Authors: Weronika Zawadzka 151943 Eliza Czaplicka 151963

Date: January 11, 2025

Best solutions have been checked with the solution checker

Summary Report for w9_HEA

Based on methods from [our github project]

Problem description

We are given three columns of integers with a row for each node. The first two columns contain x and y coordinates of the node positions in a plane. The third column contains node costs. The goal is to select exactly 50% of the nodes (if the number of nodes is odd we round the number of nodes to be selected up) and form a Hamiltonian cycle (closed path) through this set of nodes such that the sum of the total length of the path plus the total cost of the selected nodes is minimized.

The distances between nodes are calculated as Euclidean distances rounded mathematically to integer values. The distance matrix should be calculated just after reading an instance, and then only the distance matrix (no nodes coordinates) should be accessed by optimization methods to allow instances defined only by distance matrices.

Pseudocode of implemented algorithms

taken from LNS:

Stopping condition = average running time of MSLS from the previous assignment

Repair operator = Greedy2RegretWeightedSum

Local search = LS_Steepest2edgesMoveEvals

Algorithm:

initialize population = generate 20 solutions with local search

initialize costs = cost for solution in population

while (System.currentTimeMillis() - startingTime < maxTime) {

randomly select 2 parents from population

child = operator(parents)

child = LS(child) (optional)

childF = evaluate(child)

if childF not in costs and max(costs) > childF:

pop.set(at worstFId, child);

costs.set(at worstFId, childF);

}

return best solution

Operators

Operator 1:

commonEdges = edgesP1 U edges P2
commonNodes = nodesP1 U nodesP2
child = NULL * size(P1);

place commonEdges in child in places where they appear in P1
place commonNodes in places where they appear in P1

for empty places in child:
choose random node not yet in solution

Operator 2:

commonEdges = edgesP1 U edges P2
commonNodes = nodesP1 U nodesP2
child = P1;

remove from child edges not in commonEdges
remove from child nodes not in commonNodes
repair with Greedy2RegretWeightedSum

Summary performance of each method

Summary for Execution Time (ms)

Method	A	B
HEA_op1	13384.2 (13383 - 13385)	13384.25 (13383 - 13386)
HEA_op1_LS	13387.9 (13383 - 13401)	13385.4 (13383 - 13392)
HEA_op2	13384.05 (13383 - 13386)	13384.3 (13383 - 13386)
HEA_op2_LS	13385.4 (13383 - 13389)	13386.15 (13383 - 13393)
IteratedLocalSearch	13174.7 (13172 - 13181)	13174.65 (13172 - 13186)
LargeNeighborhoodSearch	13471.6 (13440 - 13739)	13481.05 (13442 - 13690)
LargeNeighborhoodSearchLS	13449.95 (13437 - 13466)	13448.95 (13437 - 13465)
MultipleStartLocalSearch	14797.15 (13345 - 16217)	15006.15 (13576 - 15673)

