POZNAN UNIVERSITY OF TECHNOLOGY

Assignment 3: Local Search

Mateusz Tabaszewski 151945

Bartłomiej Pukacki 151942

1. Description of the Problem..... 3
2. Algorithms..... 5
3. Experiments.....10
4. Conclusions.....24

Source code:
https://github.com/MatTheTab/Evolutionary-Computation/blob/main/Assignment%203%20Local%20Search/Assignment_3_Local_Search.ipynb

1. Description of the Problem

This report describes the solution to the modified TSP problem using the Local Search algorithms to refine the results. For each instance, the random and the best performing method is used as the base solution. In the TSPA instance the weighted greedy regret cycle and in the TSPB instance, the nearest neighbor algorithm with all node insertion positions considered.

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*

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

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

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

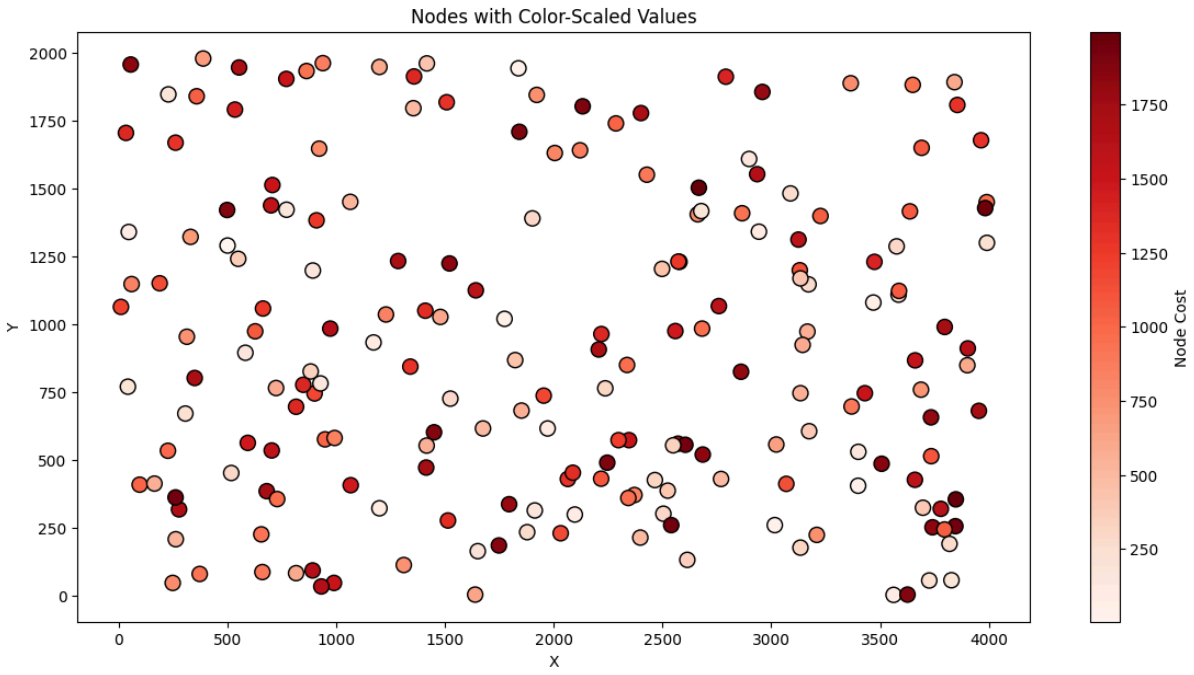

Fig 1. Visualization of the TSPA problem instance, each node's x and y locations on the plot correspond to their given x and y locations and the color intensity signifies the weight/cost of each node. The total length of the cycle and the sum of node weights should be minimized.

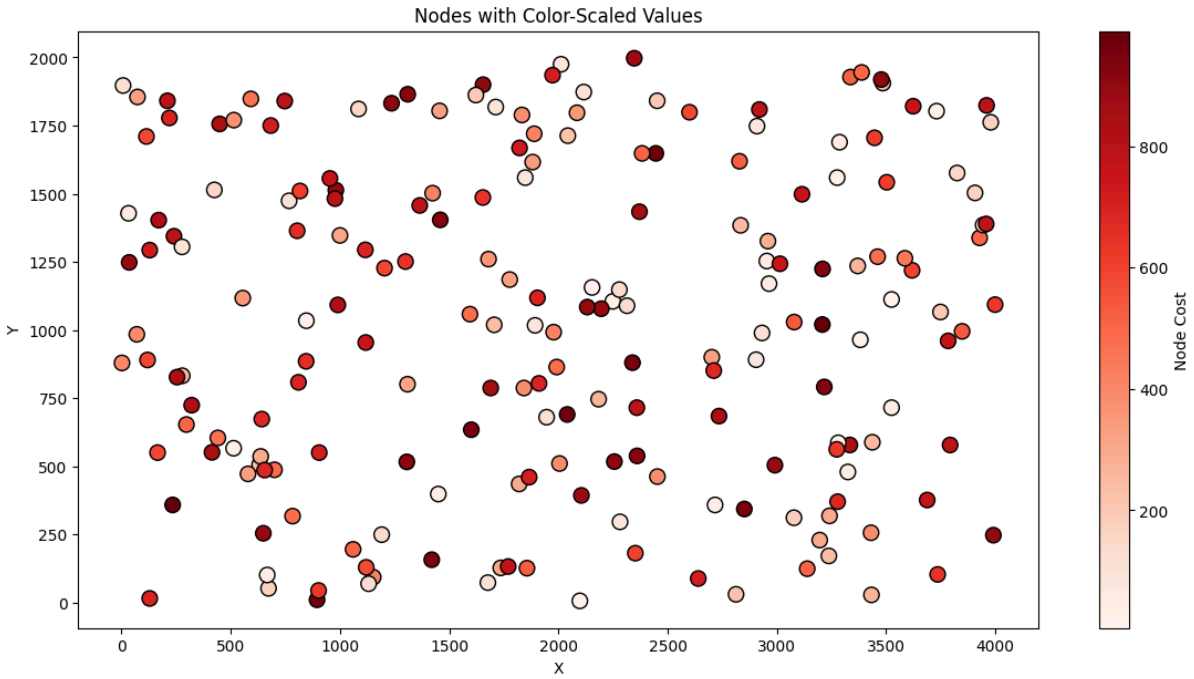

Fig 2. Visualization of the TSPB problem instance, each node's x and y locations on the plot correspond to their given x and y locations and the color intensity signifies the weight/cost of each node. The total length of the cycle and the sum of node weights should be minimized.

2. Algorithms

Four versions of the local search algorithm were used in the assignment depending on the type of local search (greedy or steepest) and the intra-route move selection (two-node exchange or two-edge exchange). As the inter-route moves, the two-node exchange schema is used in each version. The starting solution of each algorithm was a random solution and the best solution found for the instance in the previous assignments.

The **greedy local search** algorithm utilizing two-node exchange moves as intra- and inter-route moves creates a list of all possible positions of pairs of nodes to be exchanged and while an improvement to the solution is found, shuffles the list and selects the first move improving the score encountered to be applied to the solution. The complexity of every local search algorithm is $O(n^2)$.

```
FUNCTION greedy_local_search(solution, score, distance_matrix, weights):
    INPUT:
        solution - the starting point solution
        score - the initial score of the solution
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node

    moves ← []
    FOR cycle_position in [0, 1, …, length of solution]:
        FOR any_node_position in [0, 1, …, number of all nodes]:
            IF cycle_position IS SMALLER THAN any_node_position:
                APPEND (cycle_position, any_node) to moves

    WHILE improvement is found:
        SHUFFLE moves
        FOR cycle_position, any_node_position in moves:
            IF any_node_position IS BIGGER THAN length of solution
                tested_score, tested_solution ← node_inter_change(...)
            ELSE
                tested_score, tested_solution ← node_intra_change(...)

            IF tested_score IS SMALLER THAN score:
                solution ← tested_solution
                score ← tested_score
                break loop

    RETURN solution, score
```

```
FUNCTION node_inter_change(solution, score, distance_matrix, weights,
                                    cycle_position, any_node_position):
    INPUT:
        solution - the previous point solution
        score - the previous score of the solution
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node
        cycle_position - position of a node in the solution
        any_node_position - position of a node outside the solution

    remaining_nodes ← nodes NOT IN solution
    relative_position ← any_node_position - length of solution

    node1 ← solution[cycle_position]
    node2 ← remaining_nodes[relative_position]
    node_before ← solution[cycle_position-1]
    node_after ← solution[cycle_position+1] (or solution[0])

    temp_score ← score
    temp_score -= distance_matrix[node_1][node_before]
    temp_score -= distance_matrix[node_1][node_after]
    temp_score -= weights[node_1]
    temp_score += distance_matrix[node_2][node_before]
    temp_score += distance_matrix[node_2][node_after]
    temp_score += weights[node_2]

    temp_solution ← solution
    SWITCH temp_solution[cycle_position] and remaining_nodes[relative_position]

    RETURN temp_score, temp_solution


FUNCTION node_intra_change(solution, score, distance_matrix, cycle_position,
                                                        any_node_position):
    INPUT:
        solution - the previous point solution
        score - the previous score of the solution
        distance_matrix - matrix of distances between nodes
        cycle_position - position of a node in the solution
        any_node_position - position of a node outside the solution

    temp_score ← score
    IF nodes are next to each other:
        node_1 ← node at earlier position in cycle
        node_2 ← node at later position in cycle
        node_before ← node at position before node_1
        node_after ← node at position after node_2

        temp_score -= distance_matrix[node_2][node_after]
        temp_score -= distance_matrix[node_1][node_before]
        temp_score += distance_matrix[node_1][node_after]
```

```
        temp_score += distance_matrix[node_2][node_before]
    ELSE:
      node_1 ← solution[cycle_position]
      node_2 ← solution[any_node_position]
      node_before_1 ← solution[cycle_position-1]
      node_after_1 ← solution[cycle_position+1] (or solution[0])
      node_before_2 ← solution[any_node_position-1]
      node_after_2 ← solution[any_node_position+1] (or solution[0])

      temp_score -= distance_matrix[node_1][node_before_1]
      temp_score -= distance_matrix[node_1][node_after_1]
      temp_score -= distance_matrix[node_2][node_before_2]
      temp_score -= distance_matrix[node_2][node_after_2]
      temp_score += distance_matrix[node_1][node_before_2]
      temp_score += distance_matrix[node_1][node_after_2]
      temp_score += distance_matrix[node_2][node_before_1]
      temp_score += distance_matrix[node_2][node_after_1]

    temp_solution ← solution
    SWITCH temp_solution[cycle_position] and temp_solution[any_node_position]

    RETURN temp_score, temp_solution
```

