Authors: Weronika Zawadzka 151943 Eliza Czaplicka 151963

Date: November 03, 2024

Best solutions have been checked with the solution checker

# Summary Report for w4_candidate_moves

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

- **Steepest 2 edges Candidate Moves**

```
Generate a starting solution randomly shuffling the nodes and selecting the first half.
for node in all nodes:
        for node 2 in all nodes if node != node2:
                cost = distance + cost of node 2
        sort costs and add lowest 10 to cost matrix
while True:
        bestSolutionFound
        bestDelta
        for node1 in current solution:
                for node2 in 10 closest to node1 from cost matrix:
                        if node2 in current solution:
                                if next to each other then skip

                                increase = distanceMatrix[node1][node2] -
                                distanceMatrix[node1][before i] +
                                distanceMatrix[before i][before j] -
                                distanceMatrix[before j][node2]
```

```
                    if increase<bestDelta:
                        reverse order between smaller+1 & larger id
                        save as bestSolutionFound
                    repeat for reversing between smaller and larger -1
                else:
                    prev = previous node to i
                    prev2 = node previous to prev
                    oldF = distanceMatrix[node1][prev] +
                    distanceMatrix[prev][prev2] + cost(prev)
                    newF = distanceMatrix[node1][node2] +
                    distanceMatrix[node2][prev2] + cost(node2)
                    increase = newF-oldF

                    if increase<bestDelta:
                        swap prev with node2
                        save as bestSolutionFound
                    repeat for switching of node2 with node after node1


        if besDelta<0:
            current solution = bestSolutionFound
        else:
            flag = False
return current solution
```

## Summary performance of each method

---

### *Summary for Execution Time (ms)*

| Method | A | B |
|---|---|---|
| Greedy2edgesGreedy | 24.405 (13 - 85) | 31.63 (15 - 88) |
| Greedy2edgesRandom | 64.61 (41 - 110) | 68.43 (46 - 120) |
| Greedy2nodesGreedy | 26.305 (14 - 89) | 28.47 (18 - 100) |
| Greedy2nodesRandom | 72.44 (42 - 154) | 75.935 (45 - 147) |
| Steepest2edgesGreedy | 34.535 (31 - 96) | 72.675 (67 - 156) |
| Steepest2edgesRandom | 999.27 (871 - 1240) | 983.275 (868 - 1148) |
| Steepest2nodesGreedy | 57.395 (52 - 124) | 78.475 (72 - 144) |
| Steepest2nodesRandom | 1038.67 (822 - 1496) | 1046.83 (765 - 1594) |
| Steepest2edgesCandidate | 15.01 (12 - 49) | 15.67 (13 - 48) |

### *Summary for Objective Function Value*

| Method | A | B |
|---|---|---|
| Greedy2edgesGreedy | 71798.75 (71612 - 71910) | 51770.925 (50579 - 51955) |
| Greedy2edgesRandom | 75071.865 (72146 - 78848) | 49265.74 (46899 - 51687) |
| Greedy2nodesGreedy | 71643.38 (71624 - 71675) | 51822.39 (51728 - 51931) |
| Greedy2nodesRandom | 85784.375 (79581 - 94992) | 60971.66 (53818 - 70256) |
| Steepest2edgesGreedy | 71865.0 (71865 - 71865) | 51790.0 (51790 - 51790) |
| Steepest2edgesRandom | 75305.065 (73122 - 79322) | 49511.875 (46569 - 53465) |
| Steepest2nodesGreedy | 71624.0 (71624 - 71624) | 51865.0 (51865 - 51865) |
| Steepest2nodesRandom | 88075.565 (81006 - 99501) | 63187.48 (54443 - 71877) |
| Steepest2edgesCandidate | 77731.47 (74540 - 84266) | 48474.77 (46036 - 52029) |

