Assignment 9: Hybrid Evolutionary Algorithm

Mateusz Tabaszewski 151945 Bartłomiej Pukacki 151942

Source code:
https://github.com/MatTheTab/Evolutionary-Computation/blob/main/Assignment_9_Evolutionary_Metaheuristics/Assignment_9_Hybrid_Evolutionary_Algorithm.ipynb

1. Description of the Problem

The previous exploration of the local optima revealed that the locally optimal solutions oftentimes share a lot of nodes and edges, this can serve as a hint towards utilizing hybrid evolutionary algorithms that focus on preserving commonly occurring edges and nodes to achieve better scores than previously. As such, in this report, we will focus on designing and evaluating different evolutionary algorithms in the context of the redefined TSP. As previously, the task is to find a cycle composed of half of all available nodes with the smallest value of the objective function, with the function defined as the sum of the length of the cycle and the weight/cost of every node in the cycle. The visualizations related to the redefined TSP for both available instances TSPA and TSPB can be found in Figure 1 and Figure 2.

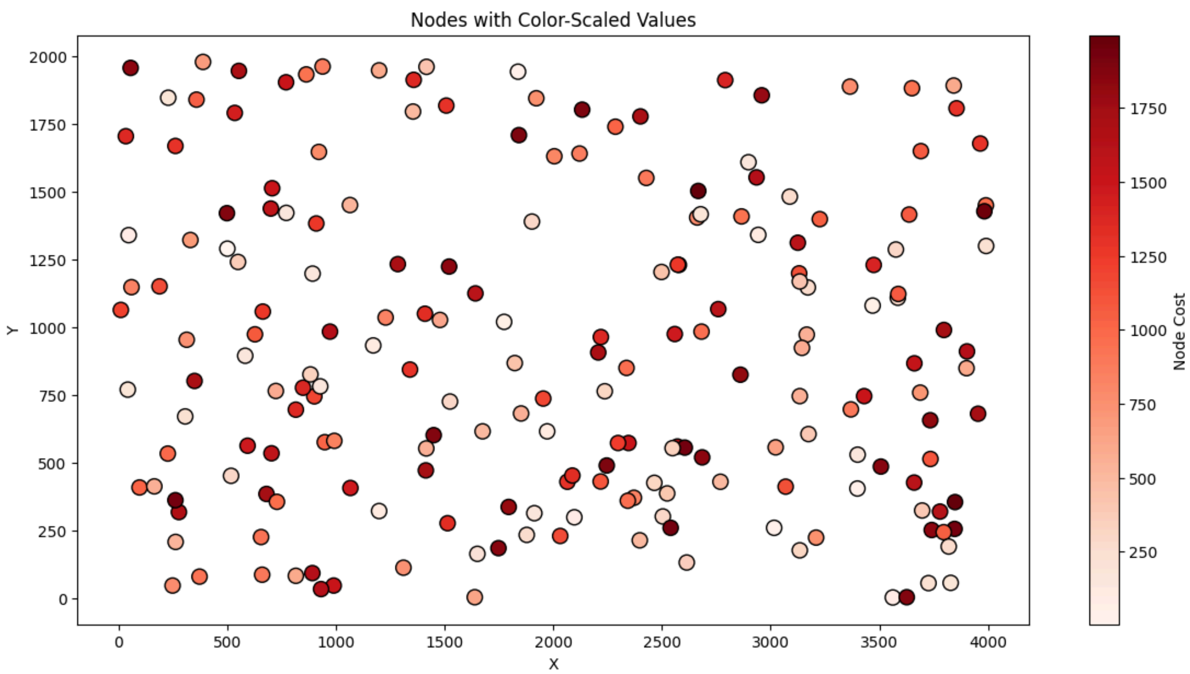

Fig 1. Visualization of the TSPA problem instance, each node's x and y locations on the plot correspond to their given x and y locations and the color intensity signifies the weight/cost of each node. The total length of the cycle and the sum of node weights should be minimized.

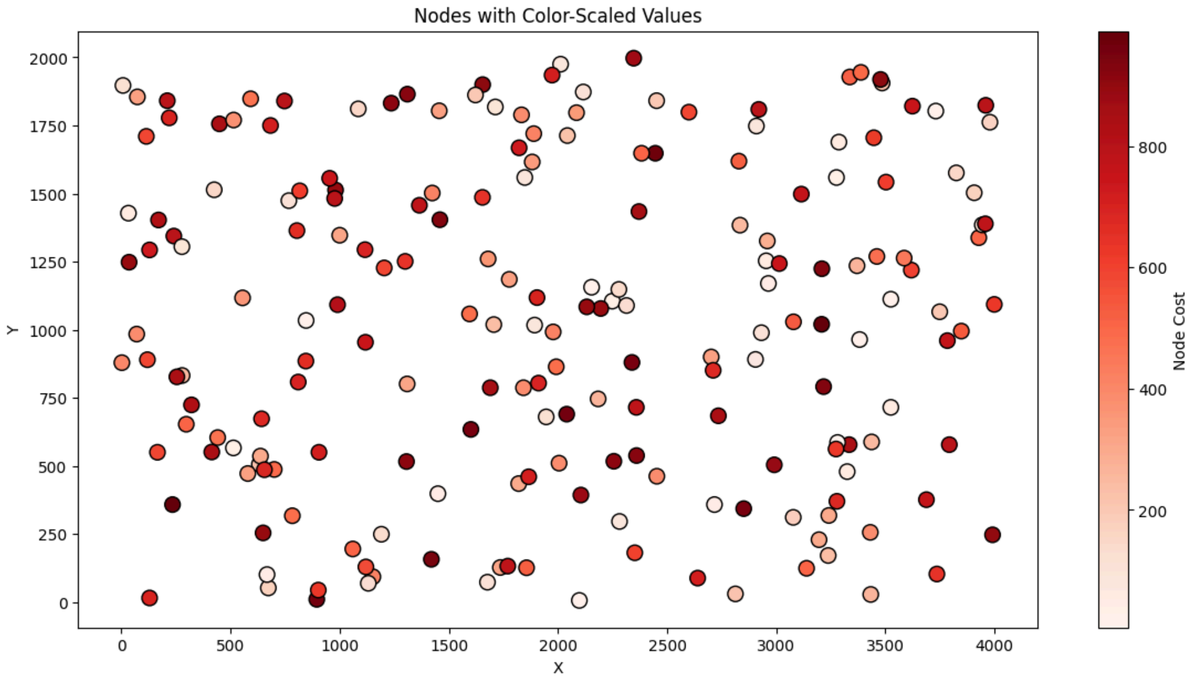

Fig 2. Visualization of the TSPB problem instance, each node's x and y locations on the plot correspond to their given x and y locations and the color intensity signifies the weight/cost of each node. The total length of the cycle and the sum of node weights should be minimized.

2. Algorithms

This time the tested algorithms were different variations of the **Hybrid Evolutionary Algorithms**. Each of the tested algorithms had a similar general idea behind it. The population of 20 initial solutions is produced using the steepest local search from a random solution for each of the 20 individuals. Then, for as long as there is time left, two solutions are chosen with a uniform probability, to be selected as parents. Then the recombination algorithm is applied, producing children. Afterward, an optional local search is run on children to find locally optimal solutions. At each point, only the best solutions are added to the population so that there is a constant elitism of the 20 best solutions within the population. Below, different variations of the cross-over operations are described:

Common + Random Recombination - For two parents, the common nodes and edges are found, and then two children are created. For the first one, the relative order of the nodes and edges is the same as for the first parent. The second child works the exact same way in relation to the second parent. The remaining, not filled positions in a cycle for both nodes, are added randomly from the remaining nodes. Local search can also be used on the offspring although it is optional.