Another implemented algorithm was the **steepest local search** utilizing two-nodes exchange as inter- and intra-route moves. This method is similar to the previous one with the exception that all moves in the neighborhood are evaluated and the best one is selected in each iteration. The complexity of this algorithm is $O(n^2)$.

```
FUNCTION steepest_local_search(solution, score, distance_matrix, weights):
    INPUT:
        solution - the starting point solution
        score - the initial score of the solution
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node

    moves ← []
    FOR cycle_position in [0, 1, …, length of solution]:
        FOR any_node_position in [0, 1, …, number of all nodes]:
            IF cycle_position IS SMALLER THAN any_node_position:
                APPEND (cycle_position, any_node) to moves

    WHILE improvement is found:
        FOR cycle_position, any_node_position in moves:
            IF any_node_position IS BIGGER THAN length of solution
                tested_score, tested_solution ← node_inter_change(...)
            ELSE
                tested_score, tested_solution ← node_intra_change(...)

            IF tested_score IS SMALLER THAN neighborhood_best_score:
```

```
                neighborhood_best_solution ← tested_solution
                neighborhood_best_score ← tested_score

        IF neighborhood_best_score IS SMALLER THAN score:
            solution ← neighborhood_best_solution
            score ← neighborhood_best_score

    RETURN solution, score
```

The following two algorithms are the implementations of a greedy and steepest local search using two-edge exchange as the intra-route move creation logic. The complexity of both **greedy_local_search_edges** and **steepest_local_search_edges** is $O(n^2)$.

```
FUNCTION greedy_local_search_edges(solution, score, distance_matrix, weights):
    INPUT:
        solution - the starting point solution
        score - the initial score of the solution
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node

    remaining_positions ← list of node positions outside solution
    moves ← []
    FOR cycle_position in [0, 1, ..., length of solution]:
        FOR any_node_position in remaining_positions:
            APPEND (cycle_position, any_node) to moves

        FOR any_node_position in [cycle_position + 2, cycle_position + 3, ...,
number of nodes in solution]:
            IF cycle_position == 0 and any_node_position == number of nodes - 1:
               SKIP iteration

            APPEND (cycle_position, any_node_position) to moves

    WHILE improvement is found:
        SHUFFLE moves
        FOR cycle_position, any_node_position in moves:
            IF any_node_position IS BIGGER THAN length of solution
                tested_score, tested_solution ← node_inter_change(...)
            ELSE
                tested_score, tested_solution ← edge_intra_change(...)

            IF tested_score IS SMALLER THAN score:
                solution ← tested_solution
                score ← tested_score
                break loop

    RETURN solution, score
```

```
FUNCTION edge_intra_change(solution, score, distance_matrix, cycle_position,
                                                        any_node_position):
    INPUT:
        solution - the previous point solution
        score - the previous score of the solution
        distance_matrix - matrix of distances between nodes
        cycle_position - position of a node in the solution
        any_node_position - position of a node outside the solution

    edge_1_start ← solution[cycle_position]
    edge_1_end ← solution[cycle_position+1]
    edge_2_start ← solution[any_node_position]
    edge_2_end ← solution[(any_node_position + 1) % number of nodes in solution]

    temp_score ← score
    temp_score -= distance_matrix[edge_1_start][edge_1_end]
    temp_score -= distance_matrix[edge_2_start][edge_2_end]

    temp_score += distance_matrix[edge_1_start][edge_2_start]
    temp_score += distance_matrix[edge_1_end][edge_2_end]

    temp_solution ← solution
    REVERSE temp_solution[from cycle_position+1 to any_node_position+1]

    RETURN temp_score, temp_solution



 FUNCTION steepest_local_search_edges(solution, score, distance_matrix, weights):
     INPUT:
         solution - the starting point solution
         score - the initial score of the solution
         distance_matrix - matrix of distances between nodes
         weights - an array of weights associated with each node


     remaining_positions ← list of node positions outside solution
     moves ← []
     FOR cycle_position in [0, 1, …, length of solution]:
         FOR any_node_position in remaining_positions:
             APPEND (cycle_position, any_node) to moves

         FOR any_node_position in [cycle_position + 2, cycle_position + 3, …,
 number of nodes in solution]:
             IF cycle_position == 0 and any_node_position == number of nodes - 1:
                SKIP iteration

             APPEND (cycle_position, any_node_position) to moves
```

```
WHILE improvement is found:
    FOR cycle_position, any_node_position in moves:
        IF any_node_position IS BIGGER THAN length of solution
            tested_score, tested_solution ← node_inter_change(...)
        ELSE
            tested_score, tested_solution ← edge_intra_change(...)

        IF tested_score IS SMALLER THAN neighborhood_best_score:
            neighborhood_best_solution ← tested_solution
            neighborhood_best_score ← tested_score

    IF neighborhood_best_score IS SMALLER THAN score:
        solution ← neighborhood_best_solution
        score ← neighborhood_best_score

RETURN solution, score
```

3. Experiments

To quantify the performance of local search algorithms, each of the four algorithms was ran 200 times on a random solution and the solutions produced on each node by the best performing algorithm in the problem instance from previous assignments. In addition to objective function values, the run times of local searches were analyzed.

Method	TSPA av (min - max)	TSPB av (min - max)
Greedy LS Rand	85812 (78831 - 93289)	61000 (53759 - 69662)
Steepest LS Rand	87935 (75935 - 95175)	63036 (55323 - 70187)
Greedy LS Edges Rand	73781 (71507 - 76491)	48427 (45646 - 51763)
Steepest LS Edges Rand	73954 (70948 - 77934)	48366 (45576 - 51616)
Greedy LS Best	71627 (70687 - 72882)	45460 (43826 - 51301)
Steepest LS Best	71619 (70626 - 72950)	45415 (43826 - 50876)
Greedy LS Edges Best	71515 (70571 - 72460)	45040 (43790 - 50495)
Steepest LS Edges Best	71468 (70510 - 72614)	44976 (43921 - 50495)
Random	264301 (223539 - 308435)	213397 (179796 - 253866)
Nearest Neighbor Closest	85109 (83182 - 89433)	54390 (52319 - 59030)
Nearest Neighbor All	73180 (71179 - 75450)	45870 (44417 - 53438)

Greedy Cycle	72606 (71488 - 74350)	51345 (48765 - 57262)
Greedy Regret Cycle	115630 (105852 - 123171)	72656 (67568 - 77329)
Weighted Greedy Regret Cycle	72133 (71108 - 73395)	50882 (47144 - 55700)

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

The best scores achieved are visualized below.

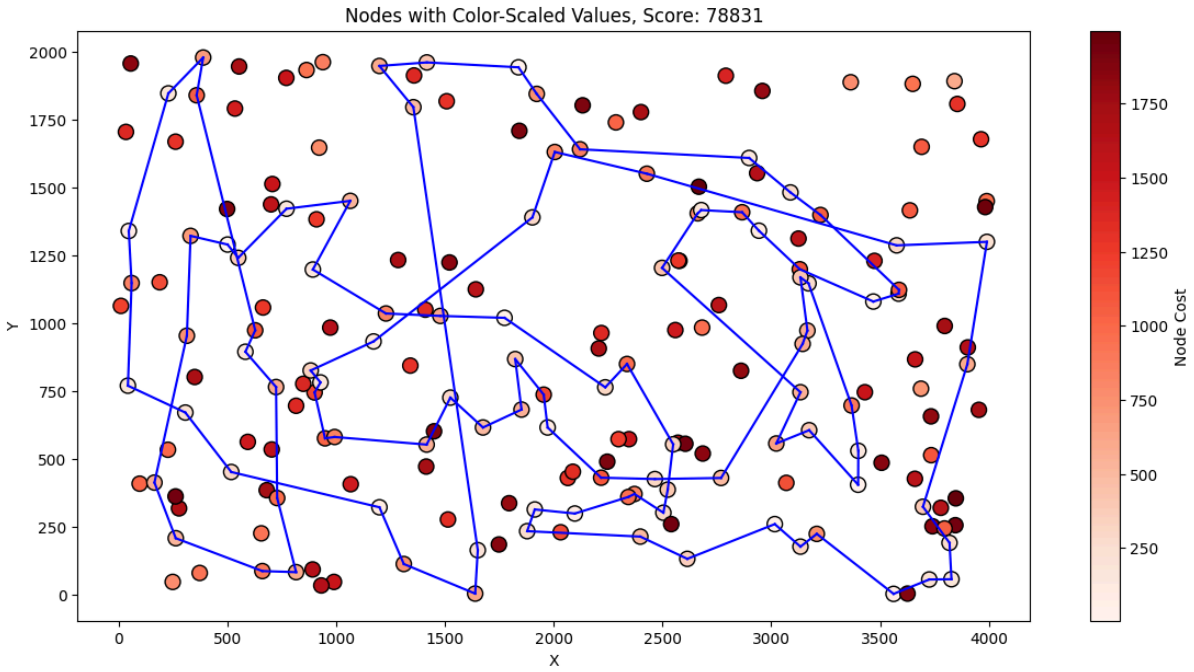

Fig 3. Visualization of the best solution found by the greedy local search algorithm with two-node exchange method as intra- and inter-route moves on the TSPA problem instance starting from a random solution.

