Evolutionary Computation

Assignment 6

Antoni Lasik 148287
Daniel Jankowski 148257
https://github.com/JankowskiDaniel/evolutionary-computation/tree/AL/assignment6

Problem description

The task involves analyzing three columns of integers, each row corresponding to a single node. The initial two columns designate the x and y coordinates, pinpointing the nodes' locations on a plane, while the third column specifies the cost associated with each node. The objective is to meticulously choose an exact half of the total nodes (in cases where the total node count is an odd number, the count of nodes to be selected is adjusted upward to the nearest whole number) to construct a Hamiltonian cycle, which is essentially a continuous loop that passes through each member of the selected set of nodes. The criterion for this selection is that the aggregate of the complete path's length and the cumulative cost of the chosen nodes should be as low as possible.

To quantify the distances between nodes, we employ the Euclidean distance formula, and the resulting figures are rounded off to the nearest integer in a standard mathematical fashion. Moreover, as part of the distance between nodes, we take into account the cost of the destination node. This ensures that cost has a significant impact on the final results.

In this assignment we are supposed to compare Multi Start Local Search and Iterated Local Search, to make this comparison meaningful the mean runtime of MSLS for a given instance will be used as a stoping condition for ILS. For ILS we used perturbation that 4 times swaps 4 edges.

Pseudocode of implemented algorithms

```
calculate_distance_matrix(coords, costs):
       dist_matrix = [][]
       FOR i IN RANGE\(len(coords)):
              FOR j IN RANGE(len(coords)):
                     dist_matrix[i][j]  =  round(sqrt((coords[i].x  -  coords[j].x)**2  +
                     (coords[i].y - coords[j].y)**2)
       RETURN dist_matrix


objective_function(solution, dist_matrix, costs):
       total_score = 0
       n = len(solution)
       FOR x in range(n):
                     total_score += dist_matrix[solution[x - 1]][solution[x]]
                     total_score += costs[solution[x]]
       RETURN total_score

generate_random_solution(n):
    RETURN random.sample(range(0, n * 2), n)


perturb(current_solution):
    l = len(current_solution)
    FOR x in range(4):
        i = random.randint(2,l-2)
        j = random.randint(0,i-1)
        k = random.randint(j+2,l)
        m = random.randint(0,k-1)

        new_solution = (current_solution[:j ]
                    + current_solution[j :i ][::-1]
                    + current_solution[i :])
        current_solution = new_solution
        new_solution = (current_solution[:m ]
                    + current_solution[m :k ][::-1]
                    + current_solution[k :])
```

```
        current_solution = new_solution

    RETRUN current_solution


MSLS(distance_matrix, costs, trials):
       best_score=float('inf')
       FOR i in tqdm(range(trials)):
              random_solution = generate_random_solution(100)
              s = SteepestLocalSearch(initial_solution=random_solution,
                            distance_matrix=distance_matrix,
                            costs=costs,)
              solution, score= s.run(random_solution, ["inter","edges"],
              show_progress=False)

              IF best_score<score:
                     best_score=score
                     best_solution=solution
       RETRUN best_score, best_solution

ILS(distance_matrix, costs, trials, mean):
       solution= generate_random_solution(100)
       s = SteepestLocalSearch(solution, distance_matrix, costs)
       solution, score = s.run(solution, ["inter","edges"], show_progress=False)
       best_score=score
       start=time.time()
       WHILE time.time()-start<mean:
              perturbed = perturb(solution,score, distance_matrix )
              s = SteepestLocalSearch(perturbed, distance_matrix, costs)
              solution, score = s.run(perturbed, ["inter","edges"], show_progress=False)
              IF score<best_score:
                     best_score=score
                     best_solution=solution
       RETRUN best_score, best_solution
```