Common + Heuristic Recombination - This method creates children by removing from the first parent the nodes and edges not present for the second parent to create the first child, and then analogously edges and nodes from the second parent that are not present in the first are removed and the second child is created. Both offspring are then completed by utilizing the heuristic weighted regret method. This method can also allow for the offspring to be further improved with local search, although it is optional.

Edge Recombination Crossover (ERX)- This operator constructs two child solutions by combining structural information from two parent solutions. An edge list is first created, associating each node with its neighbors from both parents. Starting from a random node, a child is constructed iteratively by selecting the next node based on the smallest number of neighbors remaining in the edge list, prioritizing nodes that are central to both parents. If no neighbors are available, a random unvisited node from the union of all parent nodes is chosen. Once a node is added to the child, it is removed as a neighbor from all other nodes' edge lists to avoid duplication. The process continues until the child contains half of all nodes. Finally, to ensure proper size and diversity, random unselected nodes from the complete set are swapped into the child at random positions. This method can also utilize local search to improve the offspring's score, although it is optional.

```
FUNCTION evolutionary_solution(distance_matrix, weights, allowed_time,
create_initial_population, pop_size, recombination, local_search):

    INPUT:
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node
        allowed_time - maximum time allowed for the algorithm's runtime
        create_initial_population - algorithm to create the initial population
        pop_size - the size of the population
        recombination - The defined recombination operator
        local_search - if the optional local search algorithm should be used

    num_iterations ← 0
    population ← create_initial_population(pop_size, distance_matrix, weights)

    WHILE there is time left:
       num_iterations += 1
       parent_1, parent_2 ← SAMPLE PARENTS FROM population
       child_1, child_2 ← recombination(parent_1, parent_2, all_nodes,
                                                   distance_matrix, weights)
       child_score_1 ← calculate_score(child_1, distance_matrix, weights)
      child_score_2 ← calculate_score(child_2, distance_matrix, weights)
      IF local_search:
          child_1 ← local_search(child_1, child_score_1, distance_matrix,
                                                                   weights)
          child_2 ← local_search(child_2, child_score_2, distance_matrix,
                                                                   weights)
      IF child_score_1 IS UNIQUE AND IS BETTER THAN WORST FROM population:
          ADD child_1 TO population AND DELETE WORST solution FROM population
      IF child_score_2 IS UNIQUE AND IS BETTER THAN WORST FROM population:
          ADD child_2 TO population AND DELETE WORST solution FROM population
    RETURN population, num_iterations
```

```
FUNCTION create_initial_population(pop_size, distance_matrix, weights):

    INPUT:
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node
        pop_size - the size of the population

       population ← EMPTY LIST
       FOR i IN RANGE OF pop_size:
           new_solution, new_score ← FIND RANDOM SOLUTION
           new_solution, new_score ← USE steepest local search ON new_solution
           ADD new_soltion TO population
       RETURN population

FUNCTION common_and_random_recombination(solution_1, solution_2, all_nodes,
distance_matrix, weights):
    INPUT:
        solution_1 - Parent solution 1
        solution_2 - Parent solution 2
        all_nodes - set of all nodes present in the problem
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node

    child_1 ← COMMON nodes AND edges FROM solution_1 AND solution_2 AT INDEX OF
                                                                  solution_1
    child_2 ← COMMON nodes AND edges FROM solution_1 AND solution_2 AT INDEX OF
                                                                  solution_2
    child_1 ← FILL REMAINING WITH RANDOM FROM all_nodes NOT IN child_1
    child_2 ← FILL REMAINING WITH RANDOM FROM all_nodes NOT IN child_2
    RETURN child_1, child_2

FUNCTION common_and_heuristic_recombination(solution_1, solution_2, all_nodes,
distance_matrix, weights):
    INPUT:
        solution_1 - Parent solution 1
        solution_2 - Parent solution 2
        all_nodes - set of all nodes present in the problem
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node

    child_1 ← nodes AND edges IN solution_1 AND NOT IN solution_2
    child_2 ← nodes AND edges IN solution_2 AND NOT IN solution_1
    child_1 ← FILL REMAINING WITH greedy weighted regret heuristic
    child_2 ← FILL REMAINING WITH greedy weighted regret heuristic
    RETURN child_1, child_2
```

```
FUNCTION ERX_recombination(solution_1, solution_2, all_nodes, distance_matrix,
weights):
    INPUT:
        solution_1 - Parent solution 1
        solution_2 - Parent solution 2
        all_nodes - set of all nodes present in the problem
        distance_matrix - matrix of distances between nodes
        weights - an array of weights associated with each node

    edge_list ← EMPTY EDGE LIST FOR ALL NODES
    edge_list ← ADD NEIGHBORS FROM solution_1 TO edge_list
    edge_list ← ADD NEIGHBORS FROM solution_2 TO edge_list
    edge_list ← ADD NEIGHBORS FROM solution_2 TO edge_list
    current_node_2 ← RANDOM NODE FROM solution_2
    child_1 ← BUILD CHILD FROM EDGE LIST STARTING WITH current_node_1
    child_2 ← BUILD CHILD FROM EDGE LIST STARTING WITH current_node_2
    remaining_nodes_1 ← all_nodes - NODES IN child_1
    remaining_nodes_2 ← all_nodes - NODES IN child_2
    RANDOMLY REPLACE A NODE IN child_1 WITH A NODE FROM remaining_nodes_1
    RANDOMLY REPLACE A NODE IN child_2 WITH A NODE FROM remaining_nodes_2
    RETURN child_1, child_2
```

3. Experiments

Experiments include all previous methods along with the Evolutionary Algorithm. Greedy methods were run 200 times for each instance while multi-start methods were run 20 times each. Other methods were run for the amount of time allowed to them, which was the same for all of these methods for the sake of comparison.

Method	TSPA av (min - max)	TSPB av (min - max)	Num Iterations (TSPA/TSPB)
Evolutionary (Common + Random Recombination + LS)	**69637 (69554 - 69692)**	**43897 (43811 - 43933)**	**2494/2104**
Evolutionary (Common + Random Recombination)	**74060 (72242 - 76036)**	**48800 (46931 - 50831)**	**892080/671469**
Evolutionary (Common + Heuristic	**72396 (71881 - 72708)**	**47405 (46611 - 48051)**	**101/82**

Recombination + LS)			
Evolutionary (Common + Heuristic Recombination)	**73600 (72050 - 75872)**	**48566 (47111 - 50888)**	**445/276**
Evolutionary (ERX Recombination + LS)	**70393 (69761 - 70635)**	**44757 (44287 - 44981)**	**190/200**
Evolutionary (ERX Recombination)	**74196 (73113 - 76574)**	**47880 (46856 - 48519)**	**59939/55978**
Large Neighborhood Search (optional LS)	69457 (69207 - 69821)	44133 (43873 - 44463)	387/357
Large Neighborhood Search (no LS)	69640 (69250 - 70805)	44361 (44174 - 44657)	403/391
MSLS	71340 (70919-71756)	45952 (45365-46428)	200/200
ILS - Partial Shuffle	69545 (69141 - 70200)	43952 (43448 - 44659)	1548/1495
Weighted Greedy Regret Cycle	72133 (71108-73395)	50882 (47144-55700)	
Steepest Delta Search	73910 (71118-78710)	48574 (46300-51342)	-
ILS - MST Perturbation	70133 (69246 - 71271)	44360 (43658 - 45019)	1582/1525
ILS - Coordinate Change	69882 (69460 - 70419)	44201 (43574 - 44853)	315/345
Steepest Candidate Search	77944 (73159-84951)	48497 (45342-52178)	**-**
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)	-