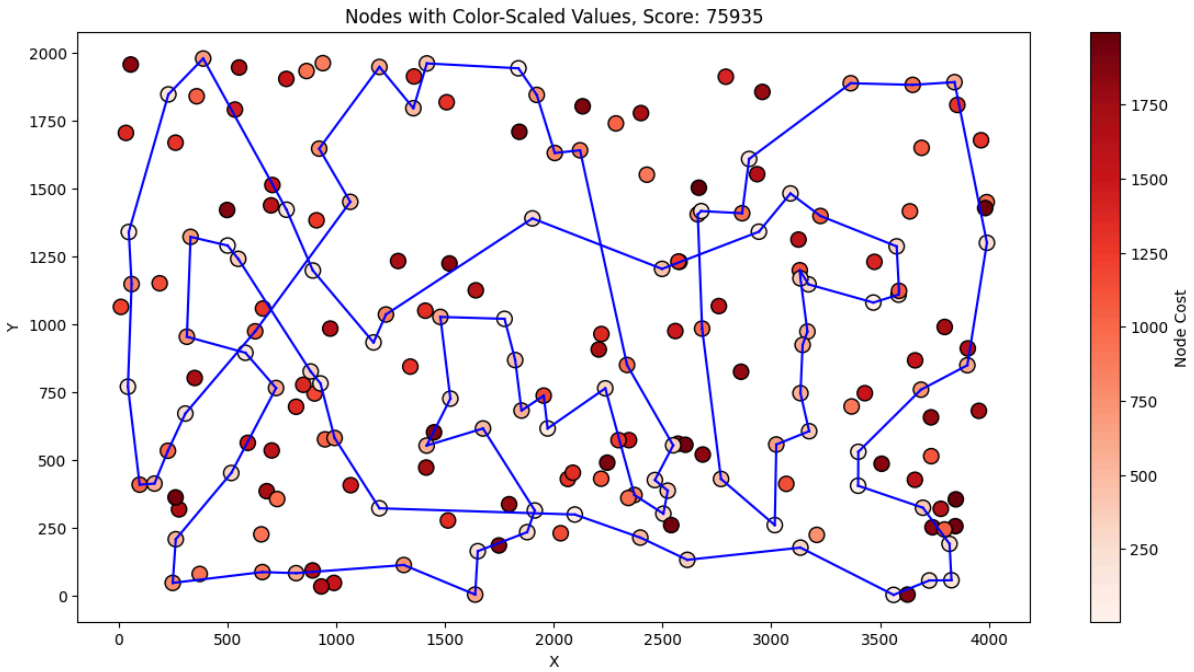

Fig 4. Visualization of the best solution found by the steepest local search algorithm with two-node exchange method as intra- and inter-route moves on the TSPA problem instance starting from a random solution.

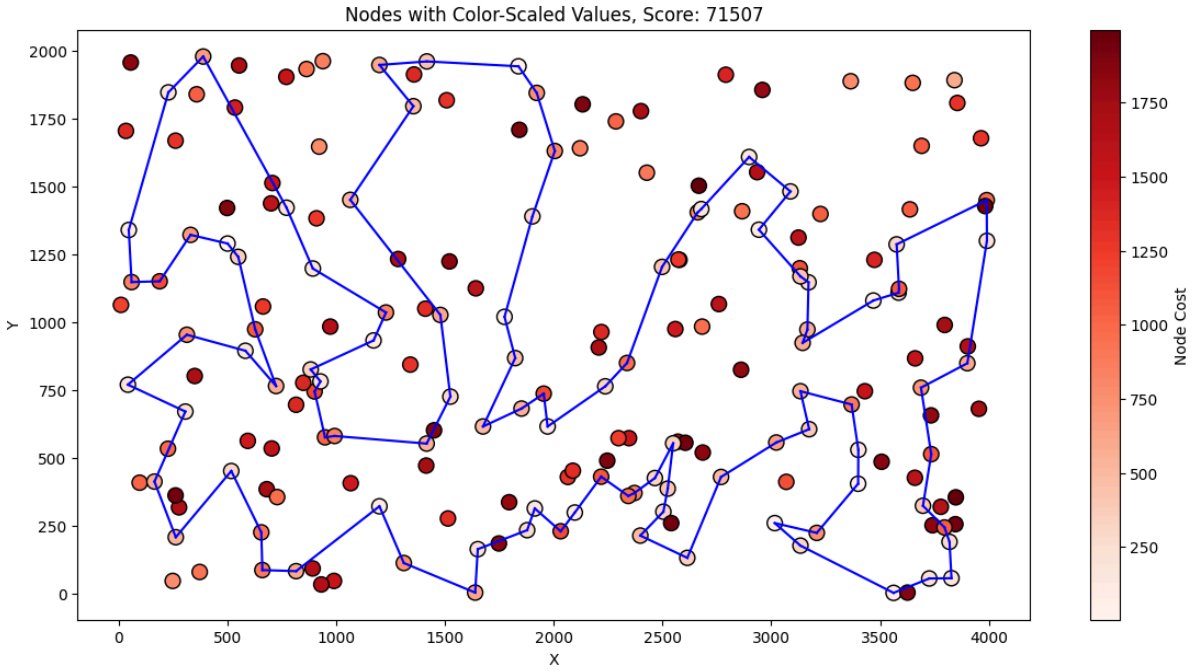

Fig 5. Visualization of the best solution found by the greedy local search algorithm with two-node exchange method as inter-route moves and two-edge exchange as intra-route moves on the TSPA problem instance starting from a random solution.

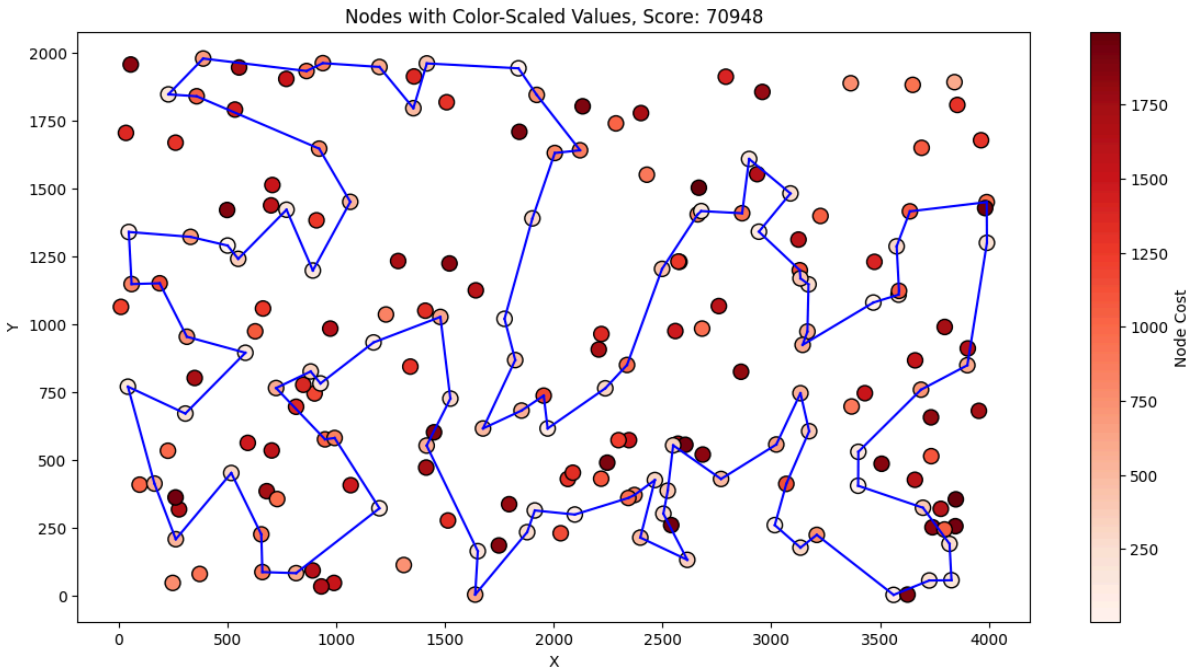

Fig 6. Visualization of the best solution found by the steepest local search algorithm with two-node exchange method as inter-route moves and two-edge exchange as intra-route moves on the TSPA problem instance starting from a random solution.

