Assignment 3: Local search

Authors: Kiril Andrukh, 162069; Uladzislau Lukashevich, 155671.
Source code: link

Description of the 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.

Local Search

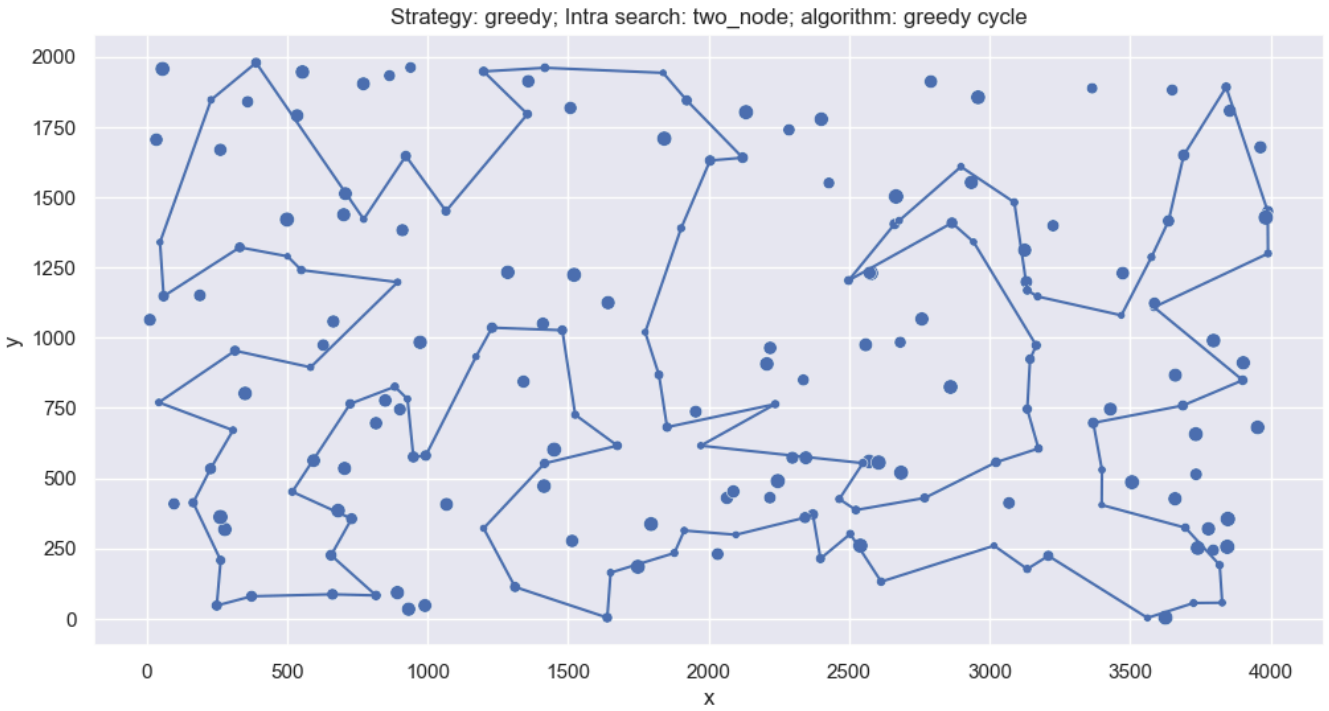
Pseudocode:

```
Initialize the cost of the initial solution and set it as the current solution
Identify selected nodes and non-selected nodes

Repeat until no improvement can be found:
- Search for intra-route neighbors based on the `intra_search` type (node or edge)
- Search for inter-route neighbors between selected and non-selected nodes
- Combine intra-route and inter-route neighbors into a single list (`all_neighbors`)

- If no improving neighbors exist:
  Exit the loop

- If strategy is `"greedy"`:
  Shuffle `all_neighbors` and select the first improving neighbor

- Else if strategy is `"steepest"`:
  Select the neighbor with the highest improvement

- Update the current solution, selected nodes, and non-selected nodes using the chosen
neighbor
- Add the improvement of the chosen neighbor to the total cost

Return the final solution as a subset of the dataset
```

