Perturbation	**70133 (69246 - 71271)**	**44360 (43658 - 45019)**	**1582/1525**
ILS - Coordinate Change	**69882 (69460 - 70419)**	**44201 (43574 - 44853)**	**315/345**
Steepest Delta Search	73910 (71118-78710)	48574 (46300-51342)	-

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

The **best scores achieved** are visualized below.

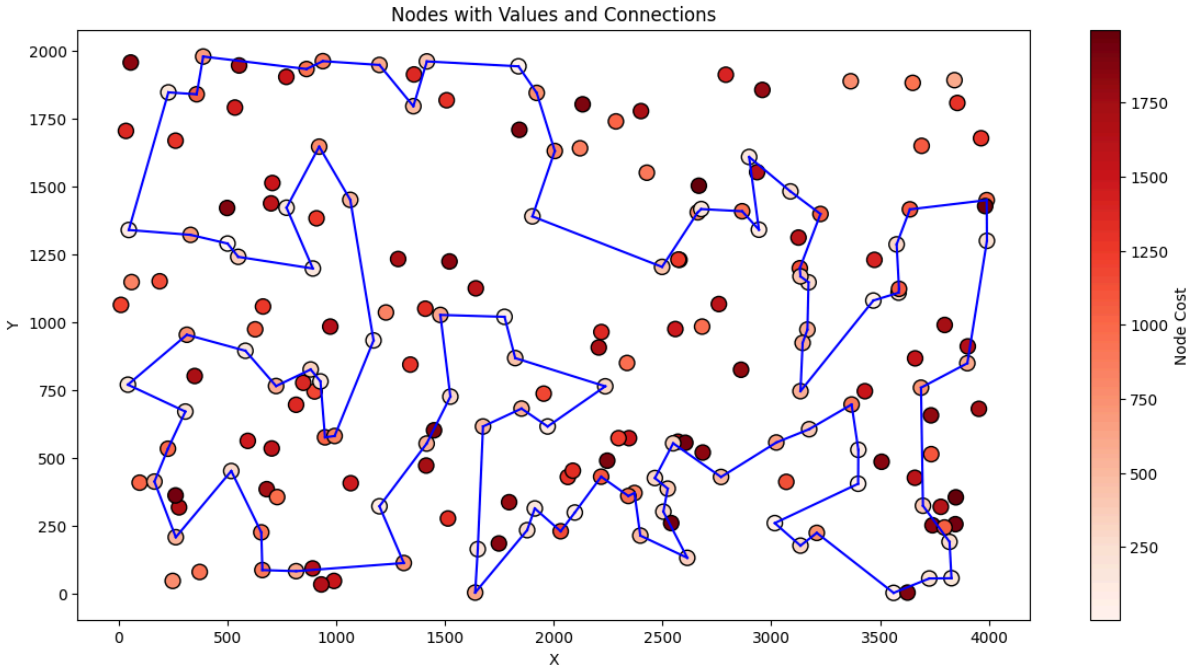

Fig 3. Visualization of the best solution found by the **MSLS** on the TSPA problem instance starting from a random solution.