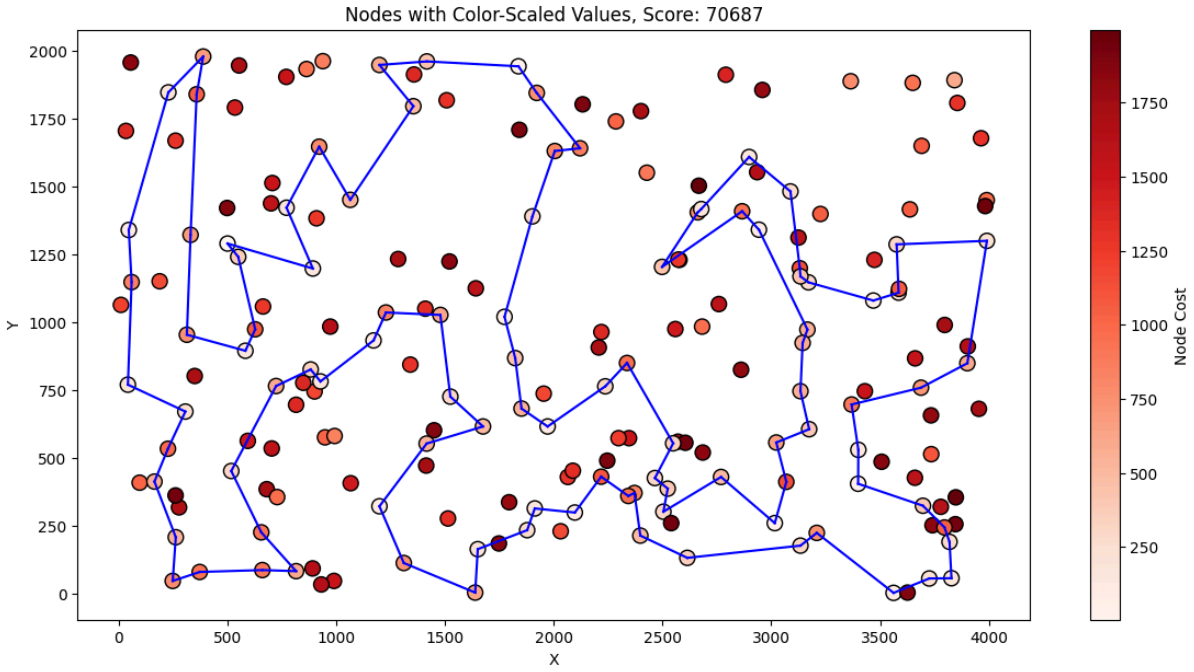

Fig 7. Visualization of the best solution found by the greedy local search algorithm with two-node exchange method as intra- and inter-route moves on the TSPA problem instance starting from a solution created by the weighted regret greedy cycle algorithm. Starting node: 0.

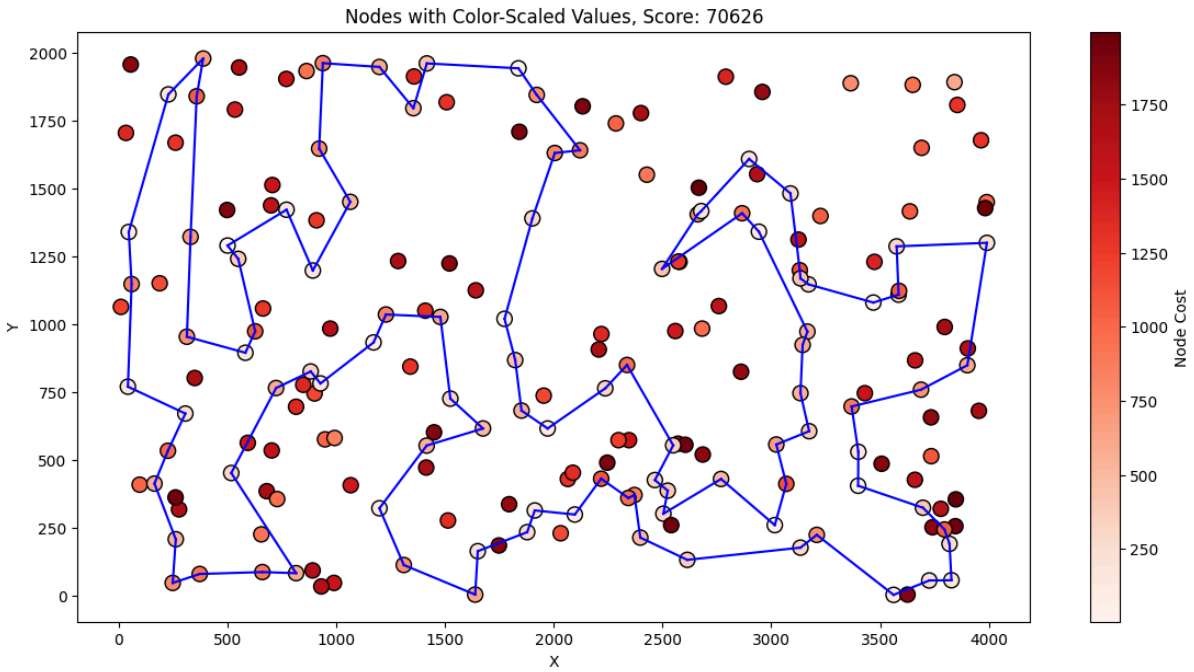

Fig 8. Visualization of the best solution found by the steepest local search algorithm with two-node exchange method as intra- and inter-route moves on the TSPA problem instance starting from a solution created by the weighted regret greedy cycle algorithm. Starting node: 153.

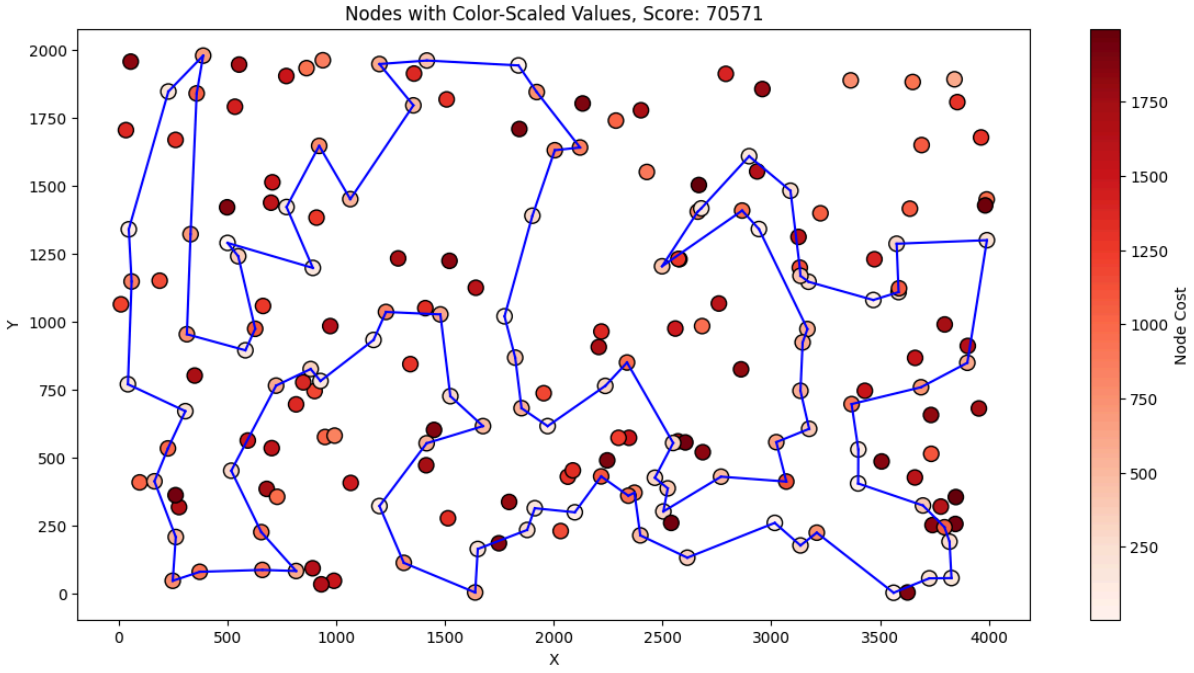

Fig 9. Visualization of the best solution found by the greedy local search algorithm with two-node exchange method as inter-route moves and two-edge exchange as intra-route moves on the TSPA problem instance starting from a solution created by the weighted regret greedy cycle algorithm. Starting node: 117.

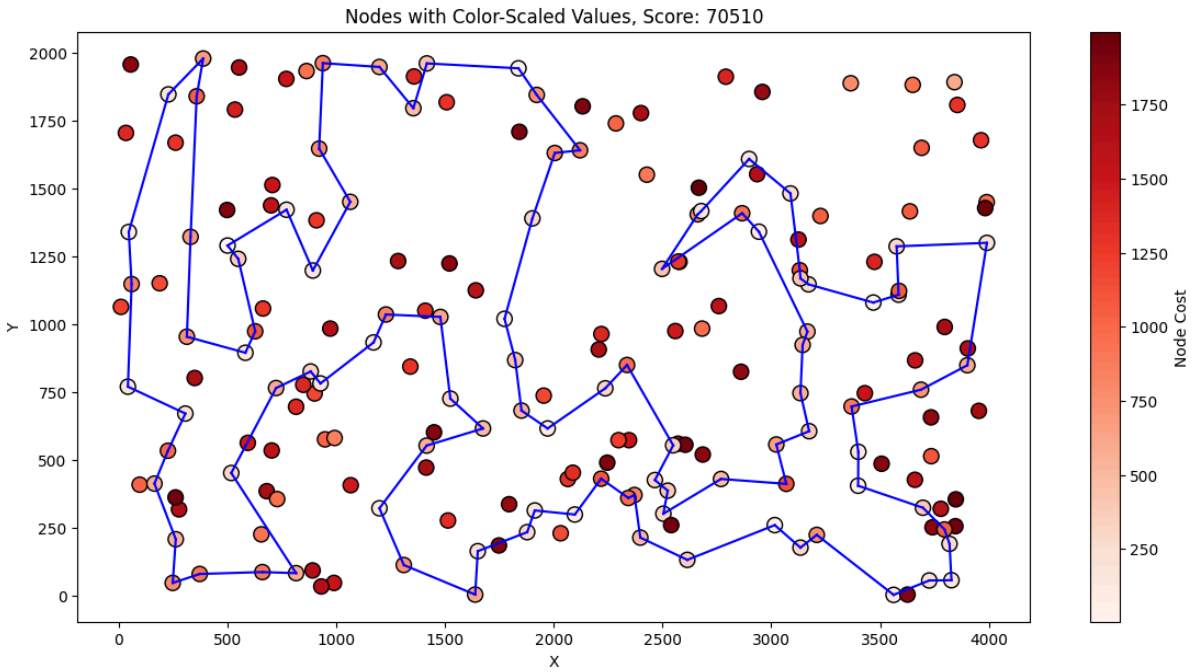

Fig 10. Visualization of the best solution found by the steepest local search algorithm with two-node exchange method as inter-route moves and two-edge exchange as intra-route moves on the TSPA problem instance starting from a solution created by the weighted regret greedy cycle algorithm. Starting node: 153.