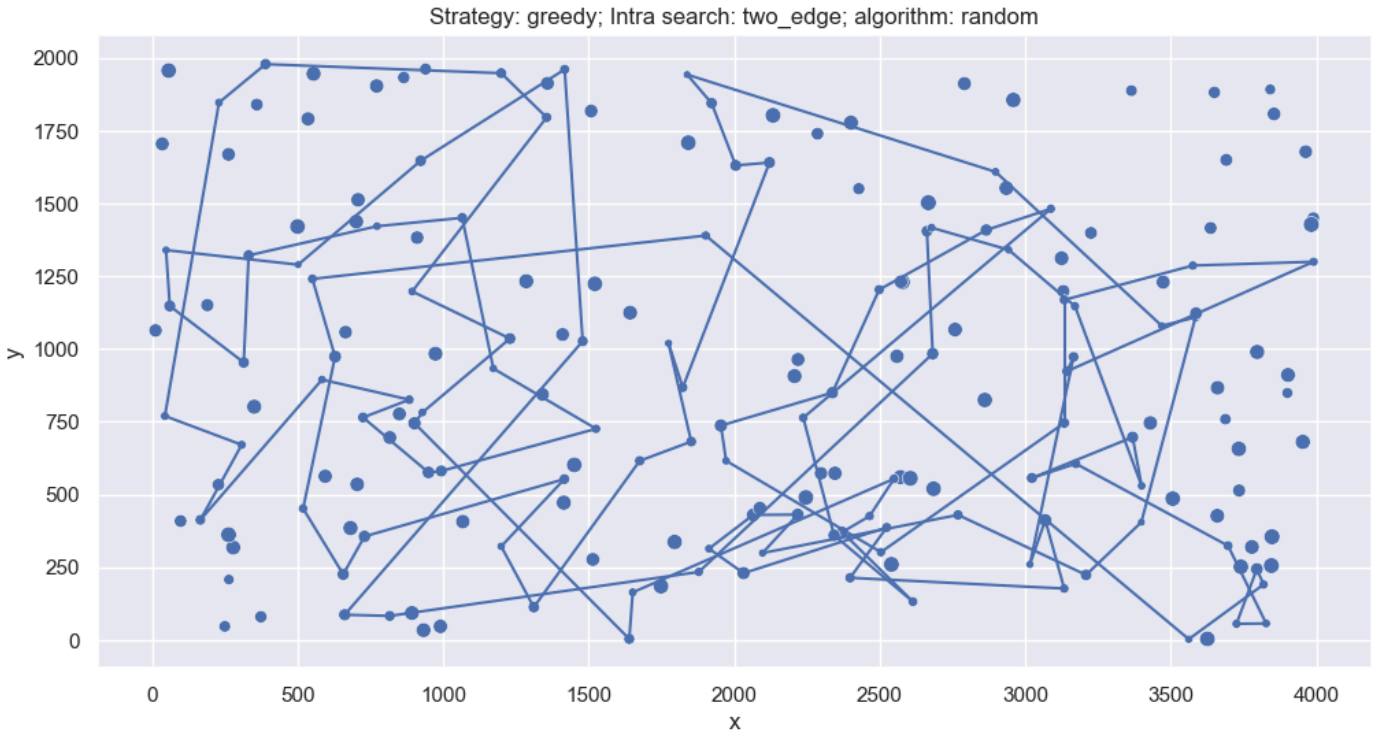
Dataset A:

Best solution: [20, 157, 92, 153, 187, 53, 176, 178, 15, 7, 71, 61, 154, 56, 50, 199, 31, 47, 109, 25, 70, 118, 126, 27, 198, 18, 90, 35, 75, 6, 88, 137, 41, 30, 122, 128, 4, 60, 140, 166, 152, 119, 98, 10, 11, 159, 42, 17, 174, 191, 141, 52, 170, 127, 83, 106, 67, 113, 105, 94, 172, 65, 14, 72, 91, 102, 136, 13, 64, 3, 132, 74, 28, 173, 101, 149, 80, 142, 24, 51, 150, 58, 63, 135, 85, 184, 110, 37, 120, 21, 107, 16, 190, 87, 129, 44, 73, 59, 160, 36]