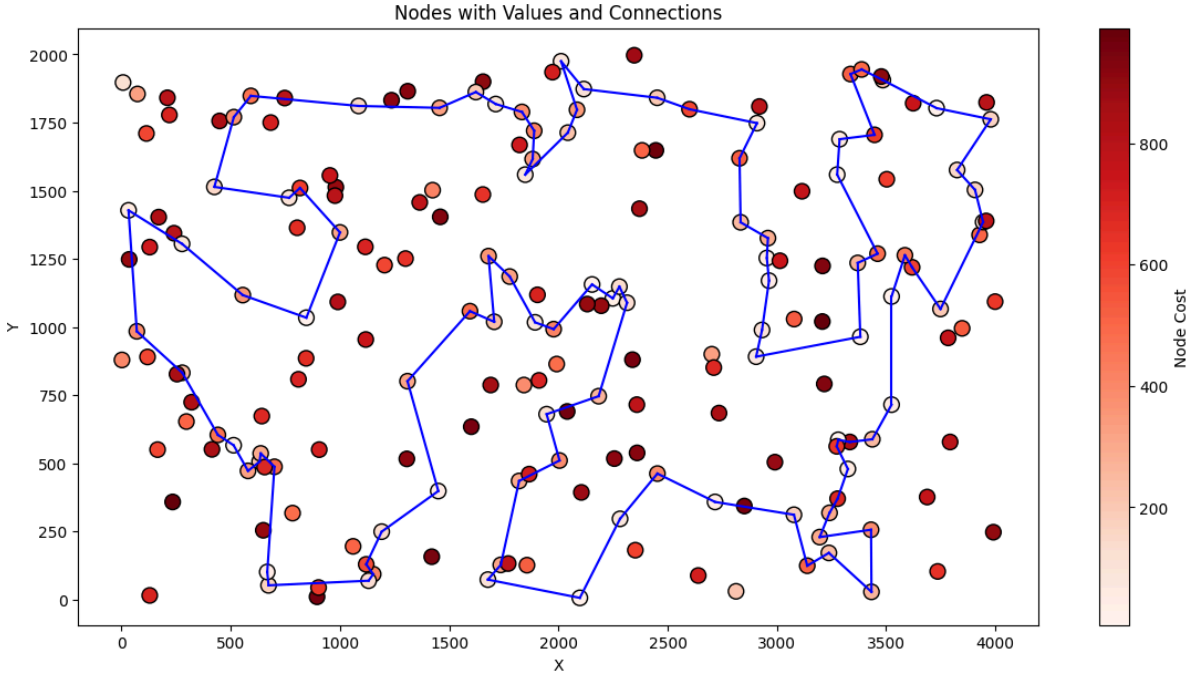

Fig 4. Visualization of the best solution found by the **MSLS** on the TSPB problem instance starting from a random solution.

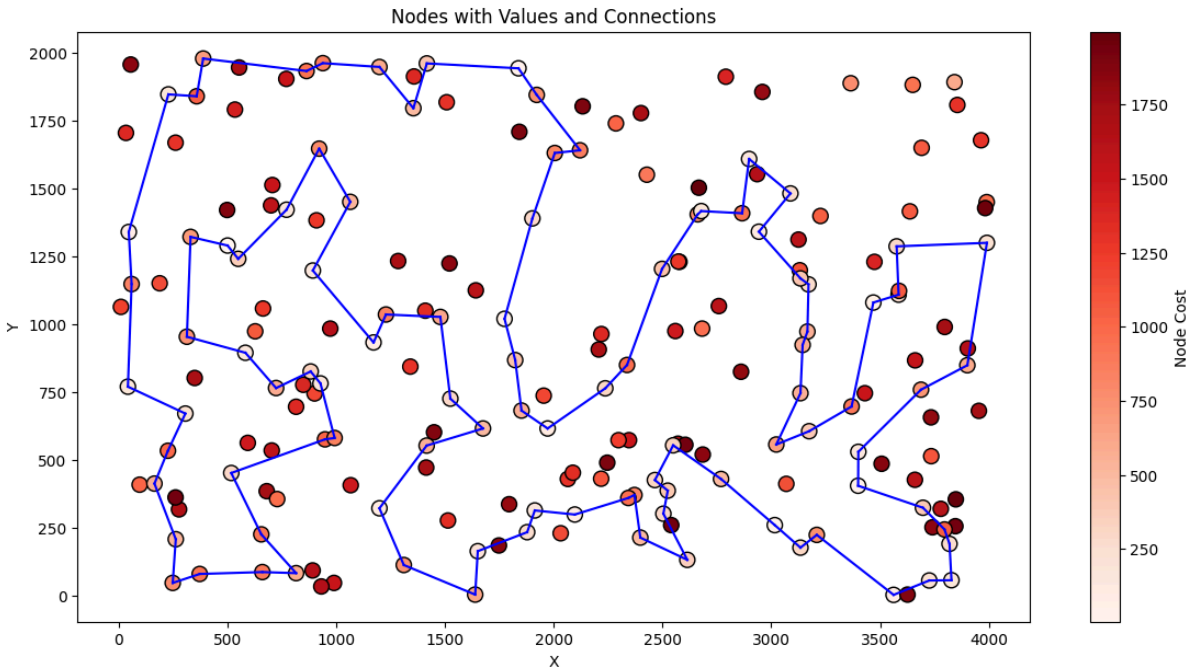

Fig 5. Visualization of the best solution found by the **ILS - Partial Shuffle** on the TSPA problem instance starting from a random solution.