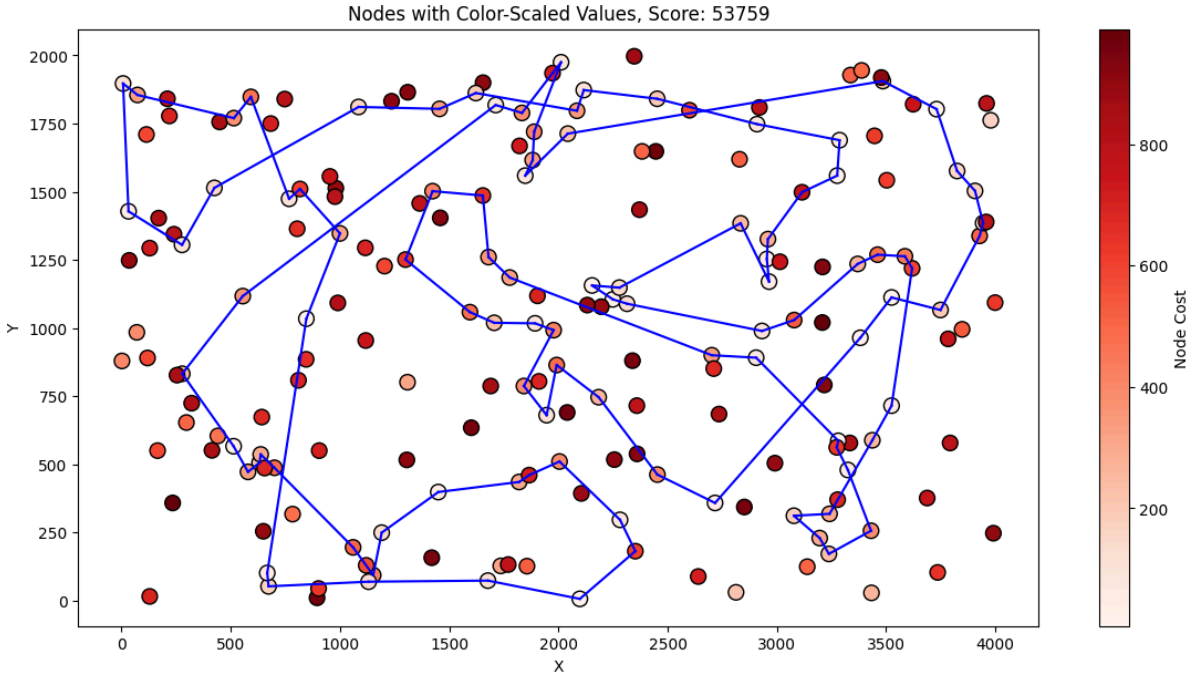

Fig 11. Visualization of the best solution found by the greedy local search algorithm with two-node exchange method as intra- and inter-route moves on the TSPB problem instance starting from a random solution.

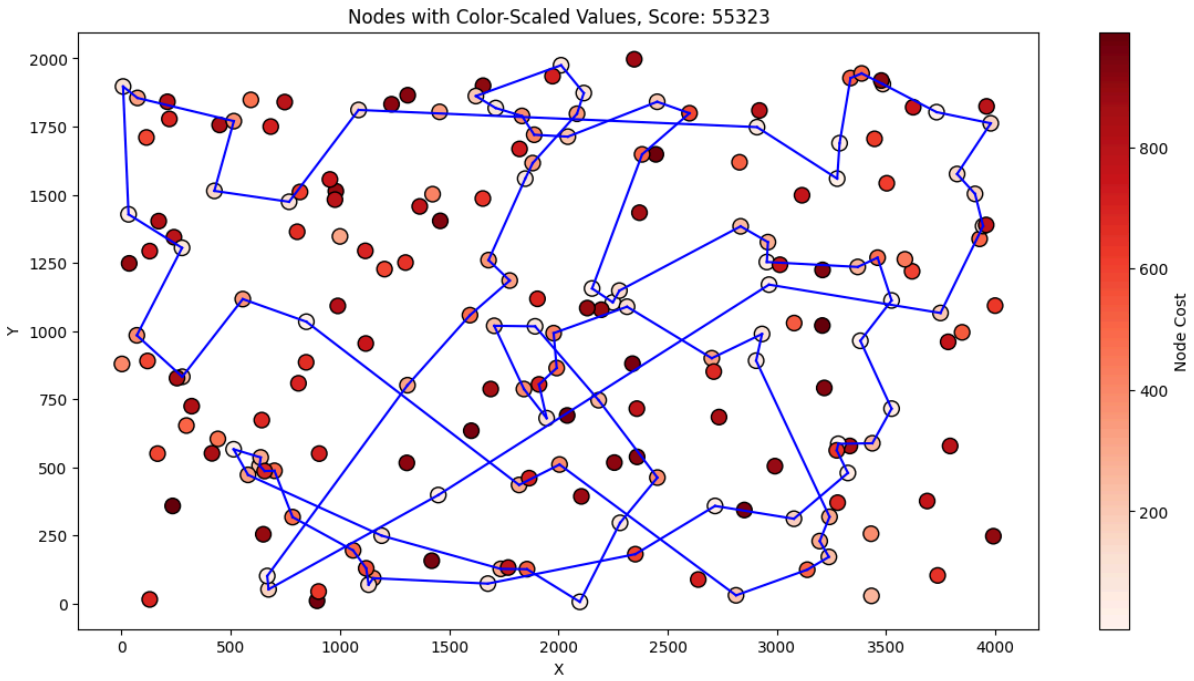

Fig 12. Visualization of the best solution found by the steepest local search algorithm with two-node exchange method as intra- and inter-route moves on the TSPB problem instance starting from a random solution.

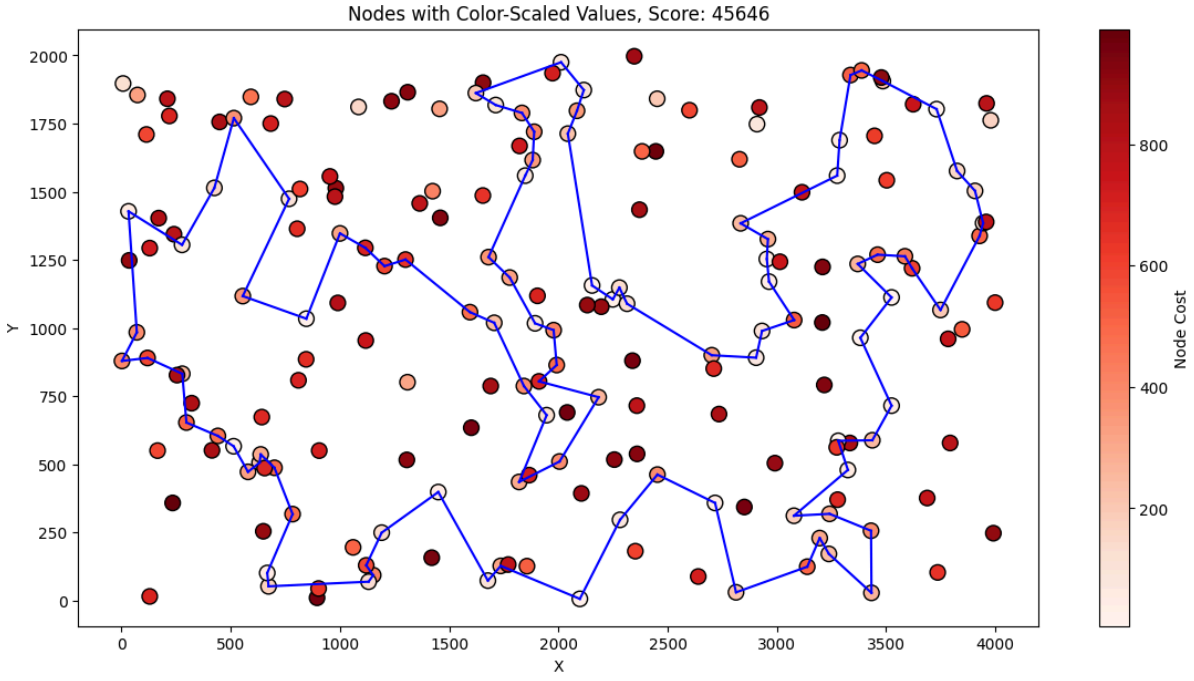

Fig 13. Visualization of the best solution found by the greedy local search algorithm with two-node exchange method as inter-route moves and two-edge exchange as intra-route moves on the TSPB problem instance starting from a random solution.

