Report

Team members:

- Sofya Aksenyuk, 150284
- Uladzimir Ivashka, 150281

Problem Description

Given a set of nodes, each characterized by their (x, y) coordinates in a plane and an associated cost, the challenge is to select exactly 50% of these nodes and form a Hamiltonian cycle.

The goal is to minimize the sum of the total length of the path plus the total cost of the selected nodes.

Distances between nodes are computed as Euclidean distances and rounded to the nearest integer.

Methodology

Simulated Annealing

Simulated Annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. It is inspired by the process of annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The algorithm starts at a high temperature and gradually cools down. At each step, it randomly selects a solution, accepting not only improvements but also, with a certain probability, worse solutions. This allows it to potentially escape local optima in the search for a global optimum. As the temperature decreases, the algorithm becomes less likely to accept worse solutions, focusing more on exploring the local neighborhood of the current solution.

Ant Colony Optimization

Ant Colony Optimization (ACO) is a probabilistic technique used to solve computational problems which can be reduced to finding good paths through graphs. Inspired by the behavior of ants searching for food, in ACO, a number of artificial ants build solutions to the optimization problem and exchange information on their quality. Ants explore paths and deposit a virtual pheromone trail, with stronger trails indicating shorter or better paths. Future artificial ants are then more likely to follow these stronger trails. This collective learning process, based on the intensity of the pheromone trails, allows the ants to find shorter paths over time. ACO is particularly effective for problems like the traveling salesman problem or routing in networks.

Source code

Link: Source Code SA

Link: Source Code ACO

Pseudocode

Simulated Annealing in our application with HEA

```
FUNCTION HEA(DistanceMatrix, Costs, EndTime, PopSize, Oper):

    StartTime = time()
    Counter = 0

    Population = (generate 20 solutions using SteepestLocalSearch)
    TotalCosts = (compute total costs for each of the solutions from
population)

    WHILE (time() - StartTime < EndTime):
        Action = (randomly choose crossover or mutation with different
probabilities)

        IF (Action is crossover):
            Parent1 = (tournament selection with size 3)
            Parent2 = (tournament selection with size 3, shouldn't be same
as Parent1)
            Child = (Operator_1 from previous lab)
        ELSE IF (Action is mutation):
            Parent = (tournament selection with size 3)
            Child = (apply perturbation from ILS to Parent)

        IF (choose between SteepestLocalSearch and Simulated Annealing with
some probabilities):
            Child = (apply Simulated Annealing with some probability)
        ELSE:
            Child = SteepestLocalSearch(Child, DistanceMatrix, Costs)

        ChildTotalCost = GetTotalCost(Child, DistanceMatrix, Costs)
        IF (ChildTotalCost not it TotalCosts):
            MaxTotalCost, Idx = (get worst total cost and its index)
            IF (ChildTotalCost < MaxTotalCost):
                Population[Idx] = Child
                TotalCosts[Idx] = ChildTotalCost

        Counter += 1

    BestSolution, BestTotalCost = (get best solution from population and
its total cost)

    RETURN BestSolution, BestTotalCost, Counter


FUNCTION SimulatedAnnealing(InitialSolution, DistanceMatrix, Costs,
T=10000, CoolingRate=0.995, StoppingTemperature=1, UseLocal)
    CurrentSolution = InitialSolution
    CurrentCost = GetTotalCost(CurrentSolution, DistanceMatrix, Costs)
```

```
    BestSolution = CurrentSolution
    BestCost = CurrentCost

    WHILE (T > StoppingTemperature):
        Neighbor = (small perturbation of CurrentSolution)
        IF (UseLocal is True):
            Neighbor = SteepestLocalSearch(Neighbor, DistanceMatrix, Costs)
        NeighborCost = GetTotalCost(Neighbor, DistanceMatrix, Costs)

        IF NeighborCost < CurrentCost OR Random(0, 1) < Exp((CurrentCost -
NeighborCost) / T)
            CurrentSolution = Neighbor
            CurrentCost = NeighborCost

            IF NeighborCost < BestCost
                BestSolution = Neighbor
                BestCost = NeighborCost

        T = T * CoolingRate

    RETURN BestSolution
```