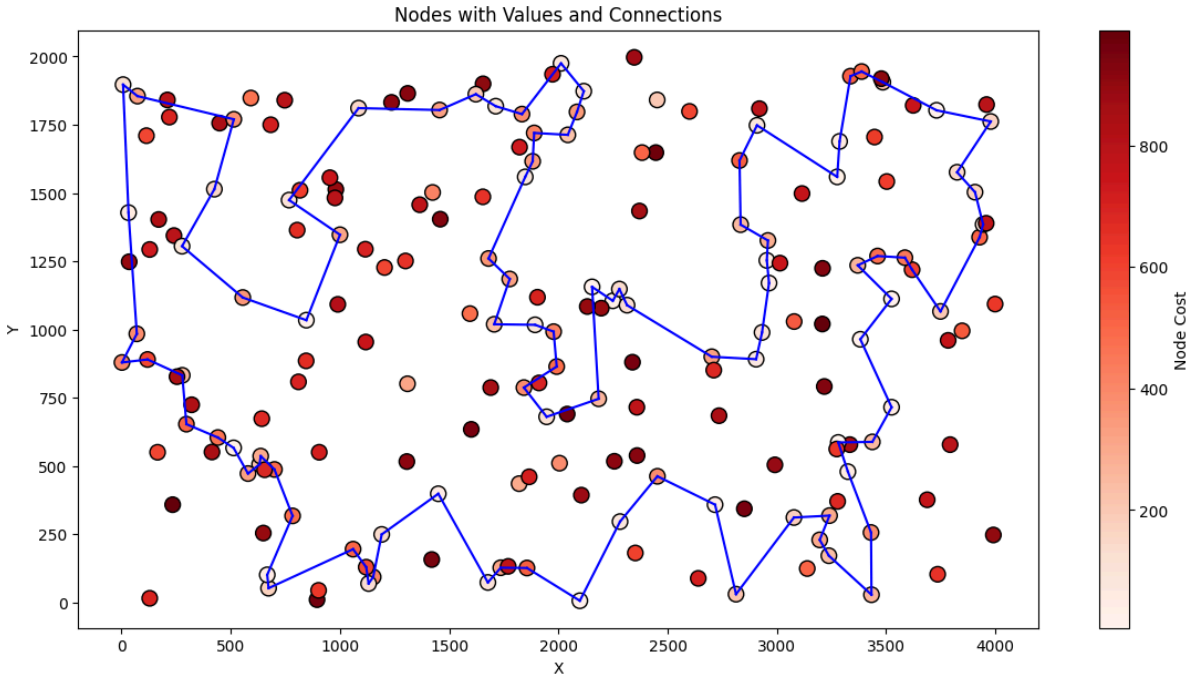

Fig 6. Visualization of the best solution found by the **ILS - Partial Shuffle** on the TSPB problem instance starting from a random solution.

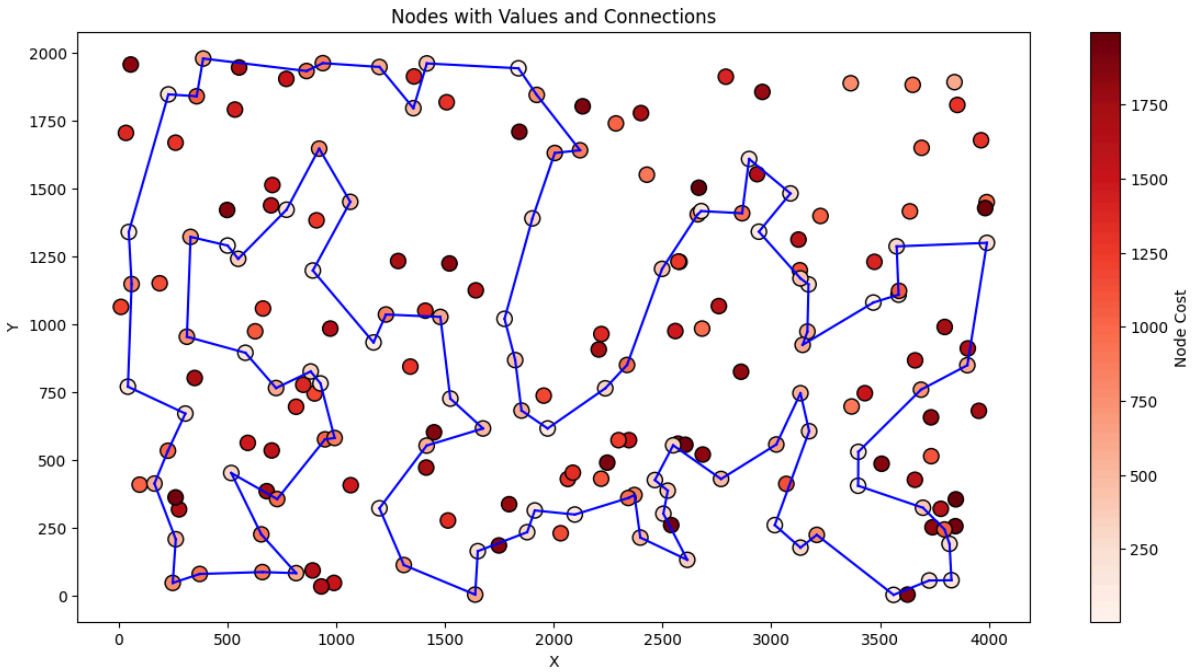

Fig 7. Visualization of the best solution found by the **ILS - MST Perturbation** on the TSPA problem instance starting from a random solution.