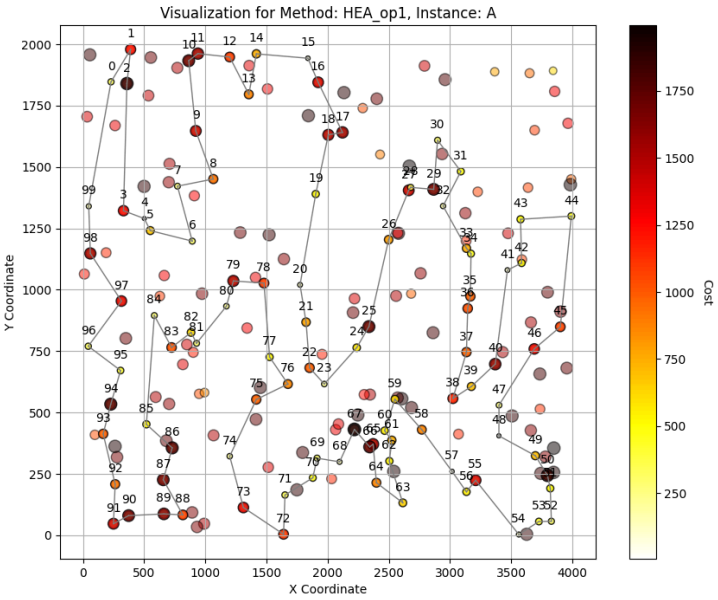
Summary for Objective Function Value

Method	A	B
HEA_op1	69363.9 (69248 - 69922)	43989.35 (43671 - 44198)
HEA_op1_LS	**69265.7 (69100 - 69501)**	**43668.8 (43586 - 44058)**
HEA_op2	69268.8 (69207 - 69532)	43945.15 (43671 - 44121)
HEA_op2_LS	69298.2 (69207 - 69501)	43884.65 (43668 - 44106)
IteratedLocalSearch	69615.25 (69239 - 69951)	43879.4 (**43456** - 44527)
LargeNeighborhoodSearch	69438.5 (69327 - 70034)	44103.75 (43897 - 44223)
LargeNeighborhoodSearchLS	69321.4 (69207 - 69501)	43965.4 (43829 - 44106)
MultipleStartLocalSearch	72616.2 (72053 - 73027)	46813.0 (46123 - 47377)