Ant Colony Optimization

```
CLASS Ant
    FUNCTION __Init__(StartNode)
        Tour = [StartNode]
        TotalCost = 0

    FUNCTION VisitNode(Node, Cost)
        Tour.Append(Node)
        TotalCost += Cost

CLASS AntColonyOptimization
    FUNCTION __Init__(DistanceMatrix, Costs, EndTime, NAnts=20, NBest=5,
Decay=0.5, Alpha=1, Beta=2)
        DistanceMatrix, Costs, NAnts, NBest, EndTime, Decay, Alpha, Beta =
DistanceMatrix, Costs, NAnts, NBest, EndTime, Decay, Alpha, Beta
        PheromoneMatrix = InitializeMatrix(DistanceMatrix.Shape, 1 /
Length(DistanceMatrix))

    FUNCTION _ApplyPheromoneDecay()
        PheromoneMatrix *= (1 - Decay)

    FUNCTION _UpdatePheromones(Ants)
        FOR Each Ant in Ants (Update PheromoneMatrix based on Ant's Tour
and DistanceMatrix)

    FUNCTION _SelectNextNode(CurrentNode, TabooList)
        Heuristic = 1 / (DistanceMatrix[CurrentNode] + Costs), set 0 for
nodes in TabooList
```

```
        Attractiveness = CalculateAttractiveness(PheromoneMatrix,
Heuristic, Alpha, Beta)
        RETURN ChooseNodeBasedOnProbability(Attractiveness)

    FUNCTION _ConstructSolution()
        Ants = [Create NAnts number of Ants]
        FOR Each Step in Tour Length (Move each Ant to Next Node using
_SelectNextNode and update their Tour)

        RETURN Ants

    FUNCTION Run()
        StartTime = GetCurrentTime()
        BestCost = Infinity, BestSolution = None

        WHILE (CurrentTime - StartTime < EndTime)
            Ants = _ConstructSolution()
            _ApplyPheromoneDecay()
            UpdatePheromonesForBestAnts(Ants, NBest)

            IF (Cost of Best Ant < BestCost)
                Update BestCost and BestSolution

        RETURN BestCost, BestSolution
```