Best solution: [158, 154, 70, 135, 180, 133, 151, 162, 127, 123, 149, 65, 116, 43, 42, 184, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 181, 146, 22, 18, 108, 159, 41, 193, 139, 115, 59, 118, 51, 46, 68, 140, 93, 0, 117, 143, 183, 89, 186, 23, 137, 176, 80, 79, 94, 63, 152, 97, 1, 2, 129, 92, 57, 55, 52, 49, 102, 148, 9, 62, 144, 14, 178, 106, 185, 165, 21, 164, 27, 90, 40, 81, 196, 179, 145, 78, 31, 113, 175, 171, 16, 25, 44, 120, 75, 101, 86, 26, 100, 53]



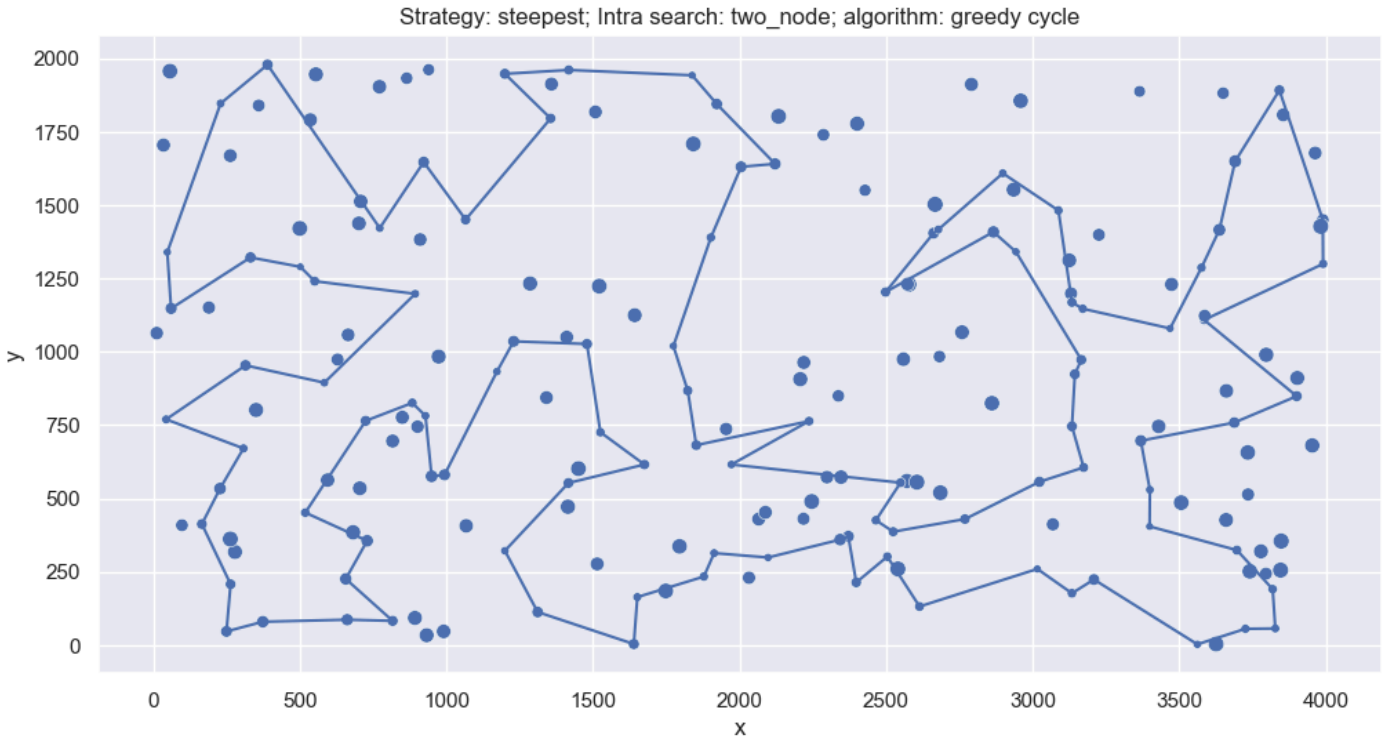
Best solution: [176, 30, 59, 189, 44, 148, 41, 26, 17, 120, 150, 186, 33, 83, 194, 197, 102, 69, 144, 78, 121, 51, 145, 138, 89, 107, 133, 8, 106, 5, 28, 12, 76, 180, 170, 55, 172, 185, 63, 72, 70, 132, 146, 52, 65, 14, 90, 174, 34, 140, 192, 43, 75, 29, 57, 188, 152, 100, 191, 118, 25, 6, 1, 50, 110, 86, 36, 116, 47, 13, 161, 175, 45, 134, 61, 127, 92, 10, 56, 35, 80, 21, 11, 68, 32, 187, 139, 104, 62, 103, 114, 93, 113, 58, 109, 73, 82, 81, 123, 46]