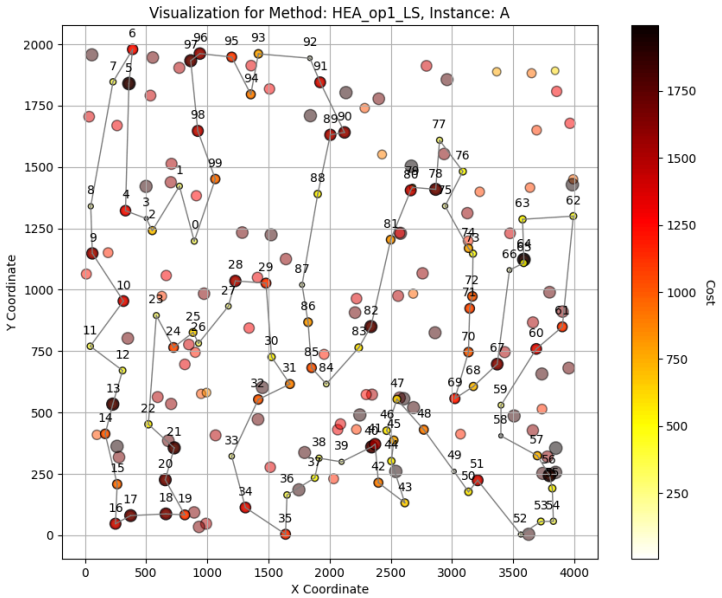
2D visualisations of best solutions

Instance: A

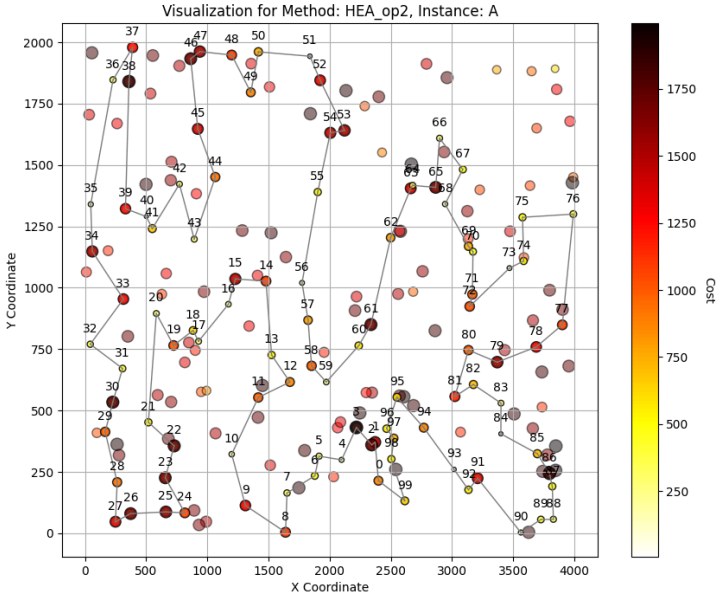
Method: HEA_op1 with score=69248



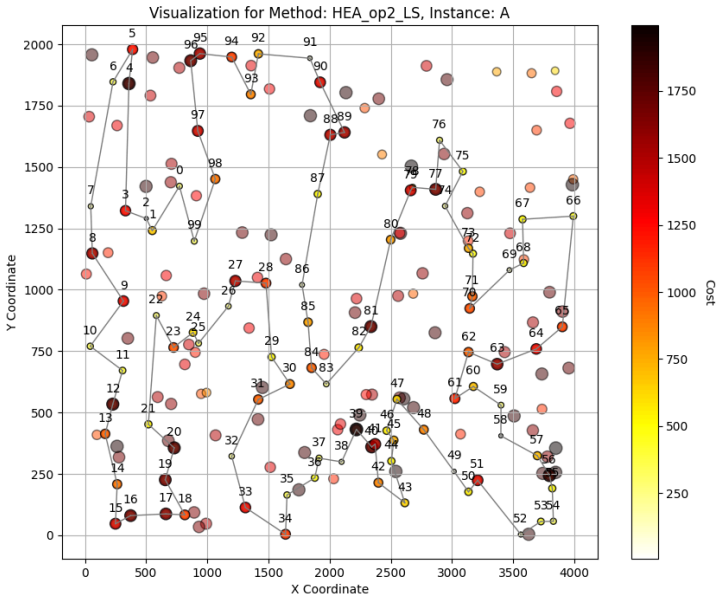
Method: HEA_op1_LS with score=69100



Method: HEA_op2 with score=69207

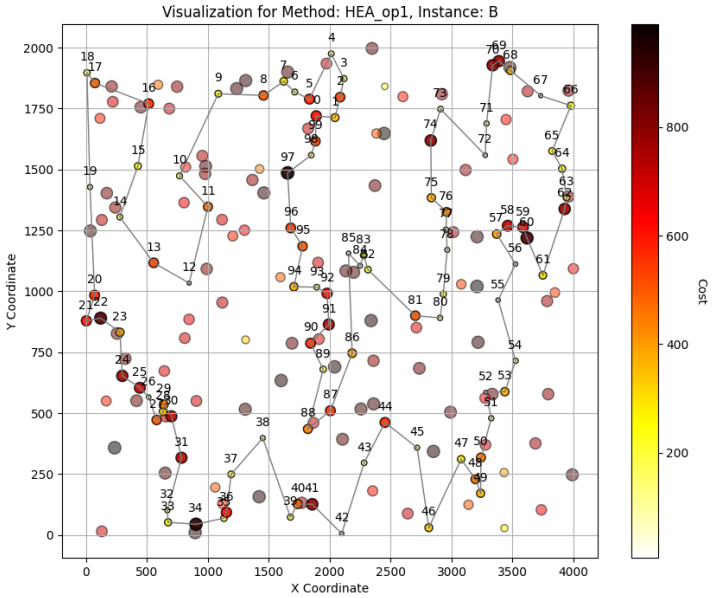


Method: HEA_op2_LS with score=69207