Computational Experiments in our application

Results

Table of Cost

	Algorithm	TSPA	TSPB	TSPC	TSPD
0	RandomSearch	264715.690 (234787.0 – 289096.0)	265095.315 (238995.0 – 287788.0)	214163.21 (192811.0 – 232188.0)	217922.530 (195986.0 – 246928.0)
1	NearestNeighbor	86319.43 (83561.0 - 93749.0)	77688.285 (75928.0 – 79791.0)	56709.06 (53466.0 – 60369.0)	52472.030 (48240.0 – 57939.0)
2	GreedyCycle	78049.310 (75990.0 – 81934.0)	71717.795 (68973.0 – 78237.0)	56404.07 (53723.0 – 58868.0)	55334.72 (50717.0 – 61896.0)
3	Greedy2Regret	117178.570 (110218.0 -124855.0)	121592.715 (111115.0 – 131138.0)	69547.98 (66114.0 – 73603.0)	71900.575 (66024.0 – 76887.0)
4	Greedy2RegretWeighted	75953.435 (74701.0 – 78510.0)	71428.365 (69147.0 – 77017.0)	54217.24 (51747.0 – 58087.0)	52285.360 (47629.0 – 57787.0)
5	Greedy-edges-Random	77965.71 (75112.0 - 82142.0)	71018.74 (68315.0 - 74769.0)	51652.62 (49206.0 - 54174.0)	48705.455 (45913.0 - 51873.0)
6	Greedy-edges-GreedyHeuristic	75438.075 (74521.0 - 77298.0)	70388.815 (67808.0 - 76131.0)	53714.495 (51034.0 - 56971.0)	51525.635 (47281.0 - 57524.0)
7	Greedy-nodes-Random	90554.405 (83565.0 - 100644.0)	85537.285 (77218.0 - 93542.0)	63718.49 (56594.0 - 71397.0)	61963.345 (54964.0 - 69495.0)
8	Greedy-nodes-GreedyHeuristic	75512.725 (74521.0 - 77471.0)	70688.61 (68572.0 - 76529.0)	53967.21 (51195.0 - 56739.0)	51145.425 (47368.0 - 57605.0)
9	Steepest-edges-GreedyHeuristic	75391.6 (74541.0 - 77063.0)	70215.965 (68122.0 - 75907.0)	53529.02 (50972.0 - 57024.0)	51093.845 (46915.0 - 57412.0)
10	Steepest-nodes-Random	93224.675 (84578.0 - 101323.0)	88096.545 (80827.0 - 96390.0)	65743.78 (58924.0 - 77555.0)	64443.6 (55944.0 - 78053.0)
11	Steepest-nodes-GreedyHeuristic	75606.625 (74630.0 - 76953.0)	70758.4 (68572.0 - 76545.0)	53807.135 (51203.0 - 57026.0)	51203.305 (46990.0 - 55945.0)
12	Steepest-CandidateMovesEdges-Random	80856.86 (76709.0 - 85935.0)	73804.335 (69986.0 - 80091.0)	51680.02 (49141.0 - 54538.0)	48294.905 (45672.0 - 52381.0)
13	Steepest-PreviousDeltas-Random	79913.365 (76798.0 - 87582.0)	73141.7 (68884.0 - 78988.0)	53064.8 (49346.0 - 57601.0)	49794.255 (46581.0 - 55387.0)
14	Steepest-edges-Random	78307.45 (75390.0 - 83932.0)	71637.29 (68186.0 - 77703.0)	51692.4 (49183.0 - 54617.0)	48647.455 (46150.0 - 51579.0)
15	MSLS	75178.125 (74562.0 - 75588.0)	68299.225 (67595.0 - 68770.0)	49113.225 (48452.0 - 49647.0)	45531.725 (44622.0 - 46017.0)
16	ILS	73078.1 (72855.0 - 73279.0)	66442.5 (66117.0 - 66770.0)	47136.4 (46811.0 - 47604.0)	43407.25 (43207.0 - 43949.0)
17	LSNS_LS	73666.65 (73082.0 - 74700.0)	66879.5 (66505.0 - 67580.0)	47558.15 (46928.0 - 48154.0)	43989.0 (43314.0 - 45305.0)
18	LSNS_noLS	73671.35 (73153.0 - 74123.0)	67113.7 (66772.0 - 67483.0)	47623.85 (47004.0 - 48236.0)	44279.2 (43587.0 - 45644.0)
19	HEA_oper1_LS	73169.65 (72885.0 - 73477.0)	66357.2 (66154.0 - 66684.0)	47001.15 (46777.0 - 47269.0)	43359.35 (43181.0 - 43781.0)
20	HEA_oper1_noLS	75712.5 (74984.0 - 76480.0)	68862.15 (68237.0 - 69963.0)	49805.3 (49015.0 - 50670.0)	46310.8 (45587.0 - 47482.0)
21	HEA_oper2_LS	73585.9 (73250.0 - 74119.0)	66956.4 (66415.0 - 67311.0)	47651.2 (47084.0 - 48065.0)	43842.45 (43240.0 - 44554.0)
22	HEA_oper2_noLS	73730.0 (73265.0 - 74184.0)	67069.4 (66722.0 - 67694.0)	47763.4 (47349.0 - 48096.0)	44256.3 (43736.0 - 45018.0)
23	Own_method_SA_HEA	73148.75 (72886.0 - 73485.0)	66549.7 (66220.0 - 66784.0)	47242.1 (46908.0 - 47731.0)	43530.25 (43252.0 - 44161.0)
24	Own_method_Ant	88805.0 (86815.0 - 89763.0)	71972.0 (69901.0 - 73311.0)	57405.142857142855 (54874.0 - 61224.0)	53207.0 (51066.0 - 54918.0)