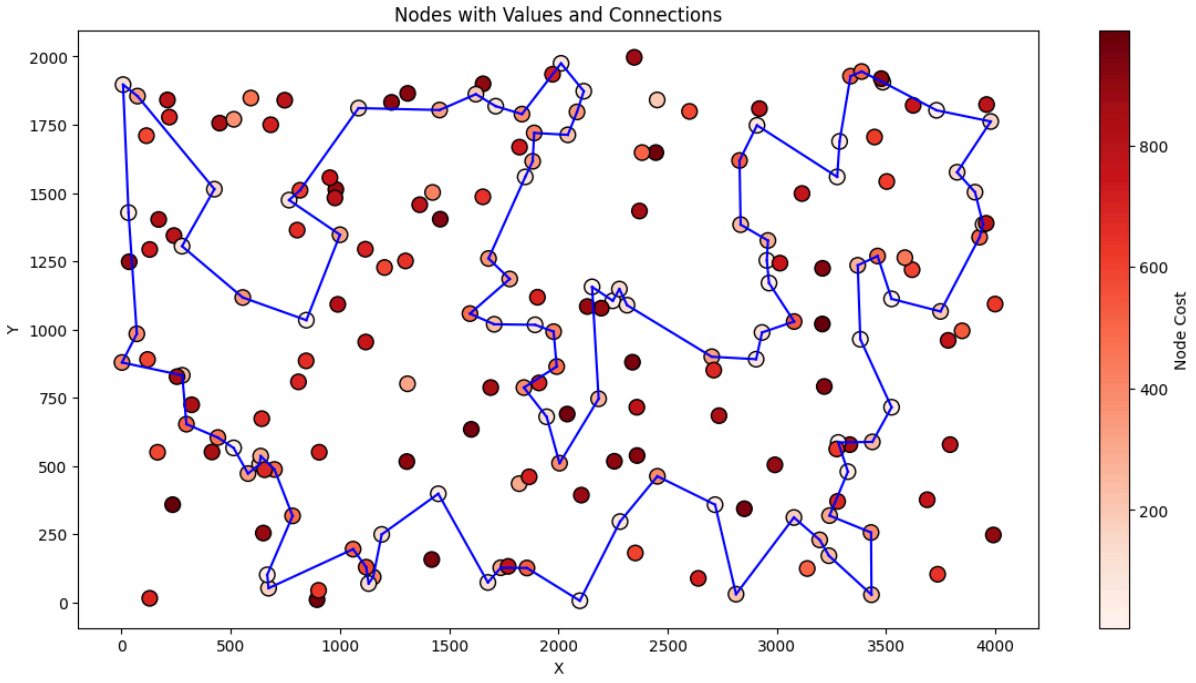

Fig 8. Visualization of the best solution found by the **ILS - MST Perturbation** on the TSPB problem instance starting from a random solution.

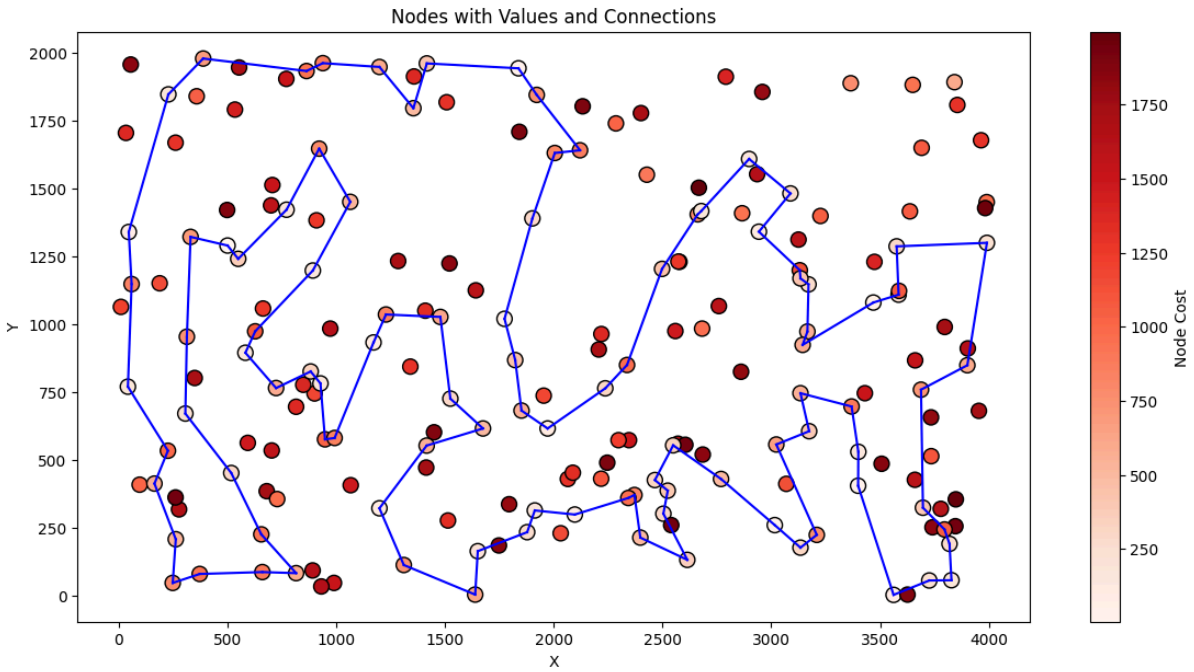

Fig 9. Visualization of the best solution found by the **ILS - Coordinate Change** on the TSPA problem instance starting from a random solution.