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

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

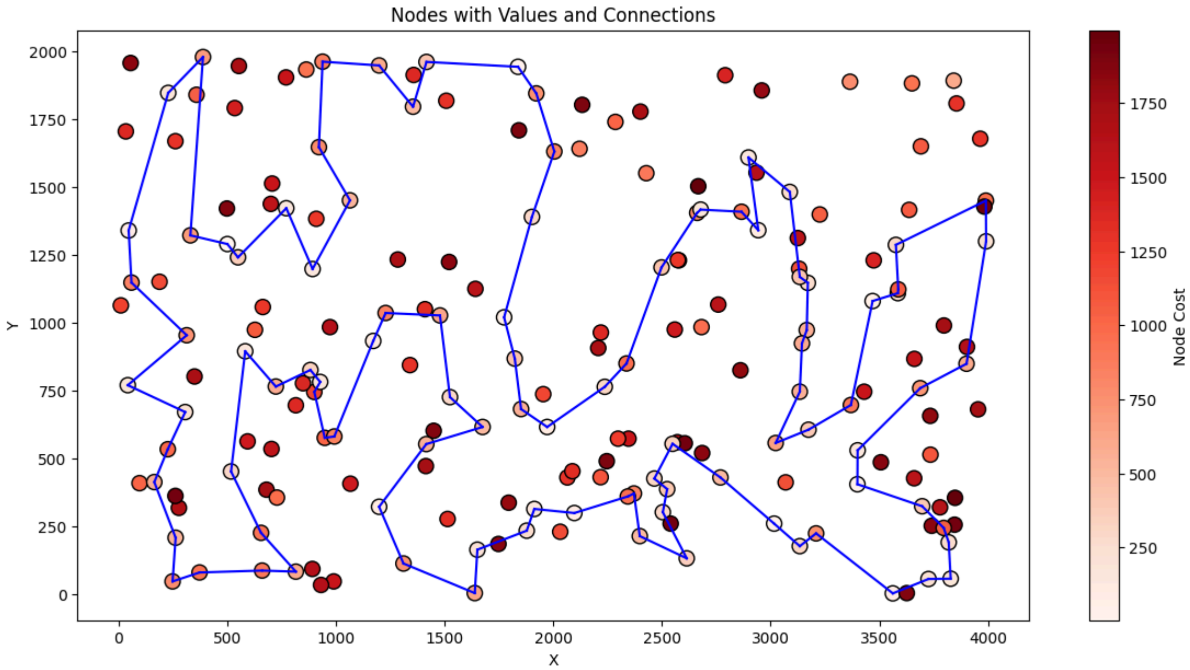

Fig 3. Visualization of the best solution found by the **Evolutionary (Common + Random Recombination + LS)** on the TSPA problem instance starting from a random solution.

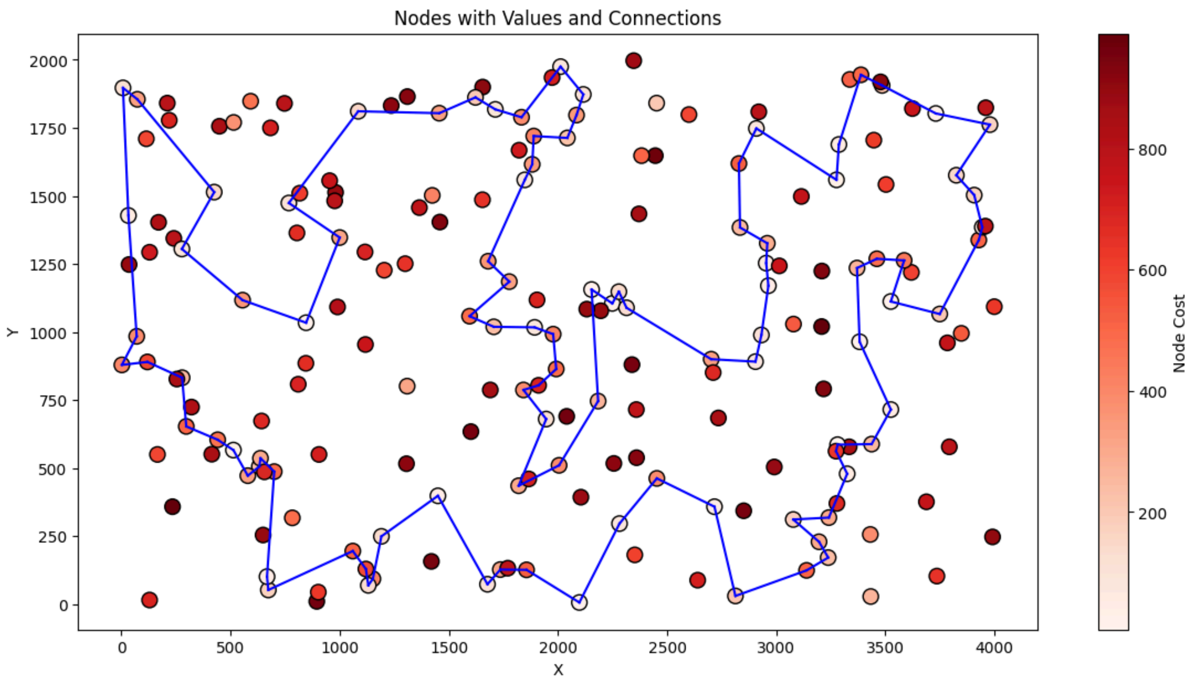

Fig 4. Visualization of the best solution found by the **Evolutionary (Common + Random Recombination)** on the TSPB problem instance starting from a random solution.