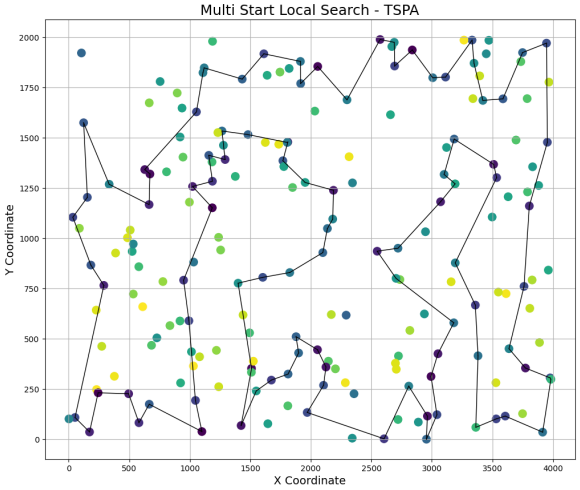
Results

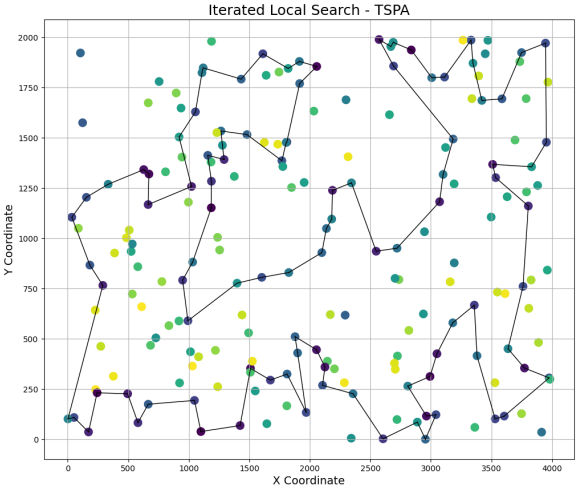
Method	Instance A	Instance B	Instance C	Instance D
Greedy LS, random solution, two-edges + inter route	77,014(74,663-79,803)	69,990(67,877-74,141)	50,998 (49,340-53,141)	48,068 (45,336-51,629)
Greedy LS, random solution, two-nodes + inter route	90,940(84,816-99,390)	85,570(77,908-97,299)	63,929 (58,135-70,886)	62,175 (54,310-71,108)
Greedy LS, best solution from 2-regret with weighted sum, two-edges + inter route	75,792 (74,221-79,688)	71,266 (67,384-77,120)	52350,15(48,931-55,758)	51,013 (45,212-59,478)
Greedy LS, best solution from 2-regret with weighted sum, two-nodes + inter route	75,932(74,344-79,315)	71,839 (67,384-77,565)	52,638 (49,649-56,472)	51,248(45,097-60,185)
Steepest LS, best solution from 2-regret with weighted sum, two-edges + inter route	75,728(74,091-79,220)	71,233 (67,384-77,057)	52,299 (49,098-5,5665)	50,977(45,097-59,478)
Steepest LS, best solution from 2-regret with weighted sum, two-nodes + inter route	75,880(74,280-79,220)	71,894(67,384-77,420)	52,607 (49,460-56,472)	51,247 (45,097-60,185)
Candidates LS, random solution, two-edges + inter route	81,129(76,609-86,447)	73,977(69,300-80,189)	51,588(49,120-54,801)	48,429(45,385-51,392)
Steepest LS, random solution, two-nodes + inter route	92,714(84,218-103,034)	87,666(79,356-97,895)	65,679(59,604-73,386)	64,162(54,716-75,351)
Steepest LS, random solution, two-edges + inter route	78,017 (74,874-82,619)	71,337(67,909-76,199)	51,485 (49,235-53,755)	48,225 (45,673-51,639)
Deltas from previous iteration, random solution, two-edges + inter route	78,192(75,149-82,556)	71,709(68,307-76,210)	51,940(49,347-55,591)	48,509(45,966-52,016)
Multi Start Local Search random solution, two-edges + inter route	75,353(74,509-76,155)	68,613(67,998-69,217)	49,168(48,156-49,550)	45,913(45,364-46,315)
Iterated Local Search random solution, two-edges + inter route	**73,780(73,388-73,996)**	**66,772(66,455-67,109)**	**48,132(47,505-48,590)**	**44,131(43,690-44,784)**