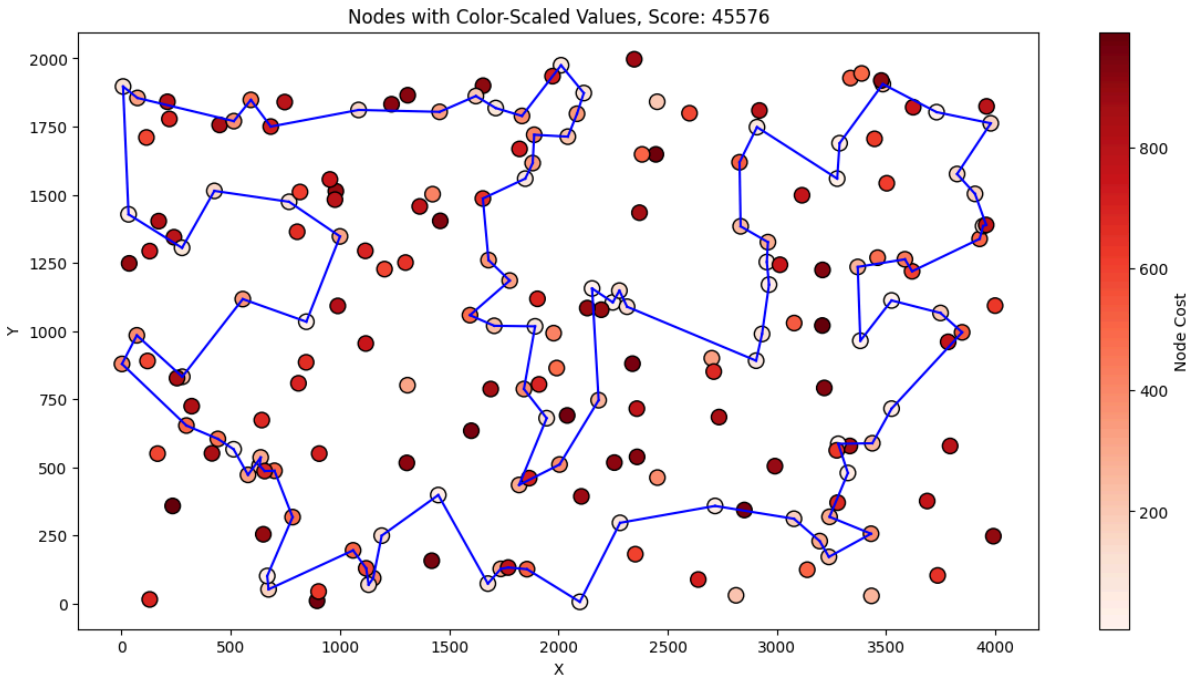

Fig 14. Visualization of the best solution found by the steepest local search algorithm with two-node exchange method as inter-route moves and two-edge exchange as intra-route moves on the TSPB problem instance starting from a random solution.

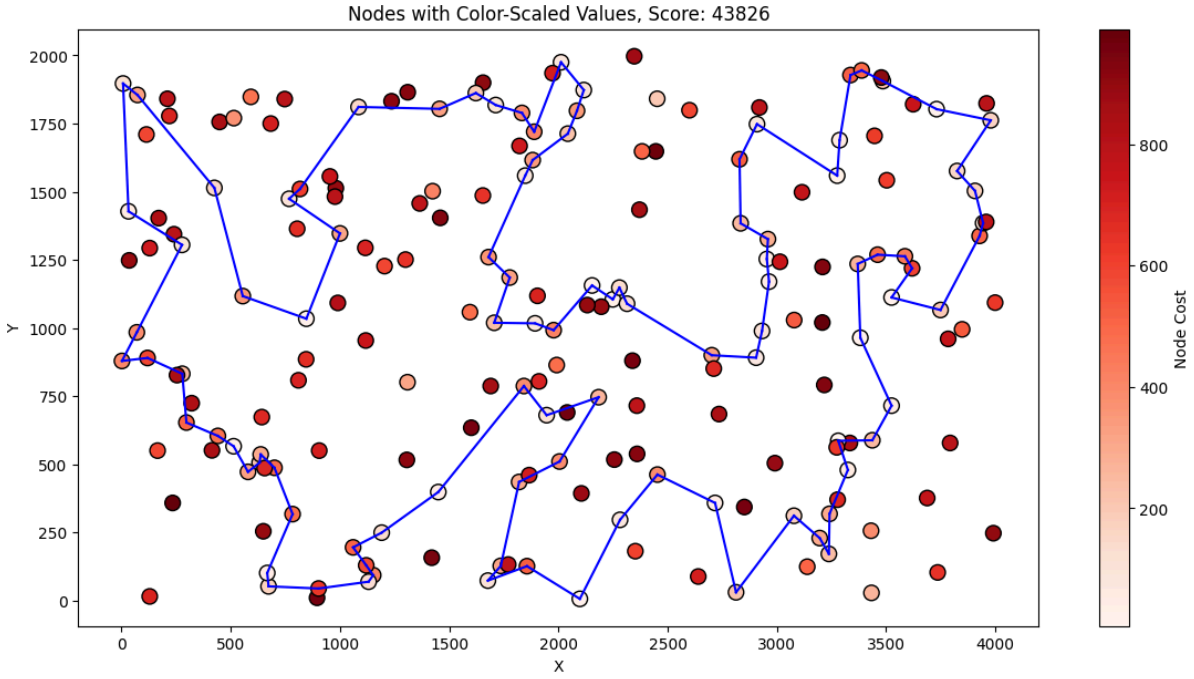

Fig 15. Visualization of the best solution found by the greedy local search algorithm with two-node exchange method as intra- and inter-route moves on the TSPB problem instance starting from a solution created by the greedy nearest neighbor algorithm with all insertion positions considered . Starting node: 78.

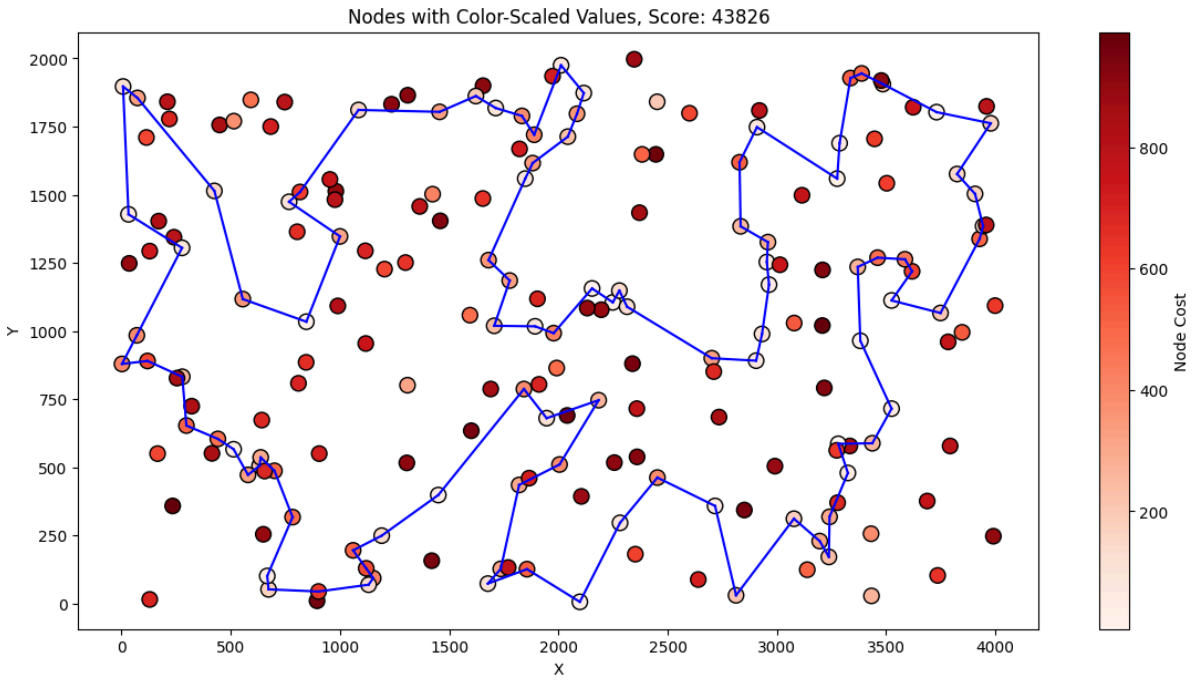

Fig 16. Visualization of the best solution found by the steepest local search algorithm with two-node exchange method as intra- and inter-route moves on the TSPB problem instance starting from a solution created by the greedy nearest neighbor algorithm with all insertion positions considered. Starting node: 1.