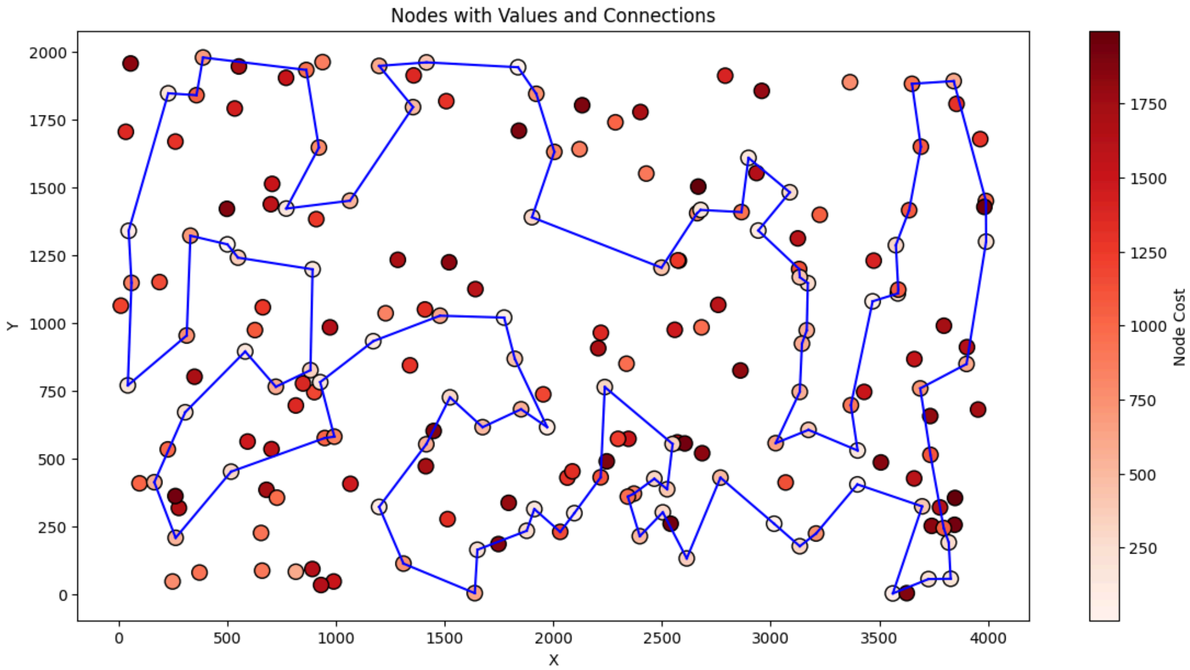

Fig 5. Visualization of the best solution found by the **Evolutionary (Common + Heuristic Recombination + LS)** on the TSPA problem instance starting from a random solution.

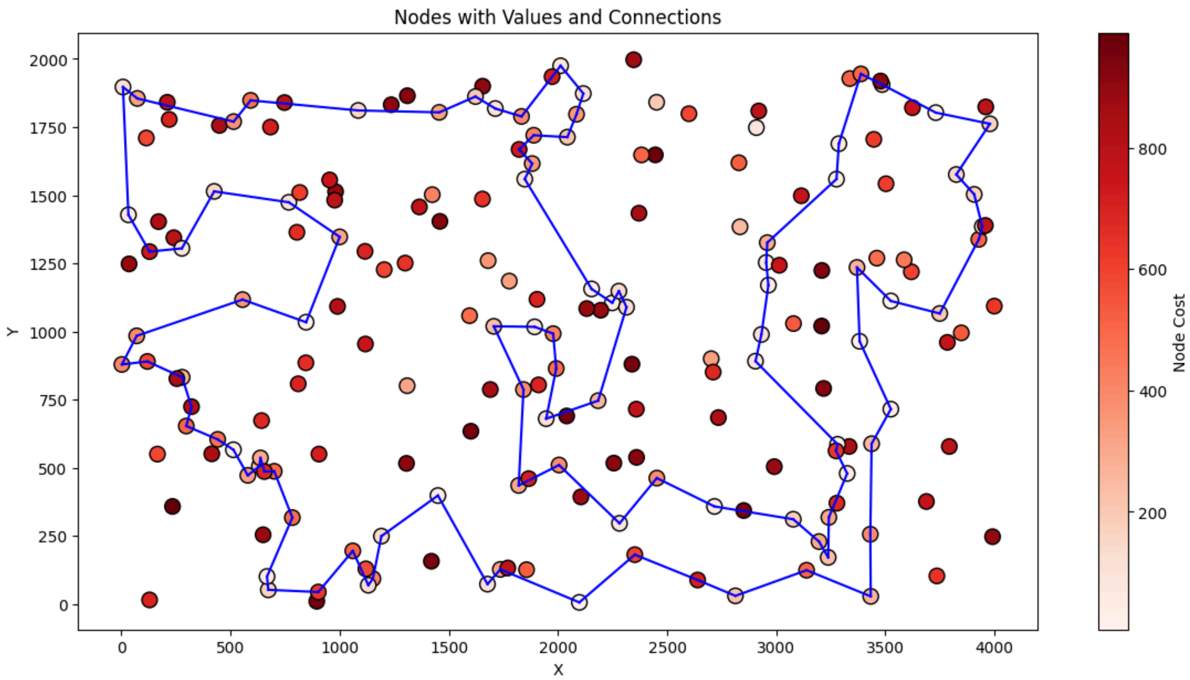

Fig 6. Visualization of the best solution found by the **Evolutionary (Common + Heuristic Recombination)** on the TSPB problem instance starting from a random solution.

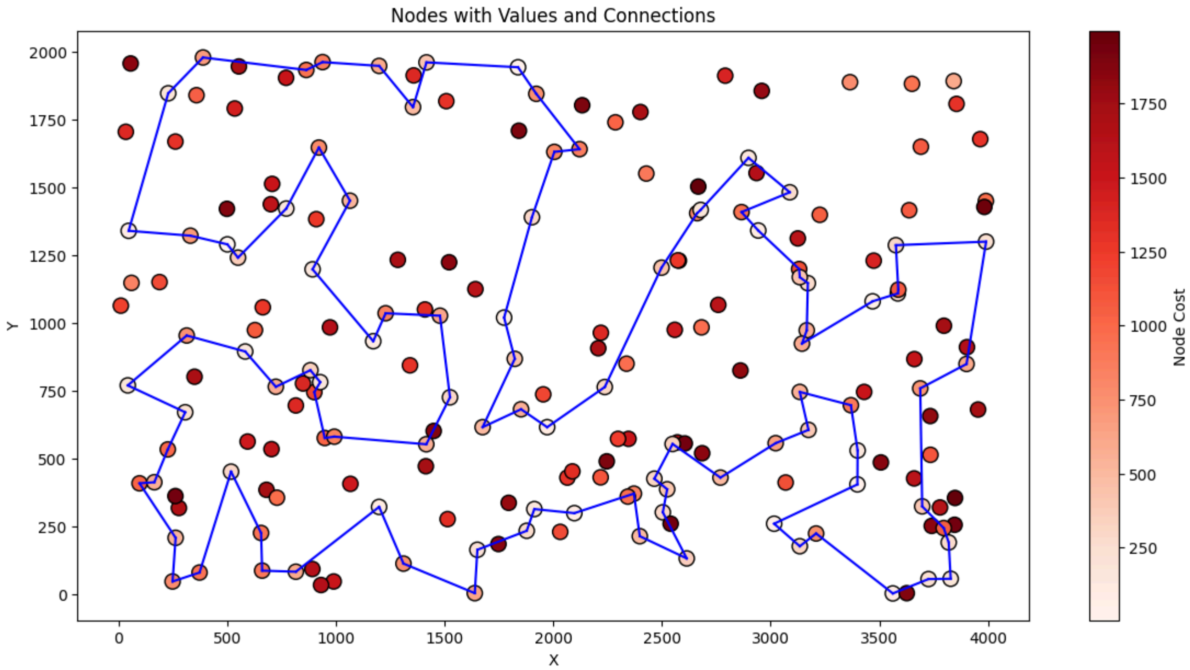

Fig 7. Visualization of the best solution found by the **Evolutionary (ERX Recombination + LS)** on the TSPA problem instance starting from a random solution.