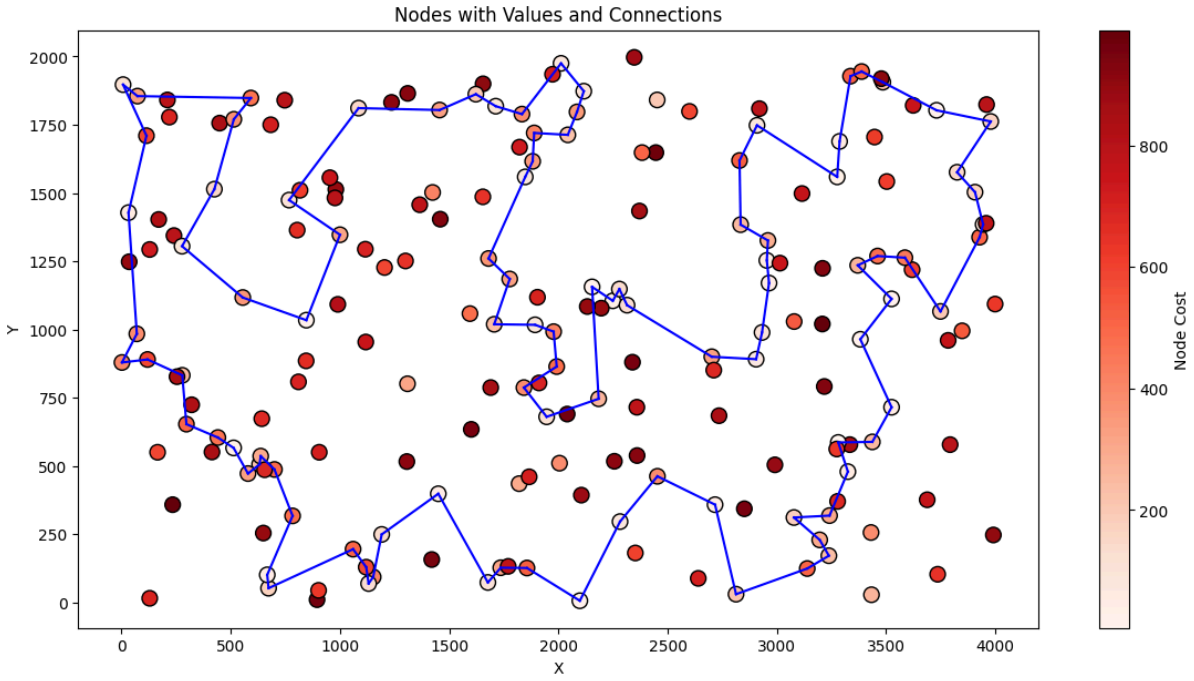

Fig 10. Visualization of the best solution found by the **ILS - Coordinate Change** on the TSPB problem instance starting from a random solution.