Best solution: [158, 154, 70, 135, 180, 133, 151, 162, 127, 123, 149, 65, 116, 43, 42, 184, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 181, 146, 22, 18, 108, 159, 41, 193, 139, 115, 59, 118, 51, 46, 68, 140, 93, 0, 117, 143, 183, 89, 186, 23, 137, 176, 80, 79, 94, 63, 152, 97, 1, 2, 129, 92, 57, 55, 52, 49, 102, 148, 9, 62, 144, 14, 178, 106, 185, 165, 21, 164, 27, 90, 40, 81, 196, 179, 145, 78, 31, 113, 175, 171, 16, 25, 44, 120, 75, 101, 86, 26, 100, 53]

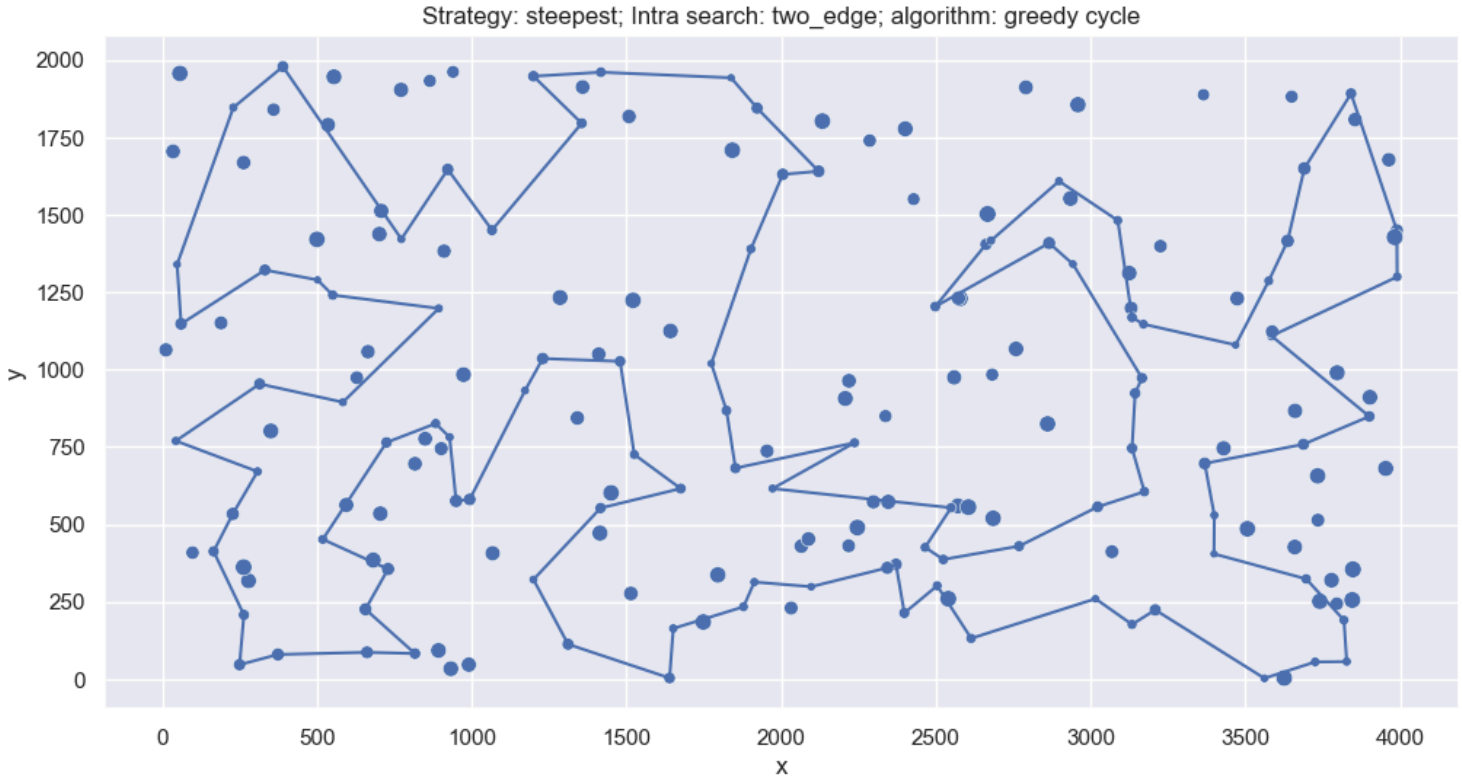
Best solution: [33, 177, 20, 104, 55, 139, 192, 190, 34, 87, 199, 72, 82, 140, 167, 80, 50, 158, 58, 172, 19, 147, 13, 129, 159, 189, 17, 118, 171, 145, 173, 128, 66, 120, 137, 150, 16, 42, 106, 187, 64, 78, 62, 186, 182, 60, 71, 45, 103, 8, 142, 107, 135, 112, 70, 127, 65, 109, 24, 7, 84, 30, 96, 154, 119, 170, 124, 49, 151, 90, 98, 9, 76, 132, 1, 130, 83, 100, 75, 191, 67, 51, 126, 198, 92, 74, 56, 54, 4, 156, 178, 133, 193, 28, 155, 0, 188, 138, 122, 41]