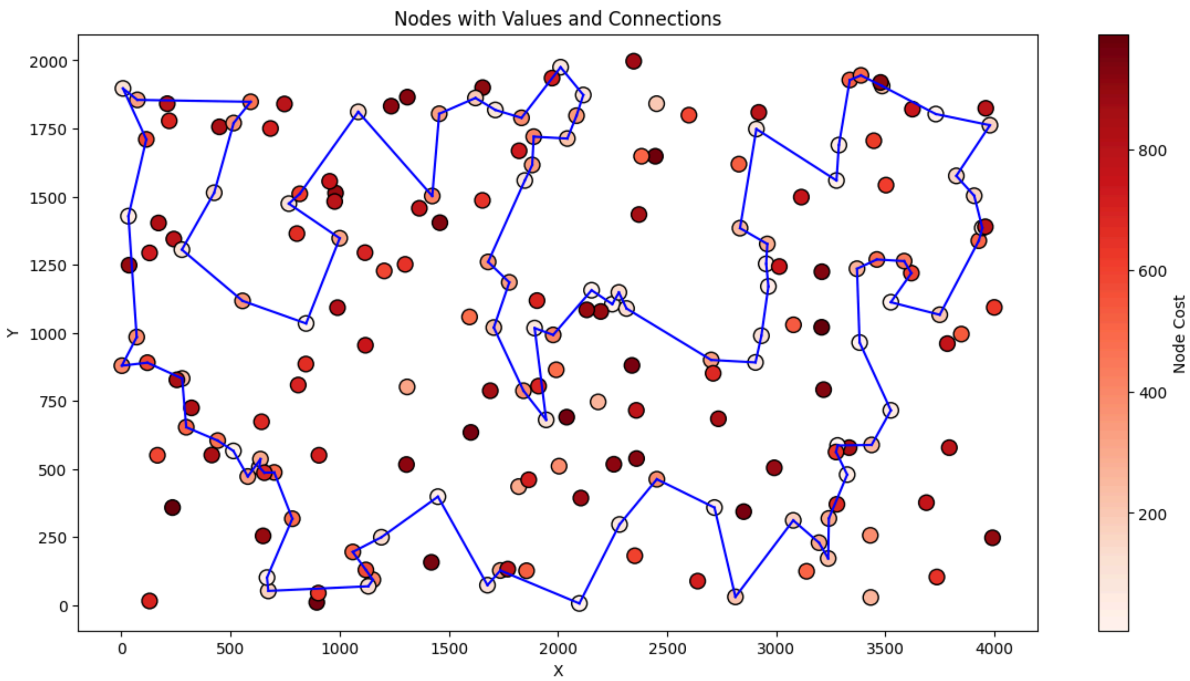

Fig 8. Visualization of the best solution found by the **Evolutionary (ERX Recombination)** on the TSPB problem instance starting from a random solution.

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	**Evolutionary (Common + Random Recombination + LS)**	**69554**	**119, 165, 27, 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, 149, 131, 47, 65, 116, 43, 42, 184, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 181, 146, 22, 18, 108, 159, 193, 41, 139, 115, 46, 68, 93, 117, 0, 143, 183, 89, 23, 137, 176, 80, 79, 63, 94, 124, 148, 9, 62, 102, 49, 144, 14, 178, 106, 52, 55, 57, 129, 92, 179, 185, 40**
	Evolutionary (Common + Random Recombination)	**72242**	**159, 193, 41, 139, 115, 59, 149, 131, 65, 116, 43, 42, 181, 34, 30, 54, 48, 160, 184, 177, 10, 190, 84, 4, 112, 127, 123, 162, 151, 51, 176, 80, 133, 79, 63, 136, 180, 135, 70, 154, 158, 53, 121, 100, 26, 86, 75, 101, 1, 97, 152, 2, 120, 44, 16, 171, 175, 113, 31, 78, 145, 92, 129, 57, 196, 81, 90, 27, 165, 40, 185, 55, 52, 106, 178, 3, 14, 144, 49, 102, 62, 9, 148, 167, 124, 94, 137, 89, 183, 143, 117, 0, 46, 68, 140, 108, 18, 22, 146, 195**
	Evolutionary (Common + Heuristic Recombination + LS)	**71881**	**1, 152, 94, 121, 53, 158, 180, 154, 135, 70, 127, 123, 162, 151, 133, 79, 63, 80, 176, 51, 59, 65, 149, 131, 184, 177, 54, 160, 42, 43, 116, 115, 41, 193, 159, 181, 34, 146, 22, 18, 69, 108, 140, 68, 139, 46, 0, 117, 143, 183, 89, 23, 137, 148, 9, 62, 102, 144, 14, 49, 3, 178, 106, 52, 55, 57, 129, 92, 145, 179, 185, 40, 119, 165, 39, 95, 7, 164, 27, 90, 81, 196, 157, 56, 113, 175, 171, 16, 31, 78, 25, 44, 120, 2, 75, 101, 86, 100, 26, 97**
	Evolutionary (Common + Heuristic Recombination)	**72050**	**31, 113, 175, 171, 16, 25, 44, 120, 82, 92, 57, 129, 2, 152, 97, 1, 101, 75, 86, 100, 26, 124, 94, 80, 176, 51, 151, 133, 79, 63, 121, 53, 158, 180, 154, 135, 70, 127, 123, 112, 4, 190, 10, 177, 54, 160, 184, 131, 149, 162, 118, 59, 65, 47, 116, 43, 42, 181, 34, 146, 22, 193, 41, 139, 115, 46, 68, 69, 18, 108, 140, 93, 117, 143, 183, 89, 23, 137, 148, 9, 62, 144, 14, 49, 3, 178, 106, 52, 55, 185, 40, 119, 165, 39, 27, 90, 81, 196, 145, 78**
	Evolutionary (ERX Recombination + LS)	**69761**	**34, 181, 42, 43, 116, 65, 47, 131, 149, 162, 151, 51, 118, 59, 115, 46, 68, 139, 41, 193, 159, 22, 18, 108, 140, 93, 117, 0, 143, 183, 89, 186, 23, 137, 176, 80, 133, 79, 63, 94, 148, 9, 62, 144, 14, 102, 49, 3, 178, 106, 52, 55, 185, 40, 119, 165, 90, 81, 196, 31, 56, 113, 175, 171, 16, 25, 44, 120, 78, 145, 179, 57, 92, 129, 2, 152, 97, 1, 101, 75, 86, 26, 53, 180, 154, 135, 70, 127, 123, 112, 4, 84, 184, 190, 10, 177, 30, 54, 48, 160**
	Evolutionary (ERX Recombination)	**73113**	**34, 103, 146, 22, 159, 193, 41, 5, 115, 139, 69, 18, 108, 68, 46, 0, 143, 183, 89, 186, 23, 137, 148, 9, 62, 144, 14, 102, 49, 3, 178, 106, 52, 185, 40, 119, 165, 39, 95, 7, 164, 58, 27, 90, 81, 196, 31, 56, 113, 175, 171, 16, 78, 145, 179, 55, 57, 92, 129, 25, 44, 120, 2, 152, 124, 94, 97, 1, 101, 75, 86, 53, 180, 154, 135, 151, 133, 79, 63, 80, 176, 51, 118, 59, 116, 65, 131, 149, 123, 112, 4, 84, 190, 10, 177, 184, 43, 42, 181, 160**
	Large Neighborhood Search (optional LS)	69207	42, 43, 116, 65, 59, 118, 51, 151, 133, 162, 123, 127, 70, 135, 154, 180, 53, 121, 100, 26, 86, 75, 101, 1, 97, 152, 2, 120, 44, 25, 16, 171, 175, 113, 56, 31, 78, 145, 92, 129, 57, 179, 196, 81, 90, 165, 40, 185, 55, 52, 106, 178, 49, 14, 144, 102, 62, 9, 148, 124, 94, 63, 79, 80, 176, 137, 23, 186, 89, 183, 143, 0, 117, 93, 140, 68, 46, 115, 139, 41, 193, 159, 69, 108, 18, 22, 146, 181, 34, 160, 48, 54, 177, 10, 190, 4, 112, 84, 35, 18