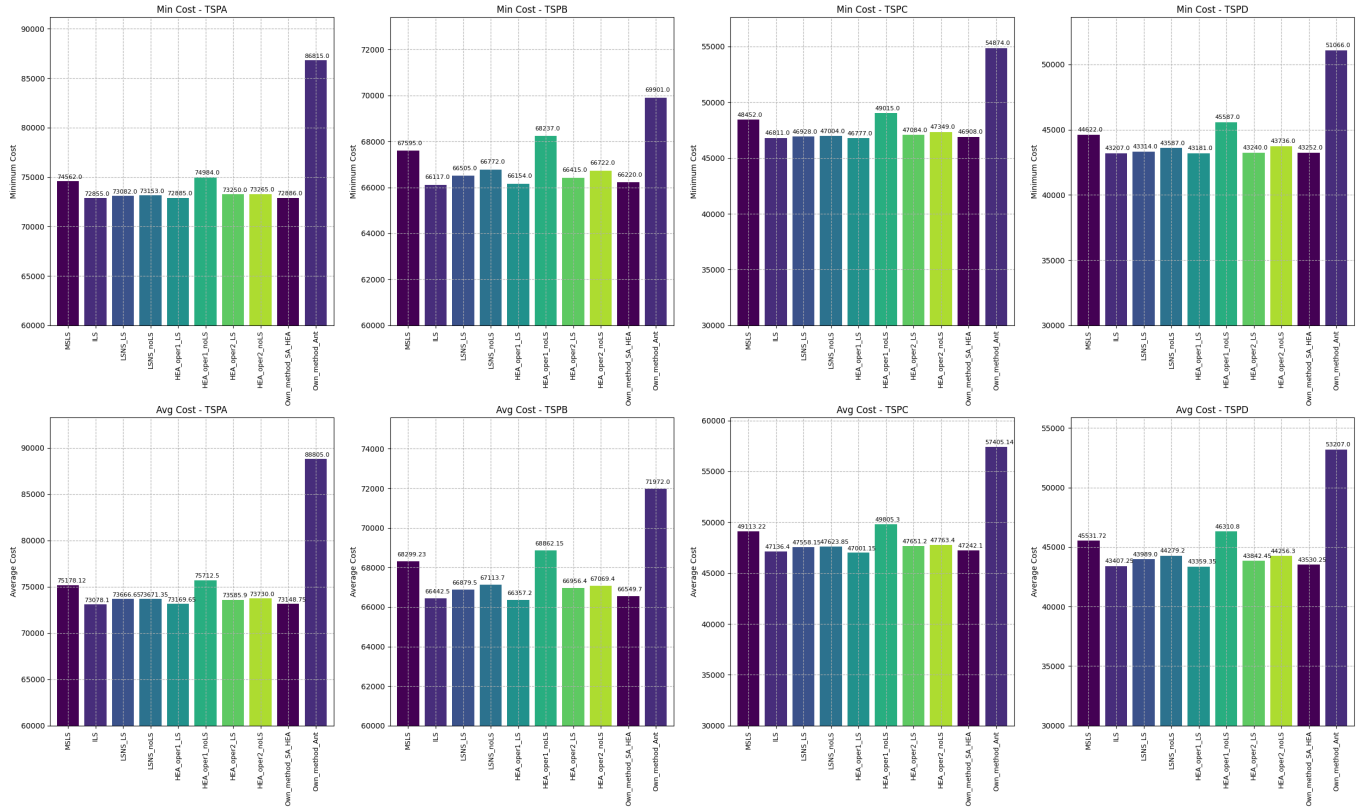
Table of Time

	Algorithm	TSPA	TSPB	TSPC	TSPD
0	Greedy-edges-Random	7.682 (6.255 - 9.005)	7.989 (6.794 - 9.754)	7.739 (6.556 - 9.202)	6.699 (4.754 - 12.083)
1	Greedy-edges-GreedyHeuristic	0.269 (0.073 - 0.655)	0.418 (0.202 - 0.92)	0.315 (0.069 - 0.831)	0.366 (0.102 - 0.995)
2	Greedy-nodes-Random	4.005 (3.243 - 5.092)	4.126 (3.348 - 5.451)	3.946 (3.129 - 5.034)	4.469 (3.086 - 9.898)
3	Greedy-nodes-GreedyHeuristic	0.243 (0.082 - 0.802)	0.291 (0.145 - 0.581)	0.242 (0.073 - 0.663)	0.284 (0.083 - 0.765)
4	Steepest-edges-GreedyHeuristic	0.538 (0.073 - 1.405)	0.883 (0.337 - 1.591)	0.648 (0.065 - 1.511)	0.73 (0.133 - 2.109)
5	Steepest-nodes-Random	14.777 (11.815 - 18.468)	14.972 (12.168 - 18.308)	14.376 (11.243 - 18.885)	14.716 (11.481 - 19.12)
6	Steepest-nodes-GreedyHeuristic	0.335 (0.074 - 1.124)	0.669 (0.345 - 1.219)	0.578 (0.077 - 1.63)	0.628 (0.145 - 1.347)
7	Steepest-CandidateMovesEdges-Random	1.448 (1.205 - 2.451)	1.476 (1.217 - 1.761)	1.437 (1.203 - 1.788)	1.364 (1.078 - 2.015)
8	Steepest-PreviousDeltas-Random	6.517 (5.292 - 7.677)	6.523 (5.425 - 8.05)	6.448 (5.265 - 7.948)	6.485 (5.131 - 7.811)
9	Steepest-edges-Random	12.176 (10.589 - 14.762)	12.516 (11.009 - 14.349)	11.872 (10.033 - 13.814)	12.583 (10.98 - 14.19)