Best solution: [158, 154, 70, 135, 180, 133, 151, 162, 127, 123, 149, 65, 116, 43, 42, 184, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 181, 146, 22, 18, 108, 159, 41, 193, 139, 115, 59, 118, 51, 46, 68, 140, 93, 0, 117, 143, 183, 89, 186, 23, 137, 176, 80, 79, 94, 63, 152, 97, 1, 2, 129, 92, 57, 55, 52, 49, 102, 148, 9, 62, 144, 14, 178, 106, 185, 165, 21, 164, 27, 90, 40, 81, 196, 179, 145, 78, 31, 113, 175, 171, 16, 25, 44, 120, 75, 101, 86, 26, 100, 53]



Best solution: [130, 76, 166, 189, 9, 122, 1, 80, 157, 85, 168, 20, 112, 47, 124, 32, 119, 198, 125, 196, 79, 3, 159, 48, 178, 52, 195, 142, 43, 63, 73, 82, 167, 107, 104, 4, 177, 87, 102, 138, 117, 139, 31, 141, 86, 44, 128, 140, 149, 158, 98, 7, 108, 51, 188, 83, 38, 34, 127, 39, 194, 91, 96, 175, 30, 23, 11, 78, 153, 74, 12, 165, 77, 155, 120, 46, 42, 71, 36, 118, 37, 180, 57, 16, 197, 101, 53, 150, 55, 183, 151, 13, 26, 116, 106, 123, 135, 92, 6, 33]BB