	Large Neighborhood Search (no LS)	69250	186, 23, 137, 176, 80, 79, 63, 94, 124, 148, 9, 62, 102, 144, 14, 49, 178, 106, 52, 55, 185, 40, 165, 90, 81, 196, 179, 57, 129, 92, 145, 78, 31, 56, 113, 175, 171, 16, 25, 44, 120, 2, 152, 97, 1, 101, 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, 181, 42, 5, 115, 46, 68, 139, 41, 193, 159, 146, 22, 18, 69, 108, 140, 93, 117, 0, 143, 183, 89
	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	**Evolutionary (Common + Random Recombination + LS)**	**43811**	**54, 73, 190, 80, 45, 142, 175, 78, 5, 177, 36, 61, 91, 141, 77, 81, 153, 187, 165, 127, 89, 163, 103, 113, 180, 176, 194, 166, 86, 95, 130, 99, 185, 179, 66, 94, 47, 148, 60, 20, 28, 149, 140, 183, 152, 170, 34, 55, 18, 62, 124, 106, 143, 35, 109, 0, 29, 111, 82, 21, 8, 104, 56, 144, 160, 33, 138, 182, 11, 139, 168, 195, 13, 145, 15, 3, 70, 132, 169, 188, 6, 147, 90, 51, 121, 131, 135, 122, 107, 40, 63, 38, 27, 16, 1, 156, 198, 117, 193, 31**
	Evolutionary (Common + Random Recombination)	**46931**	**138, 33, 160, 144, 56, 104, 8, 111, 29, 0, 35, 109, 155, 152, 170, 34, 55, 18, 62, 143, 106, 124, 128, 95, 183, 140, 28, 20, 148, 47, 94, 66, 179, 99, 185, 86, 166, 194, 176, 113, 26, 103, 114, 127, 89, 163, 187, 153, 81, 77, 141, 61, 36, 175, 78, 5, 177, 25, 112, 19, 54, 31, 73, 136, 80, 190, 193, 117, 198, 156, 1, 16, 27, 38, 135, 63, 40, 107, 122, 90, 125, 131, 121, 51, 118, 74, 134, 147, 6, 188, 169, 132, 70, 3, 15, 145, 13, 195, 168, 11**
	Evolutionary (Common	**46611**	**0, 109, 35, 111, 8, 144, 160, 33, 138, 104, 21, 82, 77, 81, 153, 163, 89, 127, 103, 113, 180, 176, 106, 124, 62, 18, 55, 183, 140, 149, 28, 20, 60, 148, 47, 94, 66, 179, 185, 95, 86, 166,**

	+ Heuristic Recombination + LS)		**194, 114, 137, 165, 187, 146, 97, 141, 61, 36, 177, 5, 78, 175, 45, 162, 80, 190, 136, 73, 164, 54, 31, 193, 117, 198, 156, 24, 1, 16, 27, 38, 131, 121, 51, 90, 122, 135, 102, 63, 40, 107, 133, 10, 147, 6, 188, 169, 132, 70, 3, 15, 145, 13, 126, 195, 168, 29**
	Evolutionary (Common + Heuristic Recombination)	**47111**	**145, 15, 70, 3, 189, 155, 184, 152, 183, 140, 4, 149, 28, 59, 20, 60, 148, 47, 94, 179, 185, 130, 95, 55, 34, 18, 62, 124, 106, 86, 166, 194, 176, 180, 113, 103, 127, 165, 89, 163, 153, 81, 77, 141, 36, 61, 21, 82, 111, 159, 143, 35, 109, 0, 29, 160, 144, 56, 8, 104, 33, 11, 139, 138, 182, 25, 177, 5, 78, 175, 80, 190, 73, 54, 31, 193, 117, 198, 1, 63, 135, 131, 19, 112, 121, 125, 51, 120, 191, 90, 122, 133, 147, 6, 188, 169, 132, 13, 195, 168**
	Evolutionary (ERX Recombination + LS)	**44287**	**8, 33, 160, 29, 0, 109, 35, 143, 106, 124, 62, 18, 55, 34, 152, 183, 140, 4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 185, 22, 99, 130, 95, 86, 166, 194, 176, 180, 113, 103, 127, 89, 163, 187, 153, 81, 77, 141, 61, 36, 177, 5, 45, 142, 78, 175, 80, 190, 136, 73, 164, 31, 54, 193, 117, 198, 156, 1, 16, 27, 38, 63, 100, 40, 107, 10, 133, 122, 135, 131, 121, 51, 90, 191, 147, 134, 6, 188, 169, 132, 70, 3, 15, 145, 13, 195, 168, 139, 11, 138, 104**
	Evolutionary (ERX Recombination)	**46856**	**153, 187, 165, 127, 89, 163, 103, 114, 113, 180, 176, 194, 166, 86, 95, 185, 179, 94, 47, 148, 20, 140, 183, 152, 34, 55, 18, 62, 124, 106, 143, 111, 8, 82, 87, 21, 104, 56, 144, 0, 35, 109, 29, 168, 195, 145, 15, 3, 70, 132, 169, 188, 6, 147, 71, 191, 90, 51, 134, 139, 11, 160, 33, 138, 182, 25, 158, 19, 112, 121, 131, 135, 122, 133, 107, 40, 63, 1, 156, 198, 117, 193, 31, 54, 73, 136, 190, 80, 45, 142, 175, 78, 5, 177, 36, 61, 91, 141, 77, 81**
	Large Neighborhood Search (optional LS)	43873	33, 160, 144, 104, 8, 21, 82, 111, 29, 0, 109, 35, 143, 106, 124, 62, 18, 55, 34, 170, 152, 183, 140, 4, 149, 28, 20, 60, 148, 47, 94, 66, 172, 179, 22, 99, 130, 95, 185, 86, 166, 194, 176, 113, 103, 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, 135, 122, 131, 121, 51, 90, 147, 6, 188, 169, 132, 13, 70, 3, 15, 145, 195, 168, 43, 139, 11, 138
	Large Neighborhood Search (no LS)	44174	193, 54, 31, 73, 136, 190, 80, 162, 175, 78, 5, 177, 25, 182, 138, 139, 11, 33, 160, 144, 104, 8, 82, 21, 36, 61, 91, 141, 77, 81, 153, 187, 163, 89, 127, 103, 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, 168, 195, 145, 15, 3, 70, 13, 132, 169, 188, 6, 147, 51, 121, 131, 90, 133, 107, 40, 63, 122, 135, 38, 27, 16, 1, 156, 198, 117
	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)
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)
Steepest Candidate Search	0.584 (0.479 - 0.705)	0.562 (0.481 - 0.693)
MSLS	97.988 (92.859 - 104.159)	90.370 (83.360 - 98.981)
ILS	97.988	90.370
Large Neighborhood Search	97.988	90.370
Evolutionary	**97.988**	**90.370**

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

To allow for a more detailed analysis of different crossover operators and the learning process of the utilized Evolutionary Algorithms, we have conducted further visualizations and experiments. However, some of these Figures required logging the full population history which could slow down the algorithm. As such, these tests were run separately to not negatively influence the initial scores and their results. This is also why some results may not be consistent with earlier-presented values.