10	MSLS	1263.926 (1233.821 - 1307.715)	1310.493 (1272.85 - 1428.453)	1266.808 (1237.736 - 1355.937)	1268.967 (1237.212 - 1352.452)
11	ILS	1264.199 (1264.006 - 1264.605)	1310.183 (1310.003 - 1310.458)	1267.235 (1267.003 - 1267.694)	1269.149 (1269.014 - 1269.642)
12	LSNS_LS	1264.387 (1264.004 - 1265.019)	1310.405 (1310.04 - 1311.135)	1267.534 (1267.005 - 1268.495)	1269.503 (1269.084 - 1269.988)
13	LSNS_noLS	1264.393 (1264.003 - 1265.225)	1310.348 (1310.034 - 1310.92)	1267.366 (1267.01 - 1267.841)	1269.263 (1269.032 - 1269.794)
14	HEA_oper1_LS	1264.138 (1264.006 - 1264.420)	1310.080 (1310.002 - 1310.302)	1267.112 (1267.023 - 1267.417)	1269.086 (1269.021 - 1269.267)
15	HEA_oper1_noLS	1264.0 (1264.0 - 1264.0)	1310.0(1310.0 - 1310.0)	1267.0 (1267.0- 1267.0)	1269.0 (1269.0 - 1269.0)
16	HEA_oper2_LS	1264.091 (1264.010 - 1264.237)	1310.105(1310.0 - 1310.314)	1267.102 (1267.0 - 1267.440)	1269.137 (1269.024 - 1269.431)
17	HEA_oper2_noLS	1264.073 (1264.003 - 1264.203)	1310.054(1310.005- 1310.170)	1267.059 (1267.000 - 1267.161)	1269.074 (1269.0 - 1269.244)
18	Own_method_SA_HEA	1501.694 (1500.012 - 1505.667)	1501.532 (1500.001 - 1505.905)	1502.307 (1500.505 - 1505.28)	1501.766 (1500.018 - 1504.912)
19	Own_method_Ant	1265.07 (1264.56 - 1267.02)	1310.265 (1310.046 - 1310.382)	1267.105 (1267.045 - 1267.183)	1269.138 (1269.005 - 1269.212)