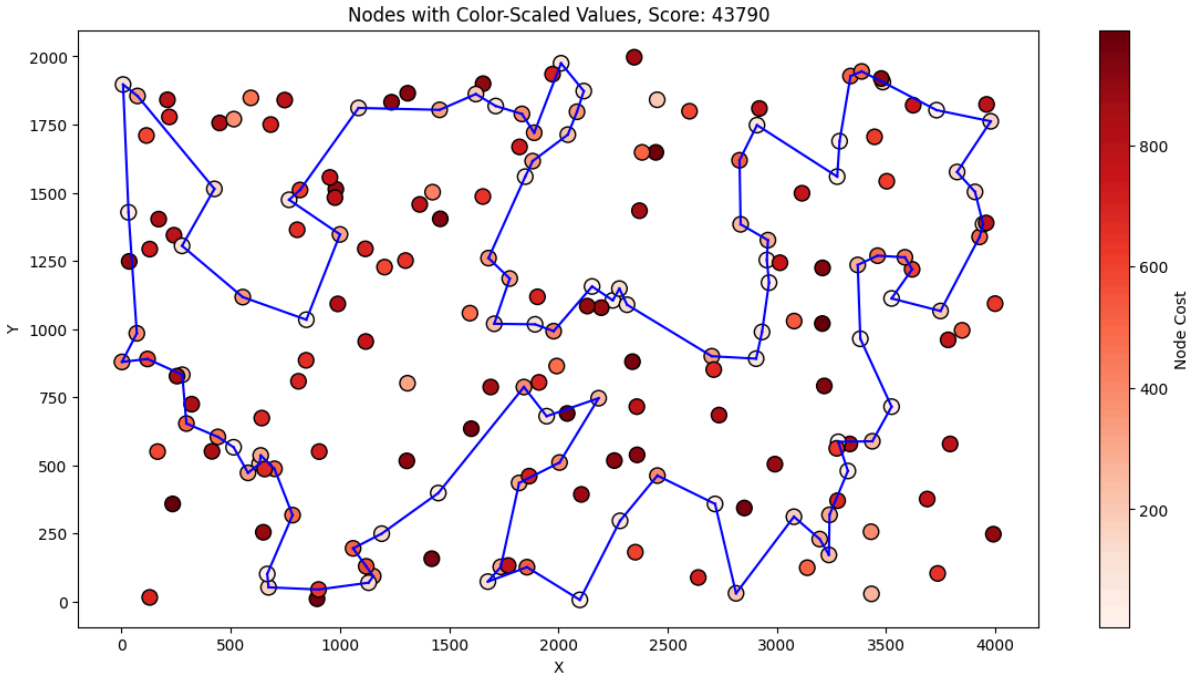

Fig 17. Visualization of the best solution found by the greedy local search algorithm with two-node exchange method as inter-route moves and two-edge exchange as intra-route moves on the TSPB problem instance starting from a solution created by the greedy nearest neighbor algorithm with all insertion positions considered. Starting node: 31.

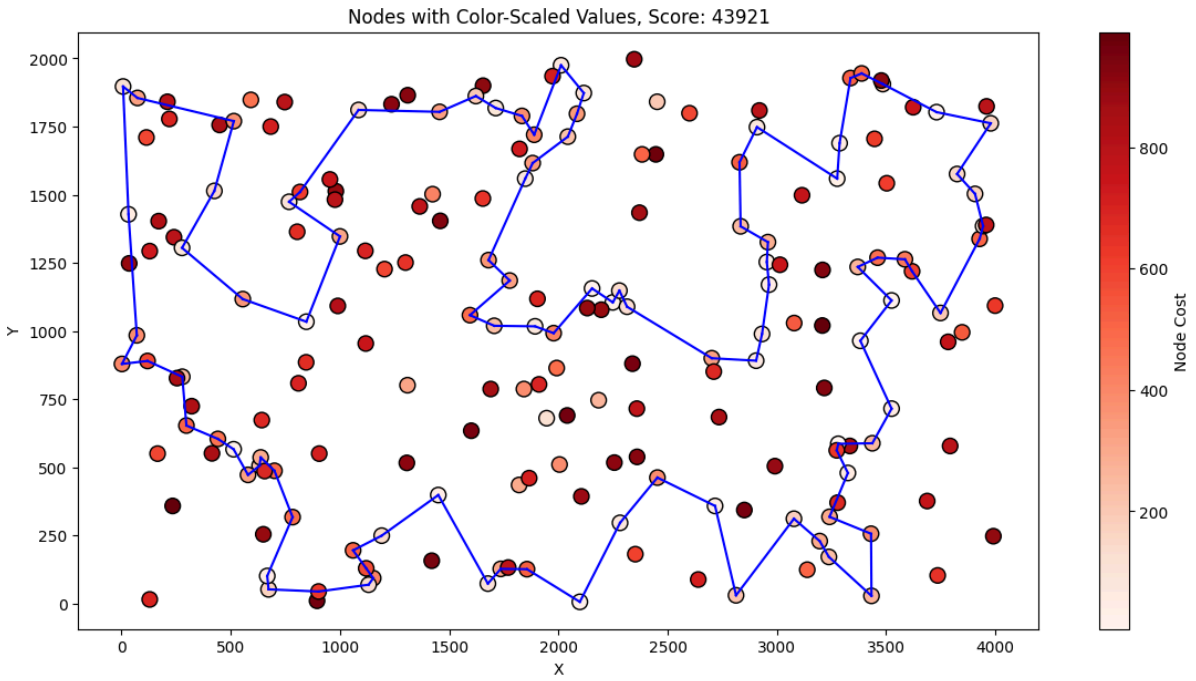

Fig 18. Visualization of the best solution found by the steepest local search algorithm with two-node exchange method as inter-route moves and two-edge exchange as intra-route moves on the TSPB problem instance starting from a solution created by the greedy nearest neighbor algorithm with all insertion positions considered. Starting node: 22.

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	Greedy LS Rand	78831	34, 146, 22, 18, 108, 69, 5, 42, 43, 35, 112, 4, 177, 54, 181, 159, 193, 41, 139, 46, 115, 118, 51, 176, 94, 124, 152, 1, 101, 26, 100, 53, 180, 154, 86, 75, 120, 44, 25, 16, 171, 175, 113, 31, 81, 90, 165, 15, 23, 137, 59, 116, 65, 47, 131, 149, 162, 151, 133, 79, 80, 122, 63, 121, 97, 2, 55, 52, 178, 106, 179, 145, 78, 92, 129, 57, 148, 9, 62, 102, 49, 3, 185, 40, 119, 138, 14, 144, 186, 89, 183, 143, 117, 0, 135, 70, 127, 123, 184, 160
	Steepest LS Rand	75935	167, 2, 120, 129, 92, 57, 55, 52, 178, 3, 106, 185, 40, 119, 165, 138, 14, 49, 148, 137, 118, 59, 115, 139, 108, 18, 22, 146, 34, 30, 54, 48, 160, 5, 46, 68, 117, 0, 143, 183, 89, 23, 186, 124, 152, 97, 1, 101, 26, 94, 63, 122, 79, 80, 176, 51, 151, 162, 133, 180, 154, 135, 70, 127, 112, 4, 10, 177, 184, 43, 42, 181, 159, 193, 41, 116, 65, 149, 123, 53, 86, 75, 44, 16, 171, 175, 113, 31, 78, 145, 196, 81, 90, 164, 7, 21, 144, 102, 62, 9
	Greedy LS Edges	71507	16, 44, 120, 25, 78, 145, 179, 57, 92, 129, 2, 75, 86, 101, 1, 152, 97, 26, 100, 121, 53, 158, 180, 154, 135, 70, 127, 123, 112, 4, 84, 184, 177, 54, 160, 34, 181, 42, 43, 5, 41, 193, 159, 195, 146,