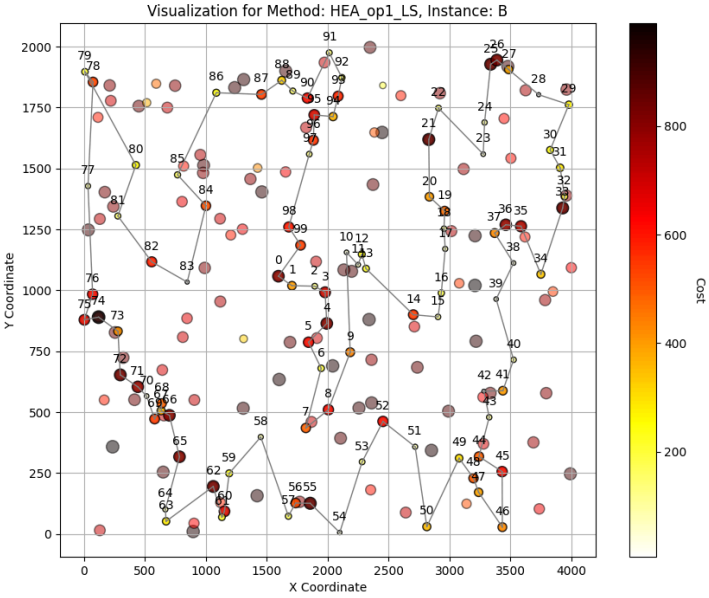


Instance: B

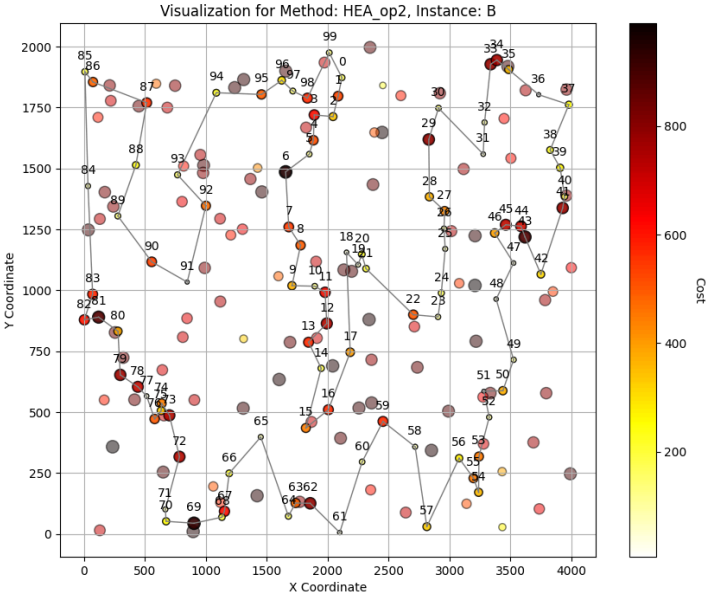
Method: HEA_op1 with score=43671



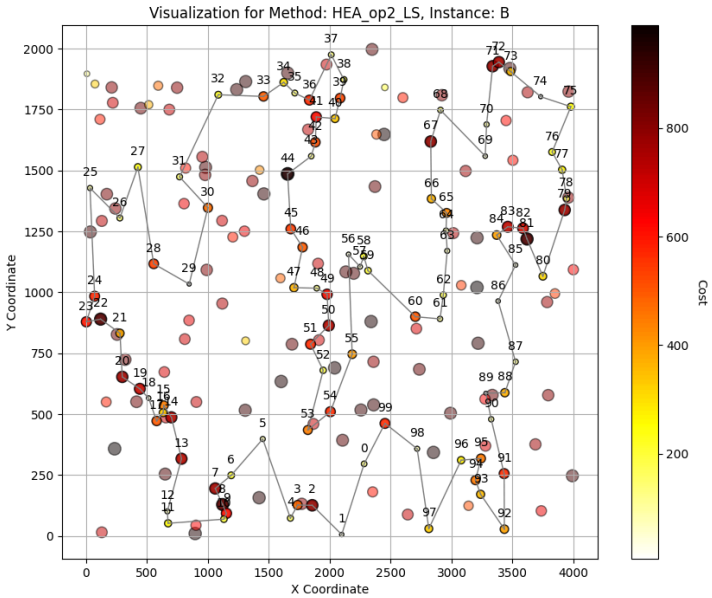
Method: HEA_op1_LS with score=43586



Method: HEA_op2 with score=43671



Method: HEA_op2_LS with score=43668



Best solutions, indices

Instance: A

Method: HEA_op1

Lowest Objective Function Value (f_val): 69248

Solution:

18, 108, 69, 159, 193, 41, 115, 139, 46, 68, 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, 121, 53, 180, 154, 135, 70, 127, 123, 162, 133, 151, 51, 118, 59, 65, 116, 43, 42, 184, 35, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 181, 146, 22

Method: HEA_op1_LS

Lowest Objective Function Value (f_val): 69100

Solution:

115, 139, 41, 193, 159, 69, 108, 18, 22, 146, 181, 34, 160, 48, 54, 177, 10, 190, 4, 112, 84, 35, 184, 42, 43, 116, 65, 59, 118, 51, 151, 133, 162, 123, 127, 70, 135, 154, 180, 53, 100, 26, 86, 75, 101, 1, 97, 152, 2, 120, 44, 25, 16, 171, 175, 113, 56, 31, 78, 145, 196, 81, 90, 165, 119, 40, 185, 179, 92, 129, 57, 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

Method: HEA_op2

Lowest Objective Function Value (f_val): 69207

Solution:

86, 26, 100, 121, 53, 180, 154, 135, 70, 127, 123, 162, 133, 151, 51, 118, 59, 65, 116, 43, 42, 184, 35, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 181, 146, 22, 18, 108, 69, 159, 193, 41, 139, 115, 46, 68, 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, 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

Method: HEA_op2_LS

Lowest Objective Function Value (f_val): 69207

Solution:

139, 41, 193, 159, 69, 108, 18, 22, 146, 181, 34, 160, 48, 54, 177, 10, 190, 4, 112, 84, 35, 184, 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

Instance: B

Method: HEA_op1

Lowest Objective Function Value (f_val): 43671

Solution:

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, 162, 175, 78, 5, 177, 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, 111, 82, 21, 8, 104, 144, 160, 33, 138, 11, 139, 43, 168, 195

Method: HEA_op1_LS

Lowest Objective Function Value (f_val): 43586

Solution:

182, 138, 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, 179, 99, 130, 95, 185, 86, 166, 194, 176, 113, 103, 114, 137, 127, 89, 163, 187, 153, 81, 77, 141, 91, 61, 36, 177, 5, 78, 175, 45, 80, 190, 136, 73, 31, 54, 193, 117, 198, 156, 1, 16, 27, 38, 63, 107, 40, 122, 135, 131, 121, 51, 90, 147, 6, 188, 169, 132, 70, 3, 15, 145, 13, 195, 168, 139, 11

Method: HEA_op2

Lowest Objective Function Value (f_val): 43671

Solution:

3, 15, 145, 13, 195, 168, 43, 139, 11, 138, 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, 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, 78, 175, 162, 80, 190, 136, 73, 54, 31, 193, 117, 198, 156, 1, 16, 27, 38, 63, 40, 107, 133, 122, 135, 131, 121, 51, 90, 147, 6, 188, 169, 132, 70

Method: HEA_op2_LS

Lowest Objective Function Value (f_val): 43668

Solution:

77, 141, 91, 61, 36, 177, 5, 45, 142, 78, 175, 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, 70, 3, 15, 145, 13, 195, 168, 43, 139, 11, 138, 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, 179, 22, 99, 130, 95, 185, 86, 166, 194, 176, 113, 114, 137, 127, 89, 103, 163, 187, 153, 81

Conclusions

This week's results turned out to be the best from all previous. Specifically, the hybrid evolutionary algorithm with operator1 was the best. HEA combines speed and ability to locally improve solutions by local search and globality of evolutionary algorithms. We used two ways of hybridization: for HEA_op1 and HEA_op2 - hybridization "First LS then EA" so that EA started with a very good population. Algorithms HEA_op1_LS and HEA_op2_LS also used "LS during EA" which performed LS after each recombination. We also experimented with mutation (destroy+repair from LNS) after recombination and it made the algorithms not converge so quickly and made our final results better. Overall, HEA_op1 with LS after recombination was the best and HEA_op1 without LS performed a little worse, but similarly.