Table of Iterations

	Algorithm	TSPA	TSPB	TSPC	TSPD
0	ILS	4197.75 (3991 - 4326)	4160.4 (4054 - 4370)	4206.3(3835 - 4365)	4184.95(4009 - 4349)
1	LSNS_LS	1827.85 (1328.0 - 2325.0)	1942.75 (1353.0 - 2558.0)	1674.45 (1123.0 - 2196.0)	1823.9 (1249.0 - 2392.0)
2	LSNS_noLS	2268.0 (1633.0 - 2723.0)	2400.75 (1724.0 - 2931.0)	2326.15 (1654.0 - 2848.0)	2313.25 (1636.0 - 2854.0)
3	HEA_oper1_LS	5346.1 (3412 - 7191)	7223.35 (4615 - 8928)	3857.9 (3077 - 5475)	4626.95 (3330 - 5847)
4	HEA_oper1_noLS	3624639.45 (3595274 - 3663433)	3790013.1 (3763971 - 3817952)	3603259.1 (3532825 - 3634250)	3653376.0 (3606872 - 3698766)
5	HEA_oper2_LS	7670.55 (5536 - 13233)	9007.85 (5469 - 16230)	6512.85 (3794 - 10266)	6944.95 (4534 - 11800)
6	HEA_oper2_noLS	14150.3 (7998 - 31735)	15206.4 (8896 - 33472)	13260.35 (7623 - 26872)	12414.55 (6152 - 34599)
7	Own_method_SA_HEA	1344.5 (1174 - 1603)	1267.85 (1131 - 1560)	1245.0 (1008 - 1471)	1238.7 (1043 - 1480)