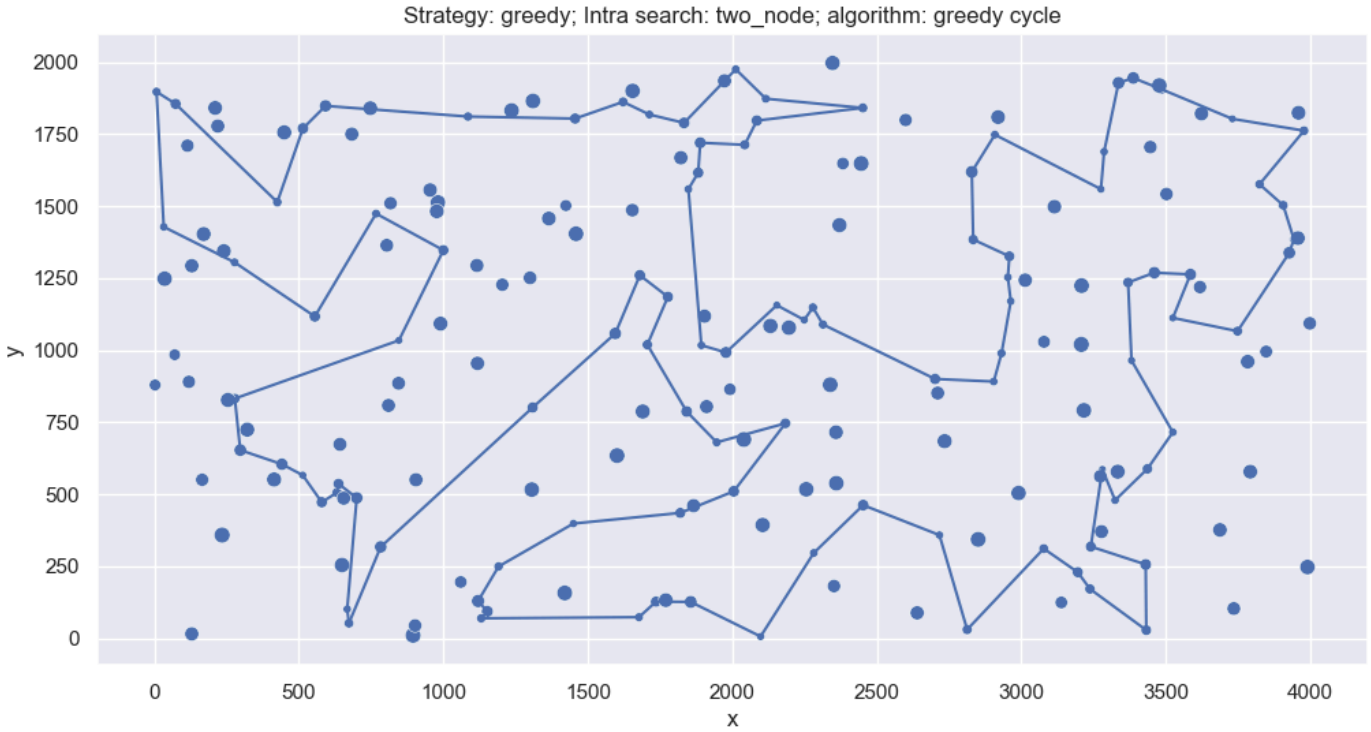
Best solution: [158, 154, 70, 135, 180, 133, 151, 162, 127, 123, 149, 65, 116, 43, 42, 184, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 181, 146, 22, 18, 108, 159, 41, 193, 139, 115, 59, 118, 51, 46, 68, 140, 93, 0, 117, 143, 183, 89, 186, 23, 137, 176, 80, 79, 94, 63, 152, 97, 1, 2, 129, 92, 57, 55, 52, 49, 102, 148, 9, 62, 144, 14, 178, 106, 185, 165, 21, 164, 27, 90, 40, 81, 196, 179, 145, 78, 31, 113, 175, 171, 16, 25, 44, 120, 75, 101, 86, 26, 100, 53]