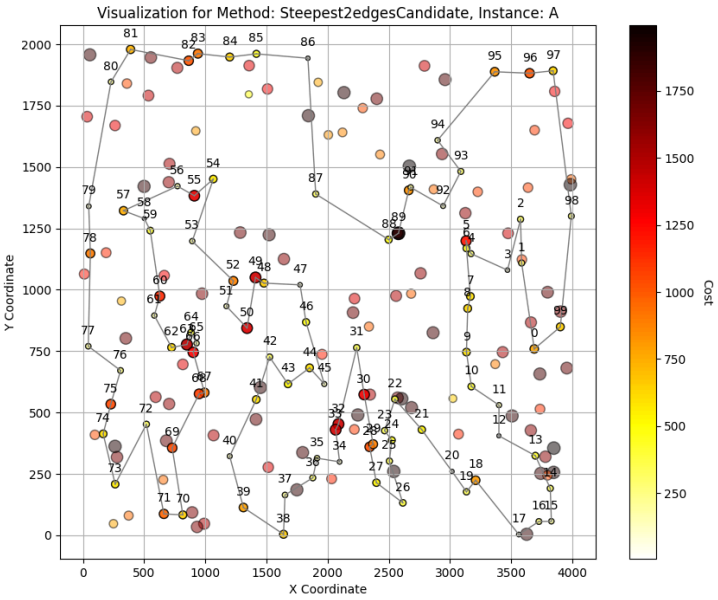
---

## 2D visualisations of best solutions

---

### Instance: A

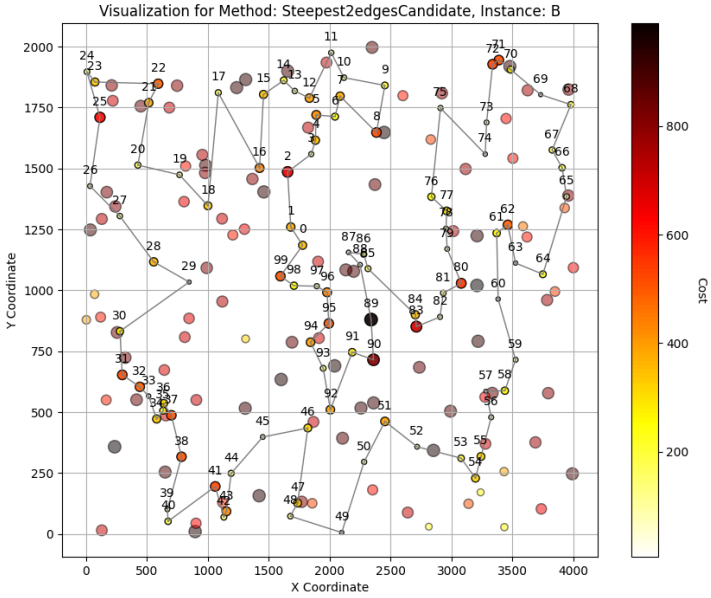
#### *Method: Steepest2edgesCandidate with score=74540*




---

## Instance: B

### *Method: Steepest2edgesCandidate with score=46036*



## Best solutions, indices

---

### Instance: A

#### *Method: Steepest2edgesCandidate*

Lowest Objective Function Value (f_val): 74540

Solution:

196, 40, 165, 185, 106, 3, 178, 52, 55, 57, 92, 145, 78, 31, 113, 175, 171, 16, 25, 44, 120, 2, 152, 97, 1, 101, 75, 86, 100, 26, 189, 94, 182, 136, 53, 180, 154, 135, 70, 127, 123, 162, 151, 133, 79, 63, 80, 176, 51, 109, 72, 59, 118, 115, 46, 198, 139, 159, 193, 41, 5, 42, 43, 105, 116, 65, 47, 149, 131, 35, 112, 4, 184, 177, 54, 48, 160, 34, 146, 22, 18, 108, 140, 93, 117, 143, 183, 137, 148, 33, 9, 62, 49, 14, 144, 21, 7, 164, 90, 81

---

### Instance: B

#### *Method: Steepest2edgesCandidate*

Lowest Objective Function Value (f_val): 46036

Solution:

11, 139, 43, 168, 195, 13, 145, 15, 189, 155, 3, 70, 132, 169, 188, 6, 134, 147, 51, 90, 122, 133, 10, 107, 40, 100, 63, 135, 131, 121, 1, 156, 198, 117, 193, 31, 54, 73, 136, 190, 80, 45, 175, 78, 5, 177, 21, 61, 36, 141, 77, 81, 153, 163, 89, 103, 113, 176, 194, 166, 86, 95, 130, 185, 179, 94, 47, 148, 60, 20, 28, 149, 4, 140, 183, 152, 34, 55, 18, 62, 128, 124, 106, 159, 143, 35, 109, 29, 0, 37, 41, 111, 82, 8, 104, 144, 160, 33, 138, 182

---

## Conclusions

Overall 2 edges swap provided better solutions than 2 nodes swap. the same goes for comparison between Greedy and random starting solutions, greedy start provided better results on average. However while considering the candidate moves together with previous methods, it provides results on the better end of the spectrum. For instance A it achieved worse results than most steepest local search methods, but still better than 2 nodes random method. However for instance B it achieved better results than all the previous weeks' methods. But what can not be omitted is the significant difference in computation time. It is nearly 2 times faster than the fastest methods from before, acquiring 15 and 16 millisecond time on average. Moreover, random starting method produced runs with exponentially longer computation time, which makes the candidate edge algorithm even more satisfactory. This algorithm provided very good results considering the computation time savings.