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	MSLS	70919	151, 162, 123, 127, 112, 4, 84, 184, 177, 54, 48, 160, 34, 181, 42, 43, 116, 65, 131, 149, 59, 46, 68, 139, 115, 41, 193, 159, 22, 18, 69, 108, 140, 93, 117, 0, 143, 183, 89, 23, 137, 148, 9, 62, 102, 49, 144, 14, 138, 3, 178, 106, 52, 55, 57, 185, 40, 119, 165, 39, 27, 90, 81, 196, 31, 113, 175, 171, 16, 25, 44, 120, 78, 145, 179, 92, 129, 2, 152, 97, 1, 101, 75, 86, 26, 100, 121, 53, 158, 180, 154, 70, 135, 133, 79, 63, 94, 80, 176, 51
	ILS - Partial Shuffle	69141	184, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 146, 22, 18, 69, 108, 140, 93, 117, 0, 143, 183, 89, 186, 23, 137, 176, 80, 79, 63, 94, 124, 148, 9, 62, 102, 144, 14, 49, 178, 106, 52, 55, 57, 129, 92, 179, 185, 40, 165, 90, 81, 196, 145, 78, 31, 56, 113, 175, 171, 16, 25, 44, 120, 2, 152, 97, 1, 101, 75, 86, 26, 100, 53, 180, 154, 135, 70, 127, 123, 162, 133, 151, 51, 118, 59, 115, 46, 68, 139, 41, 193, 159, 181, 42, 43, 116, 65, 149, 131
	ILS - MST Perturbation	69246	9, 62, 102, 144, 14, 49, 178, 106, 52, 55, 185, 40, 165, 90, 81, 196, 145, 78, 31, 56, 113, 175, 171, 16, 25, 44, 120, 92, 57, 129, 2, 152, 97, 1, 101, 75, 86, 26, 100, 53, 180, 154, 135, 70, 127, 123, 162, 133, 151, 51, 118, 59, 115, 46, 68, 139, 41, 193, 159, 181, 42, 43, 116, 65, 149, 131, 35, 184, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 146, 22, 18, 69, 108, 140, 93, 117, 0, 143, 183, 89, 186, 23, 137, 176, 80, 79, 63, 94, 124, 148
	ILS - Coordinate Change	69460	100, 53, 180, 154, 135, 70, 127, 123, 162, 133, 151, 51, 118, 59, 149, 131, 65, 116, 43, 42, 5, 115, 46, 68, 139, 41, 193, 159, 181, 160, 184, 84, 112, 4, 190, 10, 177, 54, 48, 34, 146, 22, 18, 108, 140, 93, 117, 0, 143, 183, 89, 186, 23, 137, 176, 80, 79, 63, 94, 124, 148, 9, 62, 144, 14, 49, 3, 178, 106, 52, 55, 185, 40, 165, 90, 81, 196, 31, 56, 113, 175, 171, 16, 78, 145, 179, 57, 92, 129, 25, 44, 120, 2, 152, 97, 1, 101, 75, 86, 26
TSPB	MSLS	45365	10, 147, 6, 188, 169, 132, 13, 195, 168, 145, 15, 70, 3, 155, 184, 152, 170, 34, 55, 18, 62, 124, 106, 86, 95, 130, 183, 140, 199, 4, 149, 28, 20, 60, 148, 47, 94, 179, 22, 99, 185, 166, 194, 88, 176, 180, 113, 26, 103, 89, 114, 137, 127, 165, 163, 153, 81, 77, 141, 36, 61, 21, 82, 8, 111, 35, 109, 0, 29, 160, 33, 11, 139, 138, 182, 25, 177, 5, 142, 78, 175, 80, 190, 73, 54, 31, 193, 117, 198, 1, 38, 63, 135, 131, 121, 51, 191, 90, 122, 133
	ILS - Partial Shuffle	43448	81, 153, 187, 163, 103, 89, 127, 137, 114, 113, 176, 194, 166, 86, 185, 95, 130, 99, 22, 179, 66, 94, 47, 148, 60, 20, 28, 149, 4, 140, 183, 152, 170, 34, 55, 18, 62, 124, 106, 143, 35, 109, 0, 29, 111, 8, 104, 144, 160, 33, 138, 11, 139, 168, 195, 13, 145, 15, 3, 70, 132, 169, 188, 6, 147, 90, 51, 121, 131, 135, 122, 133, 107, 40, 63, 38, 27, 16, 1, 156, 198, 117, 193, 31, 54, 73, 136, 190, 80, 45, 142, 175, 78, 5, 177, 36, 61, 91, 141, 77
	ILS - MST Perturbation	43658	89, 127, 137, 114, 103, 113, 176, 194, 166, 86, 95, 130, 185, 179, 66, 94, 47, 148, 60, 20, 28, 149, 4, 140, 183, 152, 170, 34, 55, 18, 62, 128, 124, 106, 143, 35, 109, 0, 29, 111, 82, 8, 104, 144, 160, 33, 138, 182, 11, 139, 168, 195, 13, 145, 15, 3, 70, 132, 169, 188, 6, 147, 191, 90, 51, 121, 131, 135, 122, 107, 40,

			63, 38, 27, 1, 156, 198, 117, 193, 31, 54, 73, 136, 190, 80, 45, 142, 175, 78, 5, 177, 36, 61, 91, 141, 77, 81, 153, 187, 163
	ILS - Coordinate Change	43574	81, 153, 187, 163, 103, 89, 127, 137, 114, 113, 176, 194, 166, 86, 185, 95, 130, 99, 22, 179, 66, 94, 47, 148, 60, 20, 28, 149, 4, 140, 183, 152, 170, 34, 55, 18, 62, 124, 106, 143, 35, 109, 0, 29, 111, 8, 104, 144, 160, 33, 138, 11, 139, 168, 195, 13, 145, 15, 3, 70, 132, 169, 188, 6, 147, 90, 51, 121, 131, 135, 122, 133, 107, 40, 63, 38, 27, 16, 1, 156, 198, 117, 193, 31, 54, 73, 136, 190, 80, 45, 142, 175, 78, 5, 177, 36, 61, 91, 141, 77

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

Method	TSPA av (min - max) [s]	TSPB av (min - max) [s]
Greedy LS Rand	1.273 (1.047 - 1.975)	1.258 (0.991 - 1.646)
Steepest LS Rand	4.283 (3.261 - 6.218)	4.501 (3.292 - 5.609)
Greedy LS Edges Rand	1.171 (0.981 - 1.34)	1.113 (0.945 - 1.446)
Greedy LS Best	0.067 (0.025 - 0.145)	0.077 (0.033 - 0.187)
Steepest LS Best	0.170 (0.055 - 0.529)	0.196 (0.09 - 0.746)
Greedy LS Edges Best	0.062 (0.025 - 0.115)	0.078 (0.036 - 0.212)
Steepest LS Edges Best	0.194 (0.078 - 0.379)	0.229 (0.114 - 0.836)
Steepest LS Edges Rand	3.571 (2.978 - 4.168)	3.654 (2.976 - 4.364)
Steepest Candidate Search	0.584 (0.479 - 0.705)	0.562 (0.481 - 0.693)
Steepest Delta Search	0.486 (0.400 - 0.625)	0.483 (0.383 - 0.610)
MSLS	**97.988 (92.859 - 104.159)**	**90.370 (83.360 - 98.981)**

Table 3. Minimum, average, and maximum run time achieved by local search methods on both problem instances.