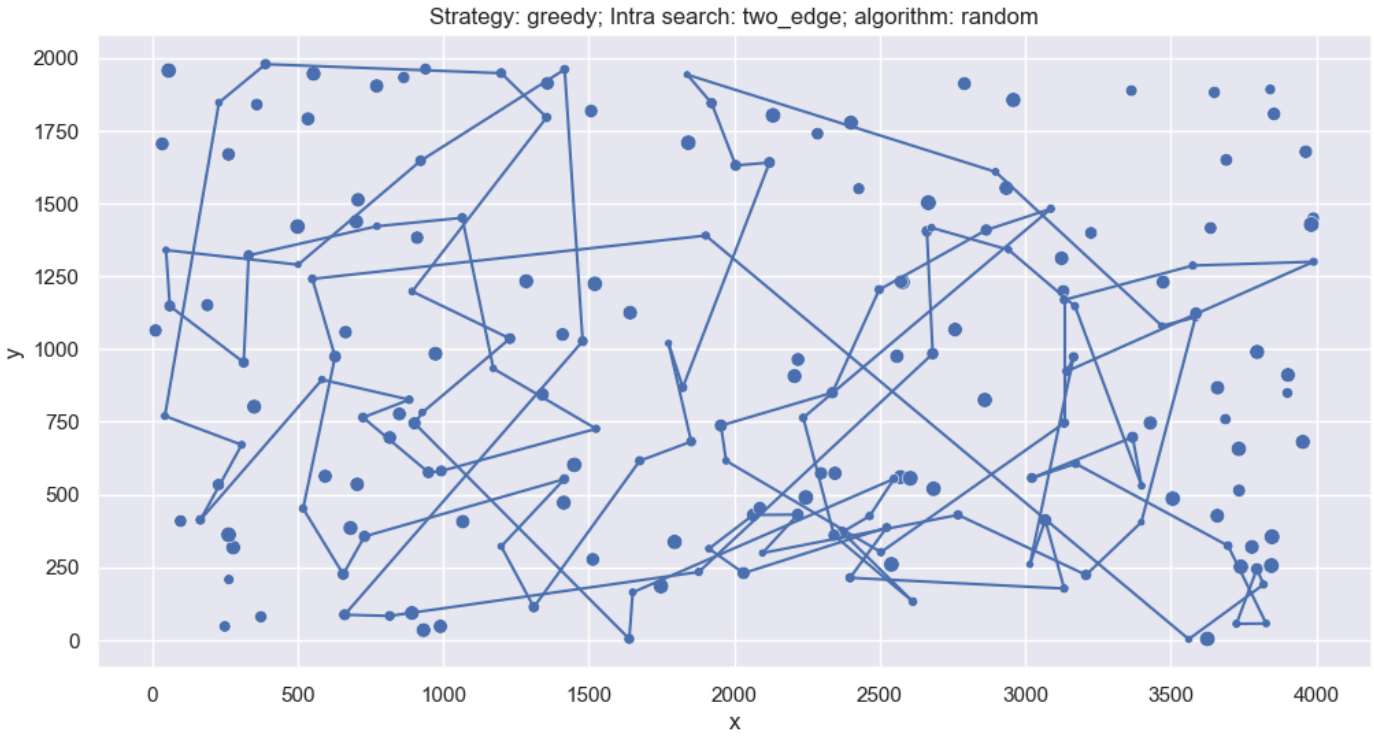
Dataset B:

Best solution: [84, 86, 192, 93, 21, 177, 44, 97, 70, 186, 47, 75, 82, 79, 76, 164, 61, 145, 194, 37, 141, 27, 153, 179, 64, 7, 165, 122, 90, 116, 137, 176, 24, 170, 109, 5, 101, 54, 38, 0, 49, 56, 166, 135, 19, 9, 31, 108, 105, 129, 13, 184, 115, 104, 155, 58, 29, 36, 113, 118, 149, 39, 43, 25, 98, 95, 12, 161, 146, 80, 103, 111, 119, 16, 48, 1, 134, 124, 69, 73, 77, 63, 41, 196, 100, 117, 174, 33, 130, 133, 123, 67, 127, 85, 102, 89, 62, 45, 114, 154]