	Rand		22, 18, 108, 139, 115, 118, 59, 116, 65, 47, 131, 149, 162, 151, 51, 46, 0, 117, 143, 183, 89, 23, 137, 176, 80, 133, 79, 122, 63, 94, 124, 148, 9, 62, 144, 14, 49, 178, 106, 52, 55, 185, 40, 119, 165, 27, 90, 81, 196, 157, 31, 56, 113, 175, 171
	Steepest LS Edges Rand	70948	4, 84, 184, 177, 54, 34, 160, 42, 181, 195, 146, 22, 159, 193, 41, 139, 115, 46, 68, 69, 18, 108, 140, 93, 117, 0, 143, 183, 89, 186, 23, 137, 176, 80, 133, 79, 122, 63, 94, 124, 148, 9, 62, 102, 144, 14, 49, 3, 178, 106, 52, 55, 185, 40, 119, 165, 39, 27, 90, 81, 196, 145, 78, 31, 113, 175, 171, 16, 25, 44, 120, 82, 92, 57, 129, 2, 152, 1, 101, 75, 86, 97, 26, 100, 53, 180, 154, 70, 135, 162, 151, 51, 59, 65, 116, 43, 131, 149, 123, 112
	Greedy LS Best	70687	117, 0, 46, 68, 139, 115, 193, 41, 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, 165, 40, 185, 106, 178, 3, 14, 144, 62, 9, 148, 102, 49, 52, 55, 57, 92, 129, 82, 120, 2, 101, 1, 97, 152, 124, 94, 63, 79, 80, 176, 137, 23, 186, 89, 183, 143
	Steepest LS Best	70626	0, 117, 93, 68, 46, 115, 139, 193, 41, 5, 42, 181, 159, 69, 108, 18, 22, 146, 34, 160, 48, 54, 177, 10, 190, 4, 112, 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, 165, 40, 185, 106, 178, 3, 14, 144, 62, 9, 148, 102, 49, 52, 55, 57, 92, 129, 82, 120, 2, 101, 1, 97, 152, 124, 94, 63, 79, 80, 176, 137, 23, 186, 89, 183, 143
	Greedy LS Edges Best	70571	117, 143, 183, 89, 186, 23, 137, 176, 80, 79, 63, 94, 124, 152, 97, 1, 101, 2, 82, 129, 92, 57, 55, 52, 49, 102, 148, 9, 62, 144, 14, 3, 178, 106, 185, 40, 165, 90, 81, 196, 179, 145, 78, 31, 56, 113, 175, 171, 16, 25, 44, 120, 75, 86, 26, 100, 121, 53, 180, 154, 135, 70, 127, 123, 162, 133, 151, 51, 118, 59, 65, 116, 43, 184, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 146, 22, 18, 108, 69, 159, 181, 42, 5, 41, 193, 115, 139, 68, 46, 0
	Steepest LS Edges Best	70510	117, 0, 143, 183, 89, 186, 23, 137, 176, 80, 79, 63, 94, 124, 152, 97, 1, 101, 2, 82, 129, 92, 57, 55, 52, 49, 102, 148, 9, 62, 144, 14, 3, 178, 106, 185, 40, 165, 90, 81, 196, 179, 145, 78, 31, 56, 113, 175, 171, 16, 25, 44, 120, 75, 86, 26, 100, 121, 53, 180, 154, 135, 70, 127, 123, 162, 133, 151, 51, 118, 59, 65, 116, 43, 184, 112, 4, 190, 10, 177, 54, 48, 160, 34, 146, 22, 18, 108, 69, 159, 181, 42, 5, 41, 193, 139, 115, 46, 68, 93
	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	71179	93, 117, 0, 143, 183, 89, 186, 23, 137, 176, 80, 79, 63, 94, 124, 152, 97, 1, 101, 2, 120, 129, 55, 49, 102, 148, 9, 62, 144, 14, 178, 106, 165, 90, 81, 196, 40, 119, 185, 52, 57, 92, 179, 145, 78, 31, 56, 113, 175, 171, 16, 25, 44, 75, 86, 26, 100, 53, 154, 16, 180, 135, 70, 127, 123, 162, 133, 151, 51, 118, 59, 65, 116, 43, 184, 35, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 146, 22, 18, 108, 69, 159, 181, 42, 5, 41, 193, 139, 115, 46, 68
	Greedy Cycle	71488	117, 0, 46, 68, 139, 193, 41, 115, 5, 42, 181, 159, 69, 108, 18, 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	Greedy LS Rand	53759	152, 155, 3, 15, 188, 6, 147, 122, 135, 63, 40, 107, 133, 10, 90, 191, 51, 121, 190, 80, 175, 36, 141, 97, 77, 82, 87, 21, 177, 5, 78, 45, 73, 54, 31, 193, 117, 1, 131, 169, 132, 161, 70, 13, 195, 168, 145, 28, 20, 148, 47, 94, 66, 179, 185, 86, 153, 81, 111, 144, 8, 104, 160, 33, 138, 182, 74, 134, 43, 139, 11, 143, 106, 176, 180, 113, 114, 127, 89, 163, 103, 194, 166, 22, 99, 130, 95, 128, 124, 35, 0, 29, 109, 34, 62, 18, 55, 174, 183, 140
	Steepest LS Rand	55323	40, 63, 135, 38, 1, 131, 121, 21, 82, 187, 165, 127, 89, 103, 106, 124, 143, 35, 160, 144, 56, 8, 104, 138, 33, 111, 81, 77, 141, 91, 61, 5, 193, 117, 54, 31, 164, 73, 136, 45, 142, 175, 78, 36, 97, 153, 163, 113, 180, 176, 194, 166, 86, 185, 130, 95, 18, 55, 34, 109, 0, 29, 189, 184, 155, 145, 13, 132, 169, 188, 70, 3, 15, 195, 168, 139, 11, 182, 25, 190, 80, 177, 62, 179, 94, 47, 148, 60, 20, 28, 149, 4, 140, 183, 152, 147, 90, 122, 133, 107
	Greedy LS Edges Rand	45646	16, 1, 156, 198, 117, 193, 31, 54, 73, 136, 190, 80, 175, 78, 142, 5, 177, 36, 61, 141, 77, 81, 153, 187, 165, 89, 127, 137, 114, 103, 163, 113, 176, 194, 166, 86, 185, 95, 130, 99, 22, 179, 66, 94, 47, 148, 20, 28, 149, 4, 140, 183, 34, 55, 18, 62, 128, 124, 106, 143, 35, 109, 0, 29, 145, 15, 3, 70, 188, 169, 132, 13, 195, 168, 139, 11, 33, 160, 144, 56, 111, 82, 21, 8, 104, 138, 182, 74, 118, 98, 51, 121, 131, 90, 133, 122, 135, 63, 38, 27
	Steepest	45576	163, 89, 127, 114, 103, 113, 176, 194, 166, 172, 179, 185, 86,

	LS Edges Rand		95, 99, 22, 66, 94, 154, 47, 148, 60, 20, 28, 140, 183, 152, 170, 34, 55, 18, 62, 124, 106, 35, 109, 0, 29, 111, 82, 21, 8, 104, 33, 138, 182, 11, 139, 43, 168, 195, 13, 145, 15, 3, 70, 132, 169, 188, 6, 147, 178, 10, 133, 107, 40, 63, 135, 122, 90, 51, 121, 131, 1, 38, 27, 156, 198, 117, 193, 54, 31, 164, 73, 136, 190, 80, 45, 142, 175, 78, 5, 177, 36, 61, 79, 91, 141, 77, 153
	Greedy LS Best	43826	121, 51, 90, 191, 147, 6, 188, 169, 132, 13, 70, 3, 15, 145, 195, 168, 139, 11, 138, 33, 160, 29, 0, 109, 35, 143, 106, 124, 62, 18, 55, 34, 170, 152, 183, 140, 4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 185, 22, 99, 130, 95, 86, 166, 194, 176, 113, 103, 127, 89, 163, 187, 153, 81, 77, 141, 91, 36, 61, 21, 82, 111, 8, 104, 177, 5, 45, 142, 78, 175, 162, 80, 190, 136, 73, 54, 31, 193, 117, 198, 156, 1, 16, 27, 38, 135, 63, 40, 107, 122, 131
	Steepest LS Best	43826	131, 122, 107, 40, 63, 135, 38, 27, 16, 1, 156, 198, 117, 193, 31, 54, 73, 136, 190, 80, 162, 175, 78, 142, 45, 5, 177, 104, 8, 111, 82, 21, 61, 36, 91, 141, 77, 81, 153, 187, 163, 89, 127, 103, 113, 176, 194, 166, 86, 95, 130, 99, 22, 185, 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, 160, 33, 138, 11, 139, 168, 195, 145, 15, 3, 70, 13, 132, 169, 188, 6, 147, 191, 90, 51, 121
	Greedy LS Edges Best	43790	121, 51, 90, 191, 147, 6, 188, 169, 132, 13, 70, 3, 15, 145, 195, 168, 139, 11, 138, 33, 160, 29, 0, 109, 35, 143, 106, 124, 62, 18, 55, 34, 170, 152, 183, 140, 4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 185, 22, 99, 130, 95, 86, 166, 194, 176, 113, 103, 127, 89, 163, 187, 153, 81, 77, 141, 91, 36, 61, 21, 82, 111, 8, 104, 177, 5, 45, 142, 78, 175, 162, 80, 190, 136, 73, 54, 31, 193, 117, 198, 156, 1, 16, 27, 38, 63, 40, 107, 122, 135, 131
	Steepest LS Edges Best	43921	131, 121, 51, 90, 191, 147, 6, 188, 169, 132, 13, 70, 3, 15, 145, 195, 168, 139, 11, 182, 138, 33, 160, 29, 0, 109, 35, 143, 106, 124, 62, 18, 55, 34, 170, 152, 183, 140, 4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 22, 99, 130, 95, 185, 86, 166, 194, 176, 180, 113, 103, 114, 137, 127, 89, 163, 187, 153, 81, 77, 141, 91, 61, 36, 177, 5, 45, 142, 78, 175, 162, 80, 190, 136, 73, 54, 31, 193, 117, 198, 156, 1, 16, 27, 38, 63, 40, 107, 133, 122, 135
	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	44417	121, 51, 90, 191, 147, 6, 188, 169, 132, 13, 70, 3, 15, 145, 195, 168, 139, 11, 138, 33, 160, 29, 0, 109, 35, 143, 106, 124, 62, 18, 55, 34, 170, 152, 183, 140, 4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 185, 22, 99, 130, 95, 86, 166, 194, 176, 113, 103, 127, 89,

			163, 187, 153, 81, 77, 141, 91, 36, 61, 21, 82, 111, 8, 104, 177, 5, 45, 142, 78, 175, 162, 80, 190, 136, 73, 54, 31, 193, 117, 198, 156, 1, 16, 27, 38, 135, 122, 63, 100, 107, 40
	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.

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)
Steepest LS Edges Rand	3.571 (2.978 - 4.168)	3.654 (2.976 - 4.364)
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)

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

4. Conclusions

Local search algorithms provide a benefit at a relatively low time cost when trying to improve a solution. They manage to significantly reduce the objective function of random solutions creating results and consistently provide small improvements to the results of the best greedy algorithms discussed so far. The usage of two-node exchange as intra-route moves creates, based on random solutions, cycles similar to the greedy nearest neighbor closest algorithm with many intersecting paths and overall achieves worse final scores than the two-edge exchange method which manages to turn random solutions into ones that resemble the greedy cycle or nearest neighbor with all insertion points considered. The two-edge exchange performs better in intra-route moves likely due to its better exploration capability. Using the best algorithms as starting points for local search provides consistently the best results. The steepest version of the local search algorithms provide on average a small improvement to the final score for good starting points and a deterioration for random starting points although the minimal scores found are often better for both good and random starting points in the steepest approach.

The run time experiments have shown that the steepest version of local search takes about three times longer to finish which is caused by the necessity to search through all possible moves which is especially costly in the early interactions of the algorithms where improvements can be found more easily. Using good solutions as starting points significantly reduces the amount of time necessary to complete the search due to less improvements being available and the algorithm quickly discovering a local optimum.