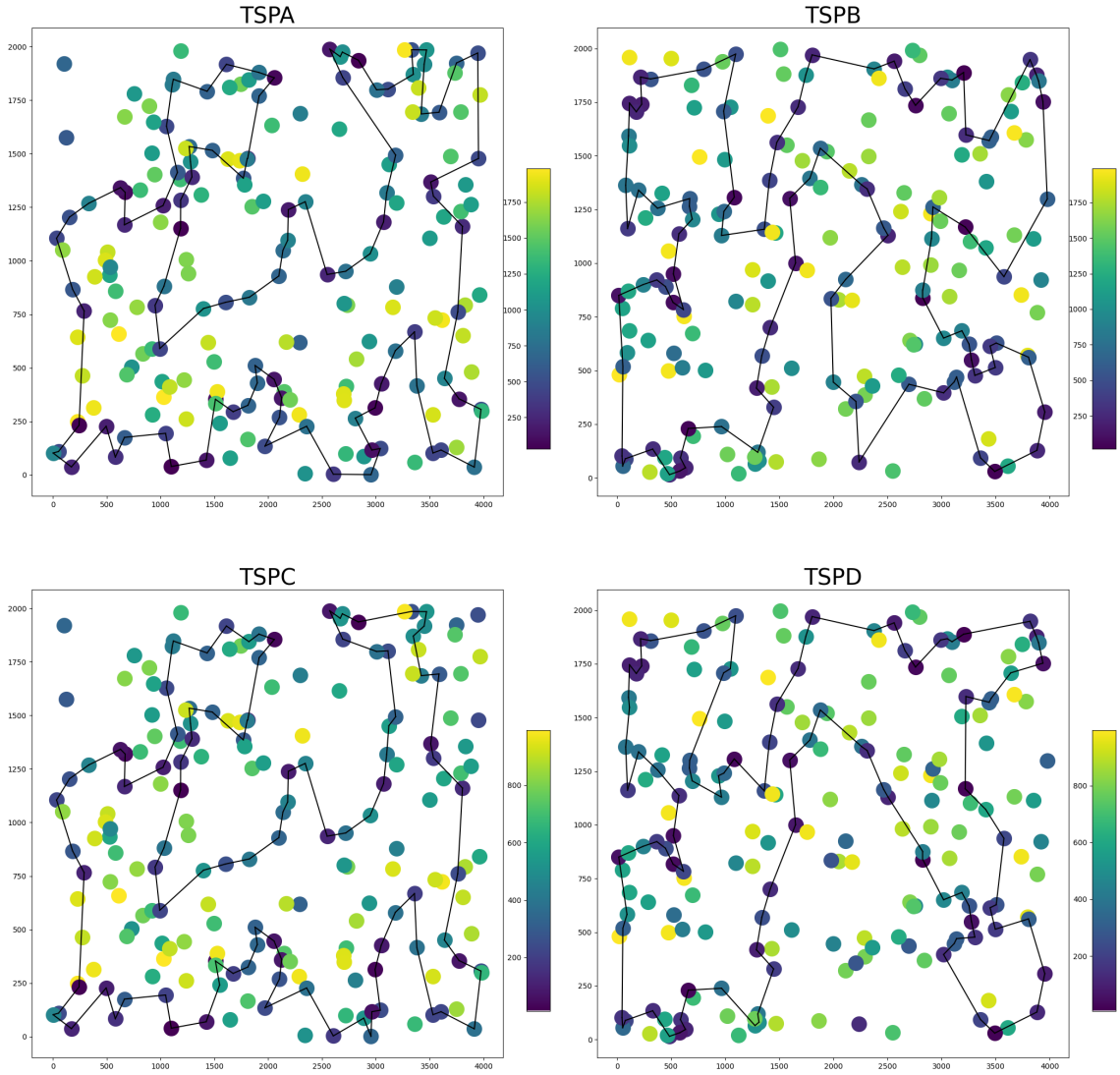
Plots of Cost



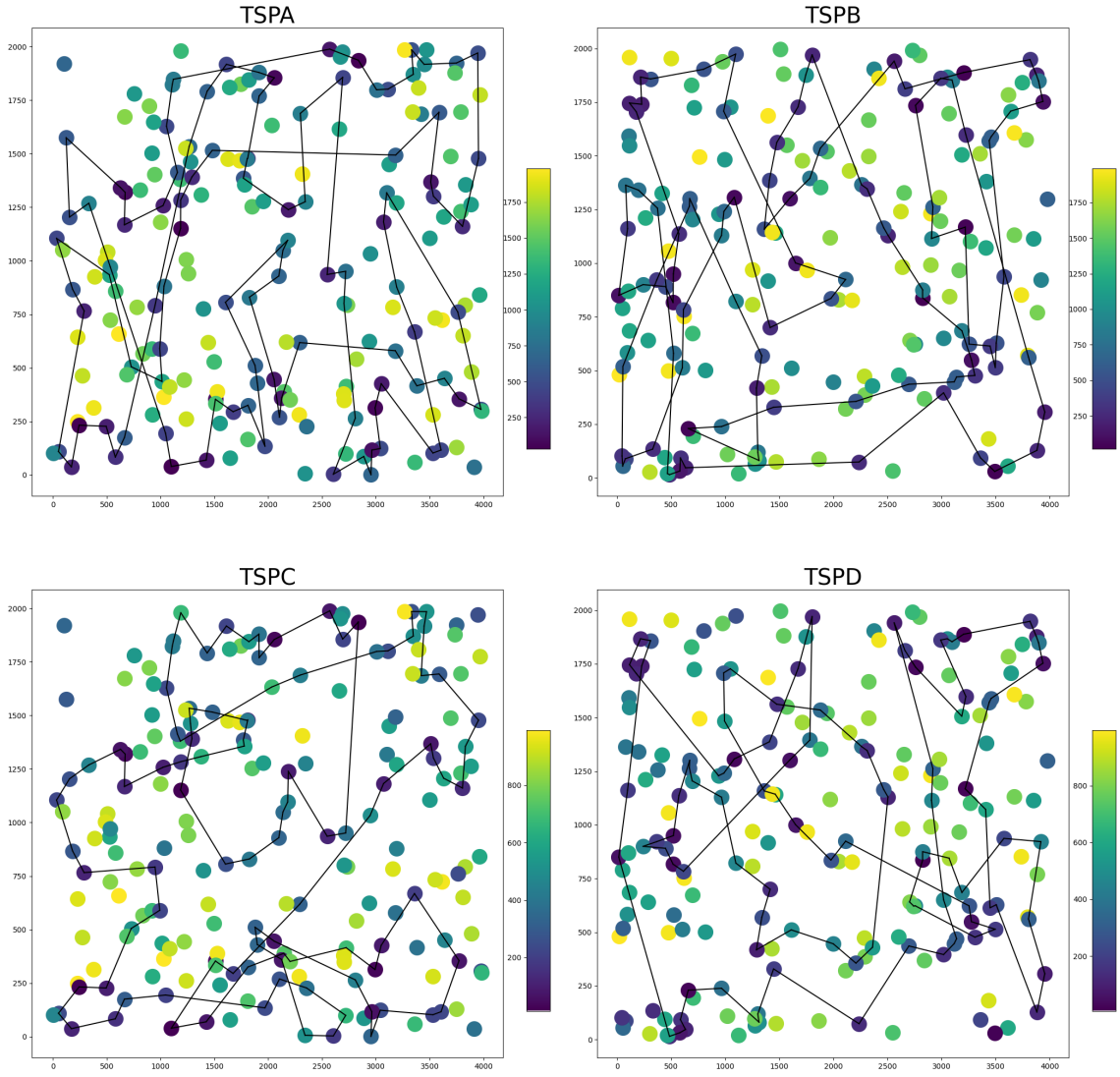
Best Solutions Plots

See plots: Plots

Own_method_SA_HEA



Own_method_Ant



Best solution among all methods so far

TSPA

[48, 106, 160, 11, 152, 130, 119, 109, 189, 75, 1, 177, 41, 137, 199, 192, 175, 114, 4, 77, 43, 121, 91, 50, 149, 0, 19, 178, 164, 159, 143, 59, 147, 116, 27, 96, 185, 64, 20, 71, 61, 163, 74, 113, 195, 53, 62, 32, 180, 81, 154, 144, 141, 87, 79, 194, 21, 171, 108, 15, 117, 22, 55, 36, 132, 128, 145, 76, 161, 153, 88, 127, 186, 45, 167, 101, 99, 135, 51, 112, 66, 6, 172, 156, 98, 190, 72, 12, 94, 89, 73, 31, 111, 14, 80, 95, 169, 8, 26, 92]

Cost: 72855.0

TSPB

[166, 59, 119, 193, 71, 44, 196, 117, 150, 162, 158, 67, 156, 91, 70, 51, 174, 140, 148, 141, 130, 142, 53, 69, 115, 82, 63, 8, 16, 18, 29, 33, 19, 190, 198, 135, 95, 172, 163, 182, 2, 5, 34, 183, 197, 31, 101, 38, 103, 131, 24, 127, 121, 179, 143, 122, 92, 26, 66, 169, 0, 57, 99, 50, 112, 154, 134, 25, 36, 165, 37, 137, 88, 55, 153, 80, 157, 145, 79, 136, 73, 185, 132, 52, 139, 107, 12, 189, 170, 181, 147, 159, 64, 129, 89, 58, 171, 72, 114, 85]

Cost: 66117.0