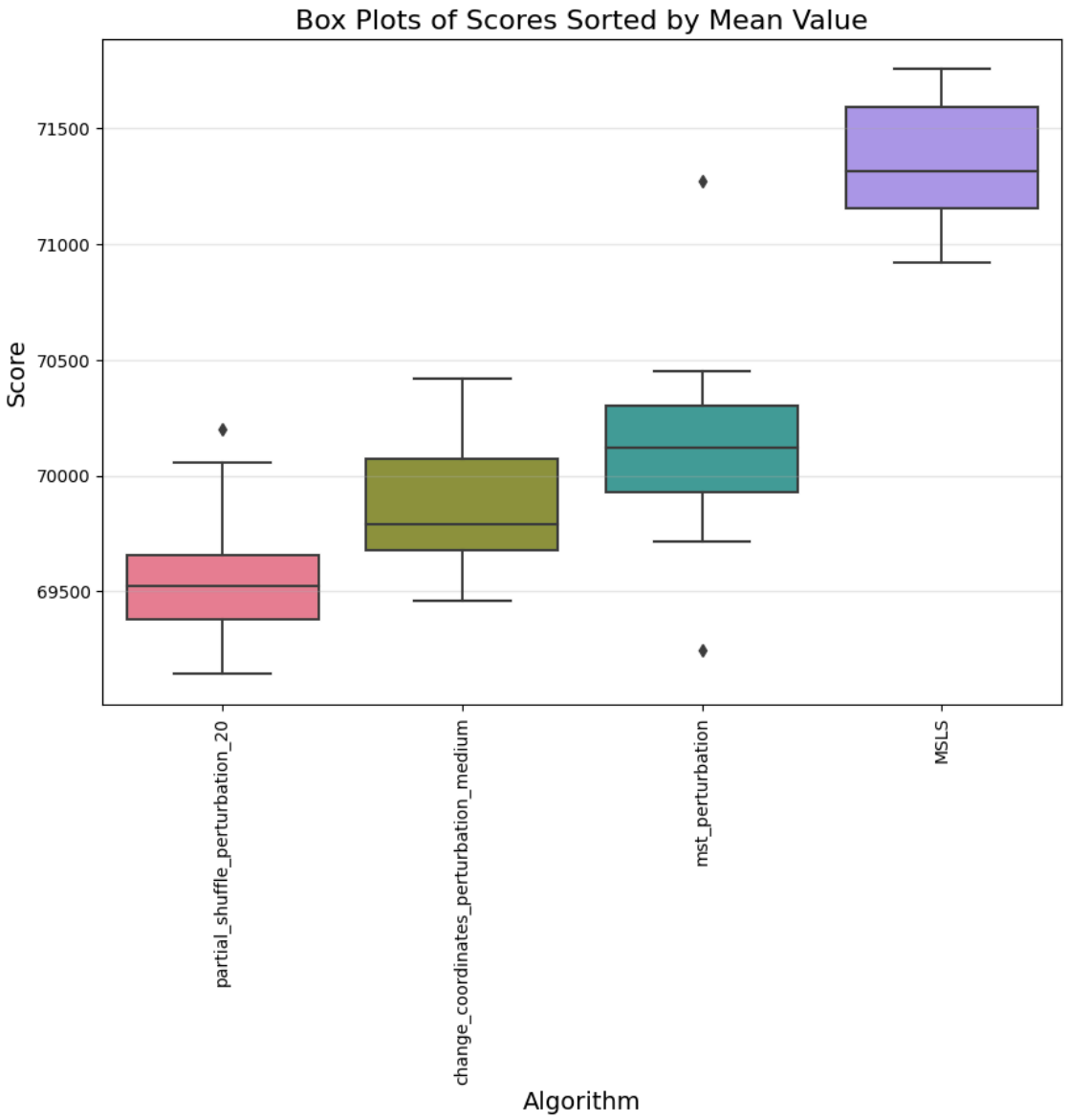

Fig 7. Visualization of score distributions for the different **MSLS** and **ILS** algorithms on the TSPA instance.