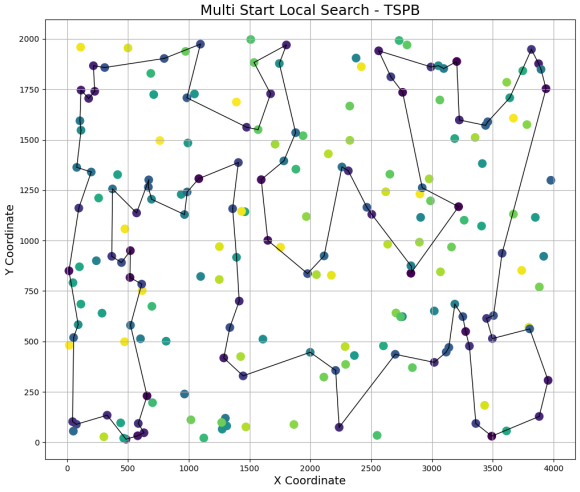
Runtimes

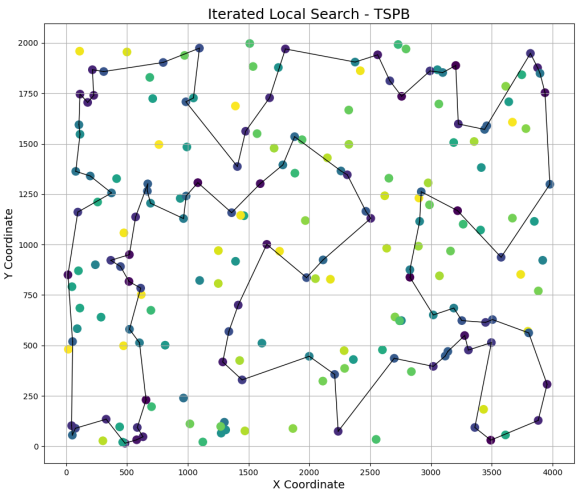
Method	Instance A	Instance B	Instance C	Instance D
Greedy LS, random solution, two-edges + inter route	1.56(1.06-2.63)	1.95(1.19-3.48)	1.25(0.77-2.28)	1.18(0.72-1.99)
Greedy LS, random solution, two-nodes + inter route	1.68(1.03-2.98)	1.95(0.81-6.66)	1.38(0.79-2.21)	1.37(0.77-2.36)
Greedy LS, best solution from 2-regret with weighted sum, two-edges + inter route	0.67(0.51-0.97)	0.7(0.5-1.18)	0.66(0.5-0.93)	0.65(0.51-0.89)
Greedy LS, best solution from 2-regret with weighted sum, two-nodes + inter route	0.63(0.46-0.89)	0.69(0.53-1.15)	0.68(0.49-1.24)	0.67(0.54-1.18)
Steepest LS, best solution from 2-regret with weighted sum, two-edges + inter route	0.85(0.55-1.53)	0.95(0.53-1.78)	0.94(0.57-1.6)	1(0.58-1.38)
Steepest LS, best solution from 2-regret with weighted sum, two-nodes + inter route	0.88(0.58-1.71)	0.83(0.53-1.57)	0.89(0.5-1.58)	1.03(0.67-1.5)
Candidate LS, random solution, two-edges + inter route	4.43(3.95-6.41)	4.52(3.99-5.70)	4.53(3.83-6.44)	4.58(4.06-5.88)
Steepest LS, random solution, two-nodes + inter route	6.82(5.46-8.96)	6.63(4.89-10.51)	6.8(5.41-9.2)	0.69(0.5-1.18)
Steepest LS, random solution, two-edges + inter route	5.46(4.47-7.46)	5.64(4.51-7.16)	5.41(4.72-6.54)	5.64(4.76-6.88)
Deltas from previous iteration, random solution, two-edges + inter route	1.34(1.06-2.13)	1.80(0.80-2.31)	1.82 (1.05-2.51)	1.88(1.23-2.44)
Multi Start Local Search random solution, two-edges + inter route**	96.51(95.50-97.42)	98.41(97.72-99.80)	95.92(95.30-96.60)	96.64(96.13-97.36)
Iterated Local Search random solution, two-edges + inter route**	96.56(96.51-96.62)	98.45(98.41-98.48)	95.95(95.92-95.98)	96.67(96.64-96.73)

