04-candidate-moves

November 24, 2024

Made by: Andrei Kulchyk (155489) and Fiodar Piatrovich (155174)

Github

1 Description of a problem

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.

2 Local Search with candidate moves

Function local_search($dataset, distance\_matrix, initial\_solution, strategy, intra\_search, use\_heuristic$) :

Set the solution as the initial solution

Identify selected nodes and non-selected nodes

Loop until no improvement can be found :

Search for intra-route neighbors based on intra_search type (node or edge).

If strategy is "greedy", take first giving improvement.

If use_heuristic, limit search to candidate nodes.

Search for inter-route neighbors between solution nodes and non-selected nodes.

If use_heuristic, limit non-selected search to candidate nodes.

If strategy is "greedy", take first giving improvement.

Combine intra-route and inter-route neighbors into all_neighbors

If there are no improving neighbors :

Exit the loop

If strategy is "greedy" :

Shuffle neighbors to get randomly either inter or intra neighbor

Else If strategy is "steepest" :

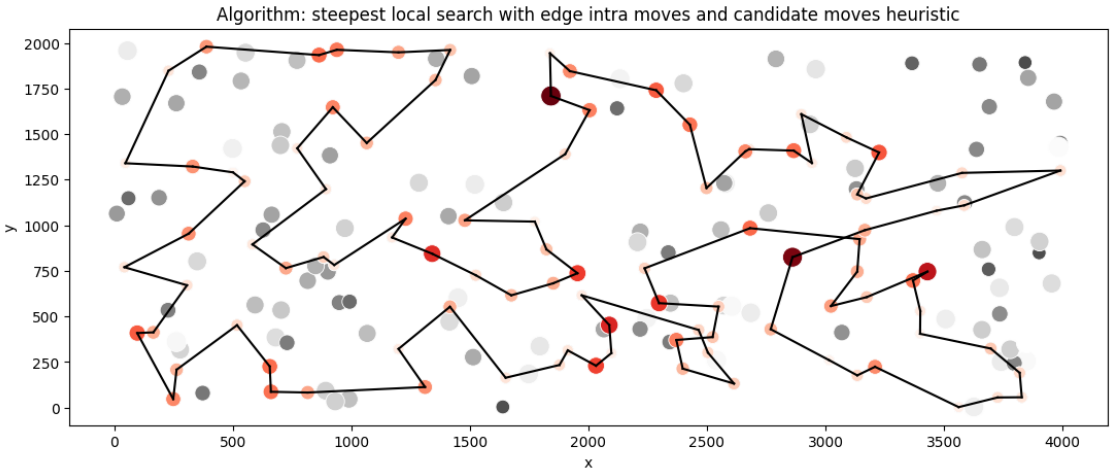
Choose the neighbor with the steepest improvement

Update solution, selected nodes, and non-selected nodes based on best neighbor

Returnthe final solution as a subset of the dataset

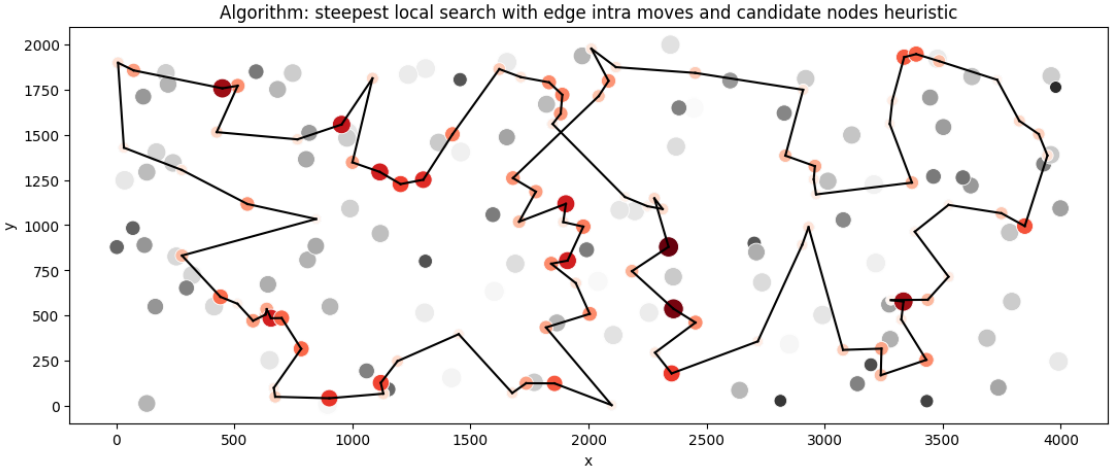
Results on Dataset A

```
Best solution: [152, 189, 94, 167, 55, 57, 129, 92, 91, 179, 145, 78, 31, 113,
175, 171, 16, 25, 44, 120, 2, 172, 52, 185, 40, 90, 165, 106, 178, 138, 14, 144,
49, 102, 62, 9, 148, 15, 114, 89, 183, 76, 23, 137, 51, 176, 80, 122, 79, 133,
151, 72, 59, 118, 65, 116, 43, 42, 115, 139, 68, 46, 0, 143, 117, 93, 140, 108,
18, 22, 159, 193, 41, 181, 34, 160, 54, 30, 10, 177, 184, 84, 4, 112, 127, 123,
162, 135, 154, 180, 158, 53, 182, 63, 97, 101, 75, 86, 26, 1]
Objective function statistics:
minimum = 76566
mean = 83201.29
maximum = 93668
```



Results on Dataset B

```
Best solution: [35, 0, 29, 168, 195, 13, 132, 169, 188, 134, 74, 118, 98, 51,
147, 71, 90, 122, 133, 44, 107, 40, 63, 135, 131, 121, 1, 198, 117, 193, 31, 54,
164, 73, 136, 190, 80, 162, 175, 142, 5, 177, 36, 61, 91, 141, 21, 82, 8, 104,
56, 160, 33, 49, 138, 11, 139, 145, 15, 70, 3, 155, 152, 34, 55, 18, 62, 95,
183, 140, 4, 149, 28, 20, 148, 47, 94, 172, 179, 185, 86, 166, 194, 176, 88,
113, 114, 127, 103, 163, 124, 106, 153, 97, 77, 81, 14, 111, 37, 109]
Objective function statistics:
minimum = 49463
mean = 54430.4
maximum = 61502
```



3 Summary

	Dataset A			\
	min	mean	max	seconds/iter
random_cycle_edge_steepest	72046.0	74033.715	78801.0	9.54
random_cycle_edge_candidate_heur	76566.0	83201.290	93668.0	1.23

	Dataset B			
	min	mean	max	seconds/iter
random_cycle_edge_steepest	45393.0	48264.78	50697.0	9.02
random_cycle_edge_candidate_heur	49463.0	54430.40	61502.0	1.27

4 Conclusion

`Candidate moves` heuristic improves significantly the computation time, but at cost of quality, the drop there is rather significant.