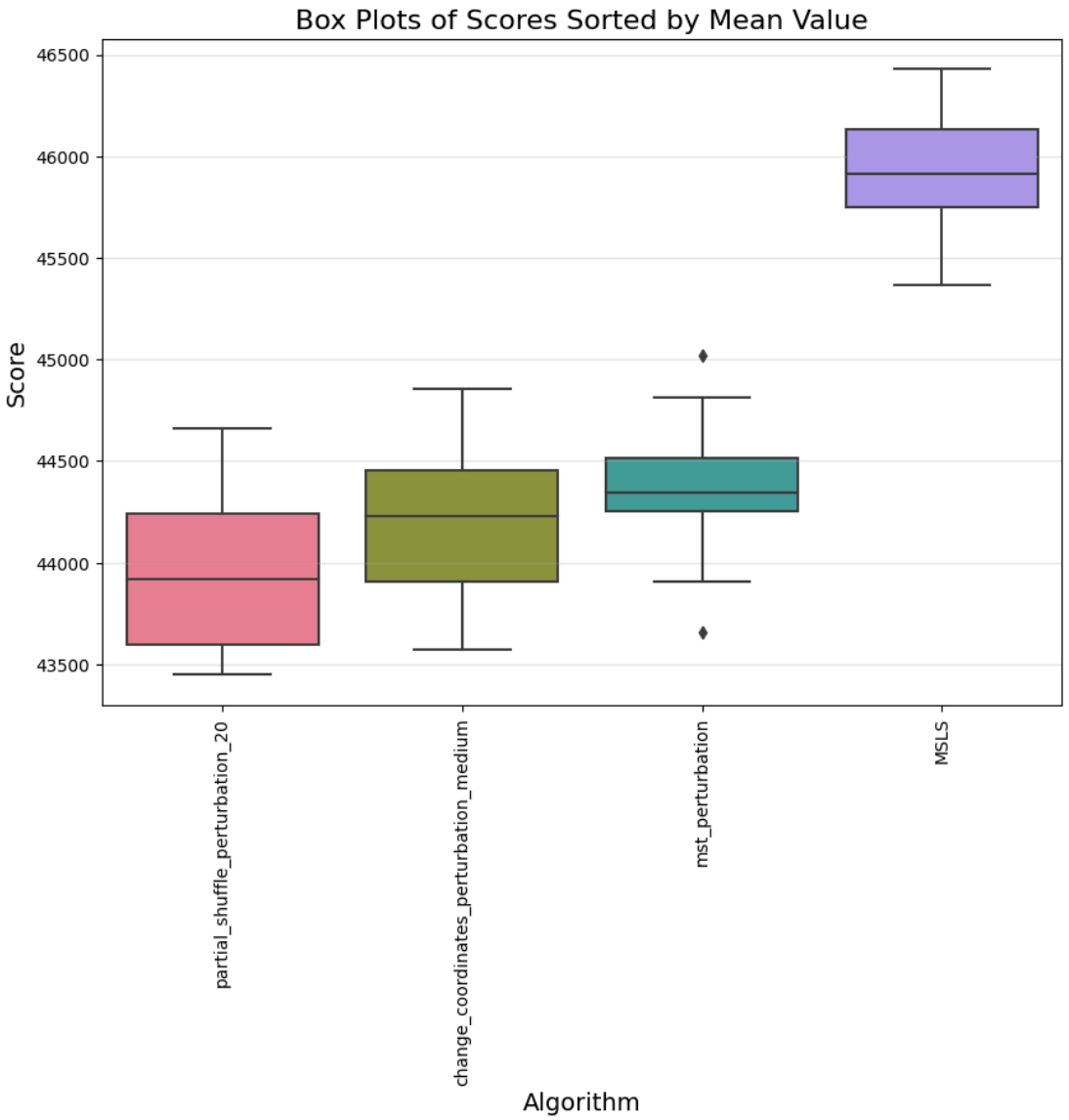

Fig 8. Visualization of score distributions for the different **MSLS** and **ILS** algorithms on the TSPB instance.

4. Conclusions

The conducted experiments showcase that it is possible to create a perturbation algorithm that, when used in ILS, achieves better performance than MSLS. We proposed three algorithms working in different ways, and each of them has managed to surpass the score of the original MSLS. The algorithms also had a higher number of iterations than the original MSLS in the same timeframe. Although this might seem weird at first, in reality, it is very likely that the most time-consuming part of these algorithms is the steepest local search. However, in the case of ILS, we are not always starting from a random solution, but a perturbation of an already good solution, meaning that the next local optimum is likely to be close and as such, can be achieved with fewer iterations of the steepest local search, leading to faster iterations. Furthermore, it has been deduced that the perturbation algorithm should introduce substantial enough change to force the steepest local search method to find a new local optimum but not too different so that the results are similar to the random solution as an input. As such, seemingly simplistic algorithms like shuffling a substring of nodes become surprisingly effective in this scenario, however solutions like changing the coordinates or generating an MST should also not be ignored. Lastly, the exact parameters of the algorithm will depend on the problem instance since the MST perturbation and partial shuffle both depend on the number of nodes in the solution while the change of coordinate values upper bound must be determined by the distances and weights of nodes. As such, from our limited experiments, we propose to use about 20% of nodes in the solution as the input parameter for the partial shuffle perturbation algorithm and the MST perturbation algorithm. We would also propose that slightly less than half of the maximum distance between nodes would be a good parameter for the upper bound in the coordinate perturbation algorithm.