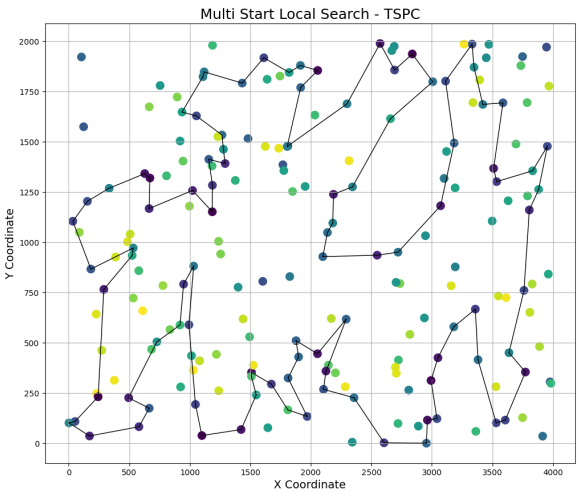
**Run on a slightly better setup

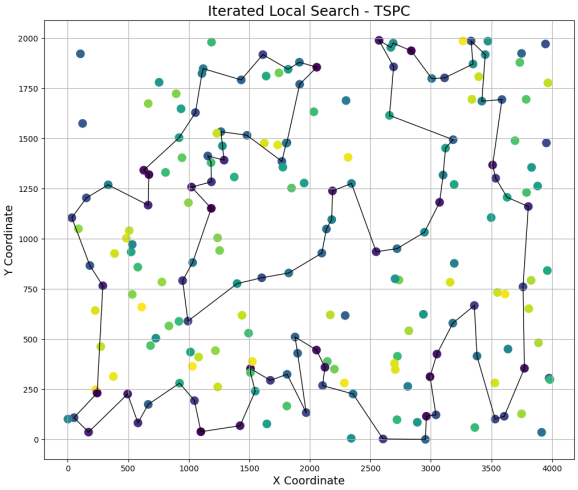

Multi Start Local Search - TSPA
X Coordinate
Y Coordinate

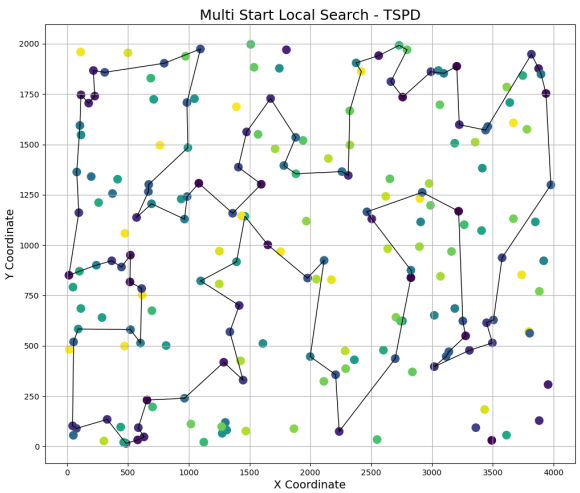

Iterated Local Search - TSPA
X Coordinate
Y Coordinate
Node Cost

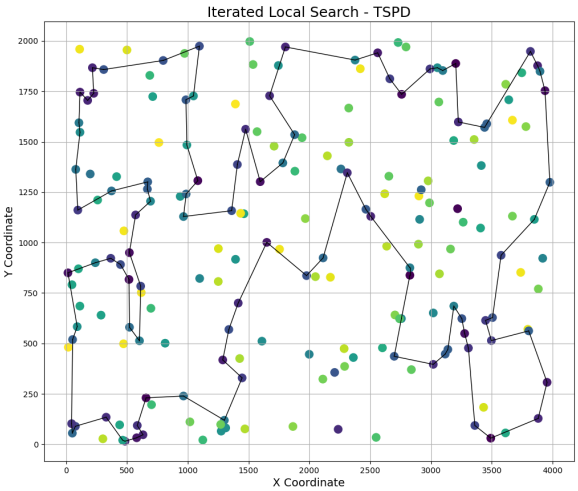

Multi Start Local Search - TSPB
X Coordinate
Y Coordinate

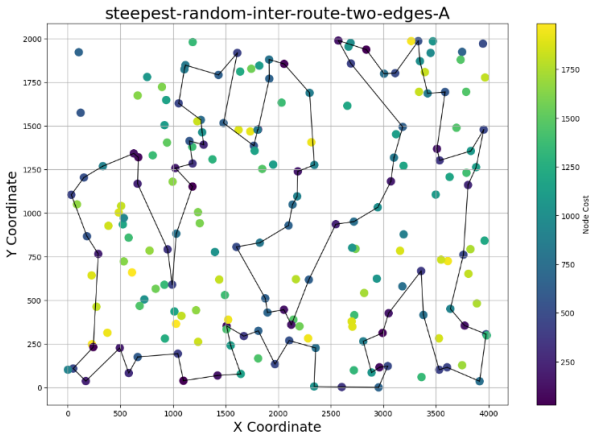

Iterated Local Search - TSPB
X Coordinate
Y Coordinate
Node Cost

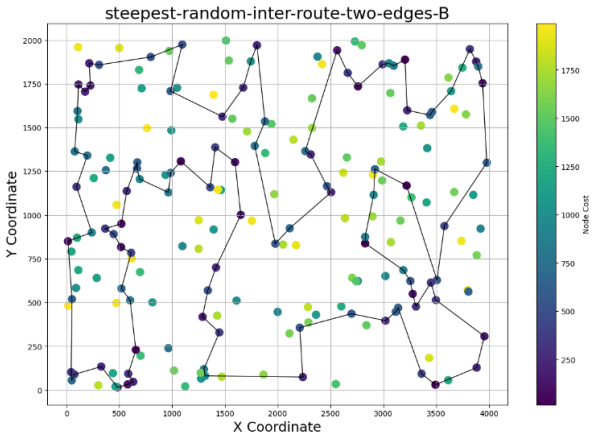

Multi Start Local Search - TSPC
X Coordinate
Y Coordinate

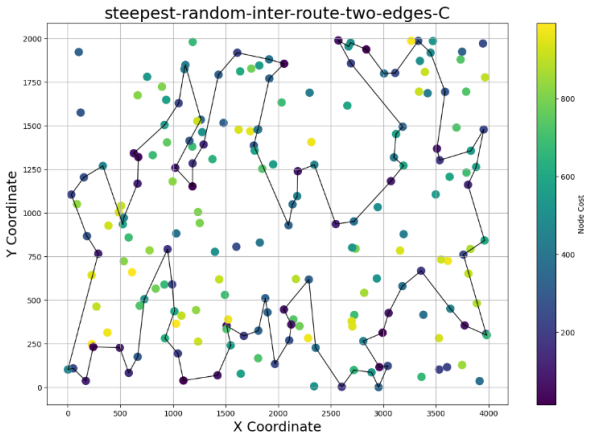

Iterated Local Search - TSPC
X Coordinate
Y Coordinate
Node Cost

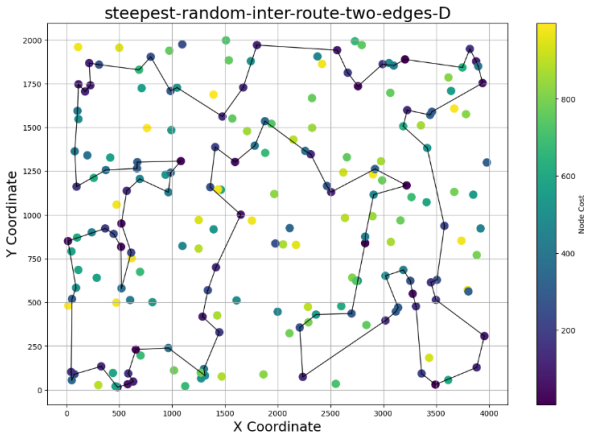

Multi Start Local Search - TSPD
X Coordinate
Y Coordinate

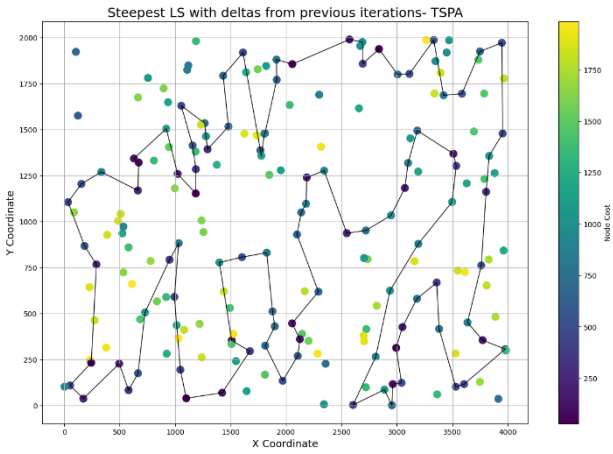

Iterated Local Search - TSPD
X Coordinate