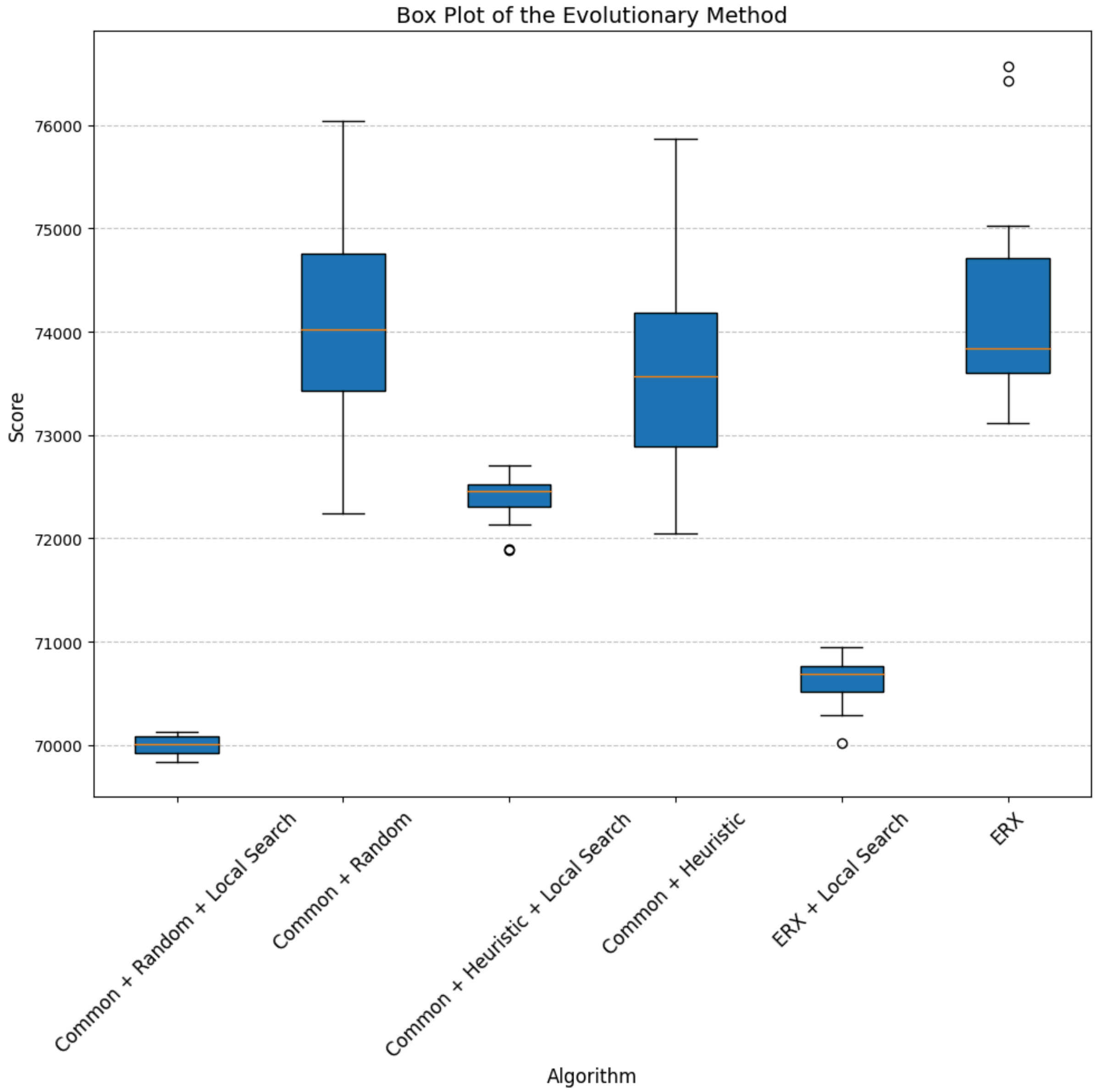

Fig 9. Comparison of performance of different **Hybrid Evolutionary Algorithms** on the TSPA instance.

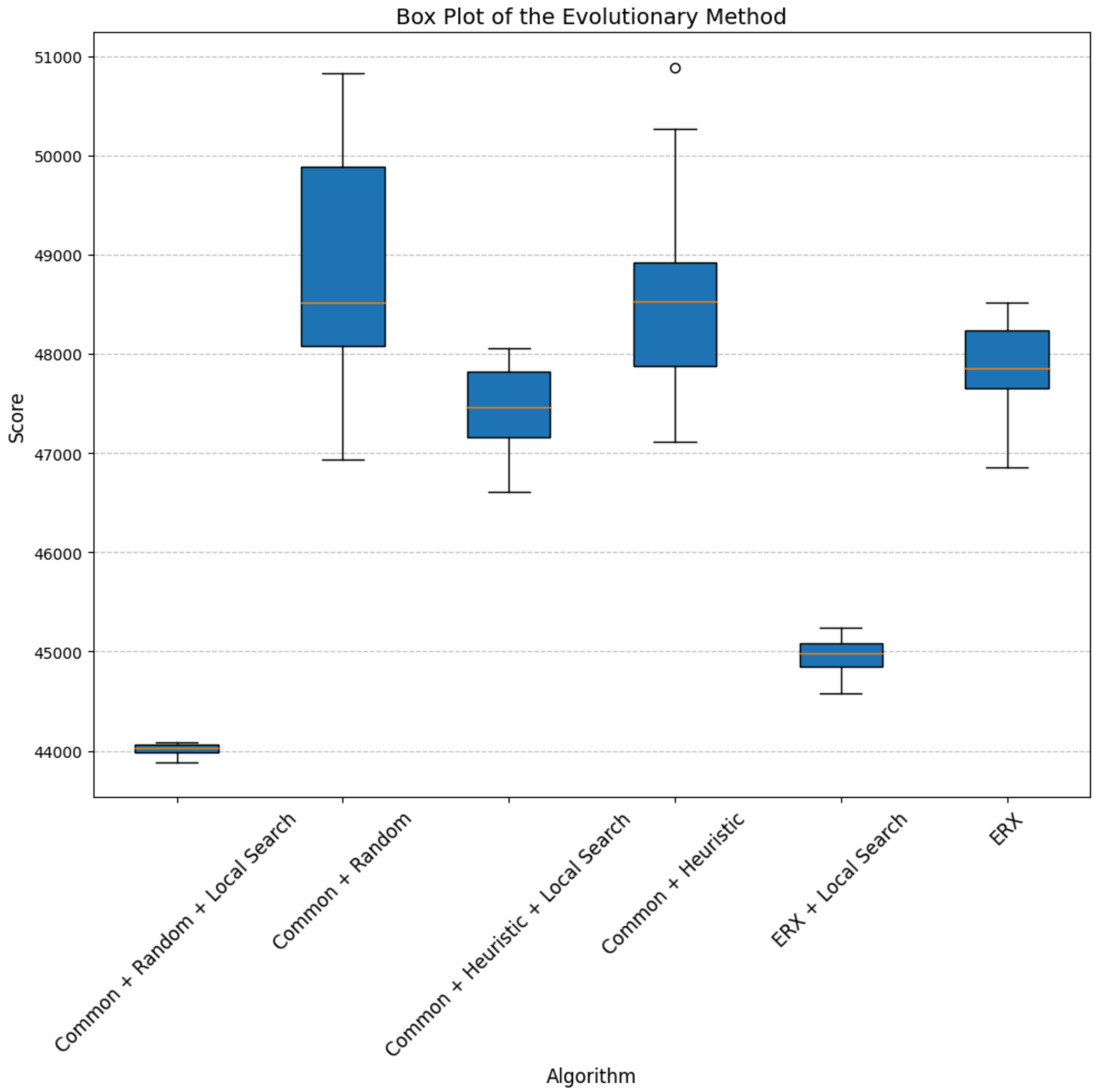

Fig 10. Comparison of performance of different **Hybrid Evolutionary Algorithms** on the TSPB instance.