TSPC

[20, 71, 61, 163, 74, 113, 195, 53, 62, 32, 180, 81, 154, 102, 144, 141, 87, 79, 194, 21, 171, 108, 15, 117, 22, 55, 36, 132, 128, 145, 76, 161, 153, 88, 127, 186, 45, 167, 101, 99, 135, 51, 5, 112, 72, 190, 66, 6, 172, 156, 98, 94, 42, 89, 12, 73, 31, 95, 169, 8, 26, 92, 48, 106, 160, 11, 152, 130, 119, 109, 189, 75, 1, 177, 41, 137, 199, 192, 43, 77, 4, 114, 91, 121, 50, 149, 0, 69, 19, 178, 164, 159, 143, 59, 147, 116, 27, 96, 185, 64]

Cost: 46777.0

TSPD

[47, 170, 181, 147, 159, 64, 129, 89, 58, 171, 72, 114, 85, 166, 71, 44, 196, 117, 150, 162, 158, 67, 3, 156, 91, 70, 51, 174, 140, 148, 141, 130, 142, 53, 32, 113, 69, 115, 82, 63, 8, 16, 18, 29, 33, 19, 190, 198, 135, 169, 66, 26, 92, 122, 143, 179, 197, 183, 34, 31, 101, 38, 103, 131, 121, 127, 24, 50, 43, 99, 137, 37, 165, 123, 154, 134, 25, 36, 88, 55, 4, 153, 80, 157, 145, 79, 136, 61, 73, 185, 132, 52, 12, 107, 97, 139, 193, 119, 59, 189]

Cost: 43181.0

Conclusions

- Own_method_SA_HEA performs relatively same as HEA_Oper_1 and ILS, despite much less number of iterations.
- Ant colony doesn't performs as good as expected, beign the worst algo among last ones.