Y Coordinate
Node Cost

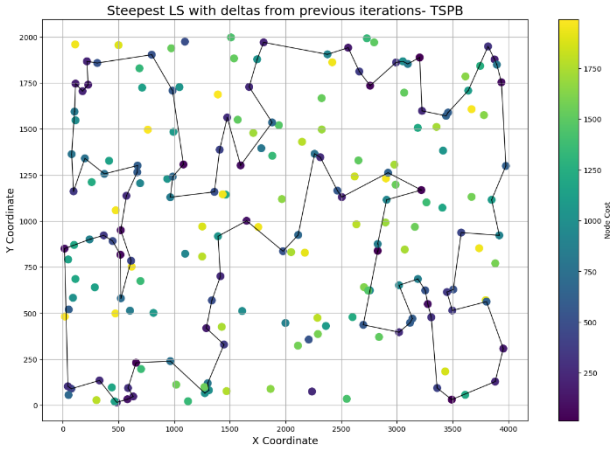

steepest-random-inter-route-two-edges-A
Y Coordinate
X Coordinate
Node Cost

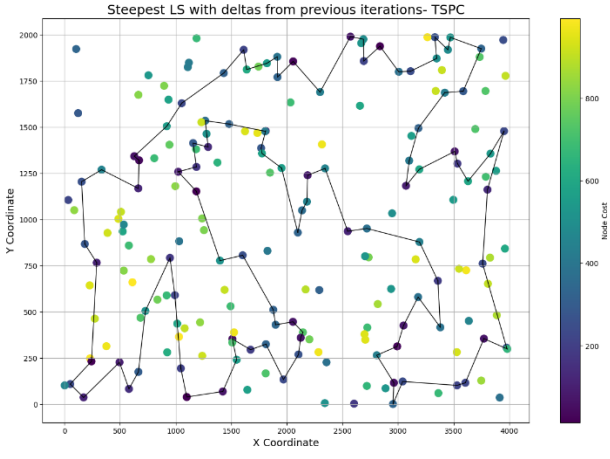

steepest-random-inter-route-two-edges-B
Y Coordinate
X Coordinate
Node Cost

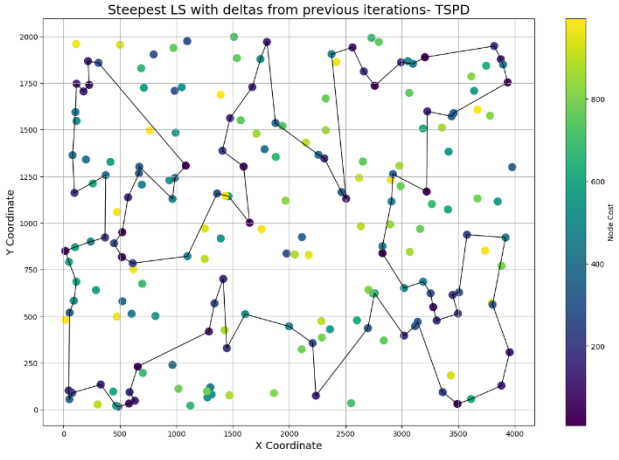

steepest-random-inter-route-two-edges-C
Y Coordinate
X Coordinate
Node Cost


steepest-random-inter-route-two-edges-D
Y Coordinate
X Coordinate
Node Cost


Steepest LS with deltas from previous iterations- TSPA
Y Coordinate
X Coordinate
Node Cost


Steepest LS with deltas from previous iterations- TSPB
Y Coordinate
X Coordinate
Node Cost


Steepest LS with deltas from previous iterations- TSPC
Y Coordinate
X Coordinate
Node Cost


Steepest LS with deltas from previous iterations- TSPD
Y Coordinate
X Coordinate
Node Cost

MSLS:

A:

[95, 169, 8, 124, 80, 14, 111, 94, 72, 190, 98, 66, 172, 156, 6, 24, 45, 186, 127, 88, 153, 161, 76, 21, 194, 79, 87, 141, 144, 154, 81, 180, 32, 62, 108, 171, 15, 117, 53, 22, 55, 195, 113, 74, 163, 71, 20, 64, 185, 116, 27, 147, 96, 59, 143, 159, 37, 132, 36, 128, 164, 178, 19, 0, 149, 50, 91, 121, 43, 77, 4, 114, 175, 167, 101, 60, 126, 174, 199, 137, 41, 177, 1, 75, 189, 109, 119, 130, 152, 11, 48, 106, 26, 134, 99, 135, 51, 112, 73, 31]

B:

[148, 140, 174, 51, 70, 91, 156, 67, 114, 158, 162, 150, 117, 192, 196, 44, 71, 119, 59, 166, 85, 58, 89, 129, 64, 159, 147, 181, 170, 189, 47, 185, 73, 61, 136, 33, 29, 18, 132, 12, 107, 139, 52, 65, 16, 172, 95, 135, 198, 190, 19, 145, 157, 80, 153, 55, 88, 137, 37, 165, 36, 25, 134, 154, 112, 99, 57, 0, 169, 66, 26, 92, 122, 143, 121, 127, 24, 50, 131, 103, 38, 101, 31, 179, 197, 183, 34, 128, 5, 2, 182, 163, 40, 8, 63, 115, 82, 142, 130, 141]

C:

[31, 73, 112, 51, 135, 196, 95, 169, 8, 26, 106, 48, 92, 130, 119, 109, 189, 75, 1, 177, 41, 137, 199, 192, 77, 43, 50, 149, 0, 19, 178, 164, 159, 143, 59, 147, 116, 27, 96, 185, 64, 20, 71, 61, 113, 163, 74, 195, 53, 22, 55, 36, 132, 128, 145, 76, 161, 91, 121, 114, 4, 2, 175, 167, 45, 186, 127, 88, 153, 157, 21, 79, 194, 171, 117, 15, 108, 62, 93, 32, 180, 81, 154, 102, 144, 87, 141, 24, 6, 156, 172, 66, 98, 190, 72, 12, 94, 89, 111, 14]

D:

[185, 18, 29, 33, 73, 61, 136, 79, 145, 157, 80, 153, 4, 55, 88, 36, 25, 134, 154, 165, 37, 137, 146, 57, 0, 99, 43, 50, 24, 127, 121, 103, 38, 101, 31, 92, 122, 143, 179, 197, 183, 34, 5, 167, 128, 66, 169, 98, 135, 198, 190, 19, 95, 172, 16, 52, 14, 8, 63, 115, 82, 53, 142, 130, 141, 148, 140, 188, 161, 174, 51, 70, 91, 156, 67, 114, 72, 58, 89, 129, 64, 159, 147, 187, 181, 100, 170, 189, 59, 119, 193, 71, 162, 150, 117, 196, 44, 107, 12, 132]