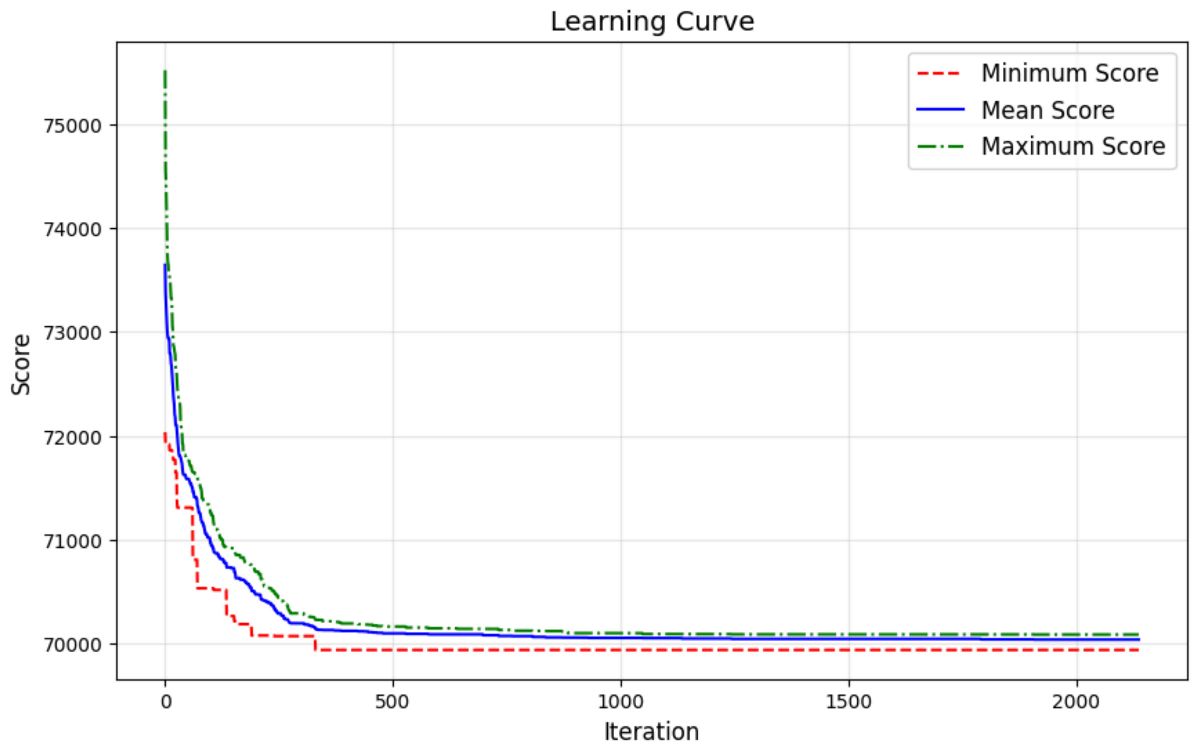

Fig 11. Comparison of learning curves for the best performing **Hybrid Evolutionary Algorithm** on the TSPA instance.

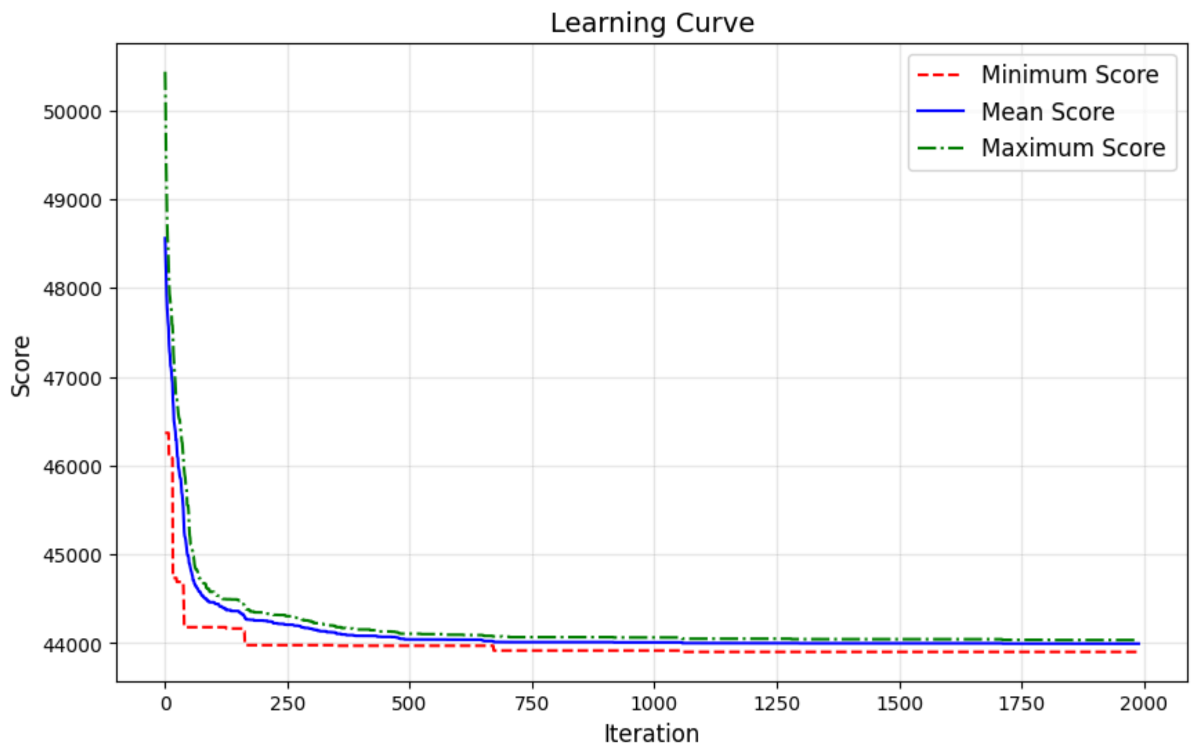

Fig 12. Comparison of learning curves for the best performing **Hybrid Evolutionary Algorithm** on the TSPB instance.

4. Conclusions

The Hybrid Evolutionary Algorithms can achieve scores comparable to ILS and LNS algorithms provided an appropriate operator is selected. Both the random recombination and Edge Recombination crossover notably outperform the MSLS algorithm with the random recombination having the best overall performance possibly due to relatively quick exploration of various local optima. The quality of this approach might be further improved with additional diversity preservation mechanisms as the population converges rather fast to a stable state. Regardless of the operator used, applying local search on generated children is crucial to find better solutions in the population as otherwise, the likelihood of creating a child surpassing local optima found in the initial solutions is extremely low despite the massively increased number of iterations. Poor performance of the heuristic operator might be due to the lack of sufficient exploration of solutions and slow execution speed limiting the possible number of iterations. Further improvements to the evolutionary algorithms can include increasing the population size and the allowed runtime.