ILS:

A:

[95, 169, 8, 26, 92, 48, 106, 11, 152, 130, 119, 109, 189, 75, 174, 177, 1, 41, 137, 199, 192, 77, 4, 114, 91, 121, 43, 50, 149, 0, 19, 178, 164, 159, 143, 59, 96, 147, 27, 116, 185, 64, 20, 71, 61, 163, 74, 113, 195, 155, 62, 32, 180, 81, 154, 102, 144, 141, 87, 79, 194, 21, 171, 108, 15, 117, 53, 22, 55, 36, 132, 128, 145, 76, 161, 153, 88, 127, 186, 45, 167, 101, 135, 51, 112, 66, 6, 172, 156, 98, 190, 72, 94, 12, 73, 31, 111, 14, 80, 124]

B:

[64, 129, 89, 58, 171, 72, 85, 166, 114, 67, 156, 91, 70, 51, 174, 140, 148, 141, 130, 142, 21, 192, 196, 117, 150, 162, 44, 71, 59, 119, 193, 139, 107, 12, 52, 18, 29, 33, 19, 190, 198, 135, 95, 172, 16, 8, 63, 82, 115, 163, 182, 2, 5, 34, 183, 197, 143, 179, 121, 31, 101, 38, 103, 131, 24, 127, 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, 189, 47, 170, 181, 147, 159]

C:

[110, 8, 26, 48, 11, 152, 130, 119, 109, 189, 75, 1, 177, 41, 199, 192, 77, 4, 114, 91, 121, 43, 50, 149, 0, 115, 69, 19, 178, 164, 34, 159, 143, 59, 147, 27, 96, 185, 64, 20, 71, 61, 113, 74, 163, 155, 62, 32, 180, 81, 154, 102, 144, 141, 87, 79, 194, 21, 171, 108, 15, 117, 53, 18, 22, 195, 55, 36, 132, 128, 145, 76, 161, 153, 88, 127, 186, 45, 167, 101, 99, 135, 51, 5, 112, 68, 66, 6, 172, 156, 98, 190, 72, 12, 94, 89, 73, 31, 95, 169]

D:

[101, 38, 103, 131, 121, 127, 24, 50, 94, 112, 154, 134, 25, 36, 165, 37, 137, 88, 55, 4, 153, 80, 157, 145, 79, 136, 73, 33, 29, 18, 185, 132, 52, 139, 107, 12, 109, 189, 47, 170, 181, 147, 159, 64, 129, 89, 58, 171, 72, 114, 166, 59, 119, 193, 71, 44, 196, 21, 192, 117, 150, 162, 158, 67, 3, 156, 91, 70, 51, 174, 188, 140, 148, 141, 130, 142, 53, 69, 115, 82, 63, 8, 16, 172, 95, 190, 198, 135, 169, 66, 128, 5, 34, 183, 197, 92, 122, 143, 179, 31]

Conclusions

Results obtained by the MSLS compared to ILS are worse because of one simple fact, number of iterations of Steepest Local Search that MSLS has to perform is far grater than the ILS that starts already close to some local minima, in this way ILS is not so time consuming and is able to perform more runs to try to get to the better local minima. Also we can conclude that random starting solutions don't produce the same outcome as starting points in ILS(perturbed local minimum), because ILS was able to find better solutions.

Simple perturbation like performing multiple times the two_edge_exchange move is sufficient to produce good results. We were thinking about bigger perturbations but our intuition led us to to the smaller ones, because we thought that when the perturbation will be large it will be more time consuming to perform one loop of ILS and we thought that the best solutions are not that far from the worse ones, sometimes its very close in optimisation landscape but not accessible because of some local optima that makes the solutions converge to it.