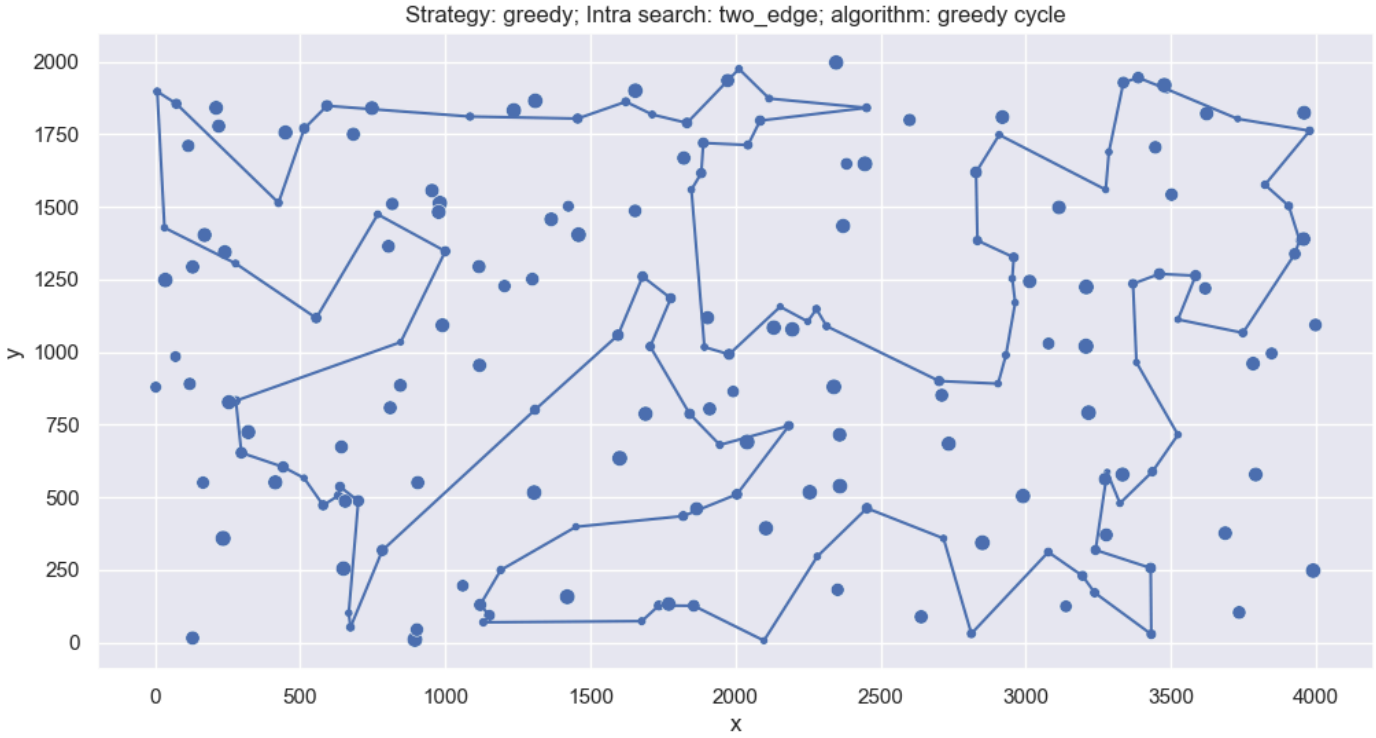
Best solution: [55, 62, 124, 106, 143, 35, 109, 0, 29, 160, 33, 168, 195, 13, 145, 15, 155, 3, 70, 132, 169, 188, 6, 147, 10, 133, 122, 107, 40, 63, 135, 131, 90, 51, 121, 1, 156, 198, 117, 193, 31, 54, 73, 190, 80, 136, 25, 182, 139, 11, 138, 104, 8, 144, 111, 82, 21, 177, 5, 45, 142, 78, 175, 36, 61, 91, 141, 77, 81, 153, 187, 163, 89, 127, 137, 114, 103, 176, 113, 194, 166, 86, 179, 66, 94, 47, 148, 60, 20, 28, 99, 185, 130, 95, 183, 140, 152, 170, 34, 18]



Best solution: [166, 99, 148, 47, 20, 28, 152, 3, 15, 145, 70, 13, 195, 109, 160, 29, 12, 0, 77, 153, 81, 82, 182, 51, 125, 131, 121, 135, 63, 40, 122, 90, 147, 155, 184, 170, 34, 55, 18, 95, 185, 128, 106, 159, 97, 141, 91, 36, 21, 8, 11, 188, 169, 33, 138, 25, 5, 175, 142, 78, 61, 104, 139, 43, 168, 134, 118, 31, 54, 136, 164, 193, 117, 198, 156, 1, 73, 80, 190, 177, 111, 35, 143, 124, 62, 86, 183, 140, 94, 179, 172, 194, 113, 163, 187, 165, 127, 89, 103, 176]

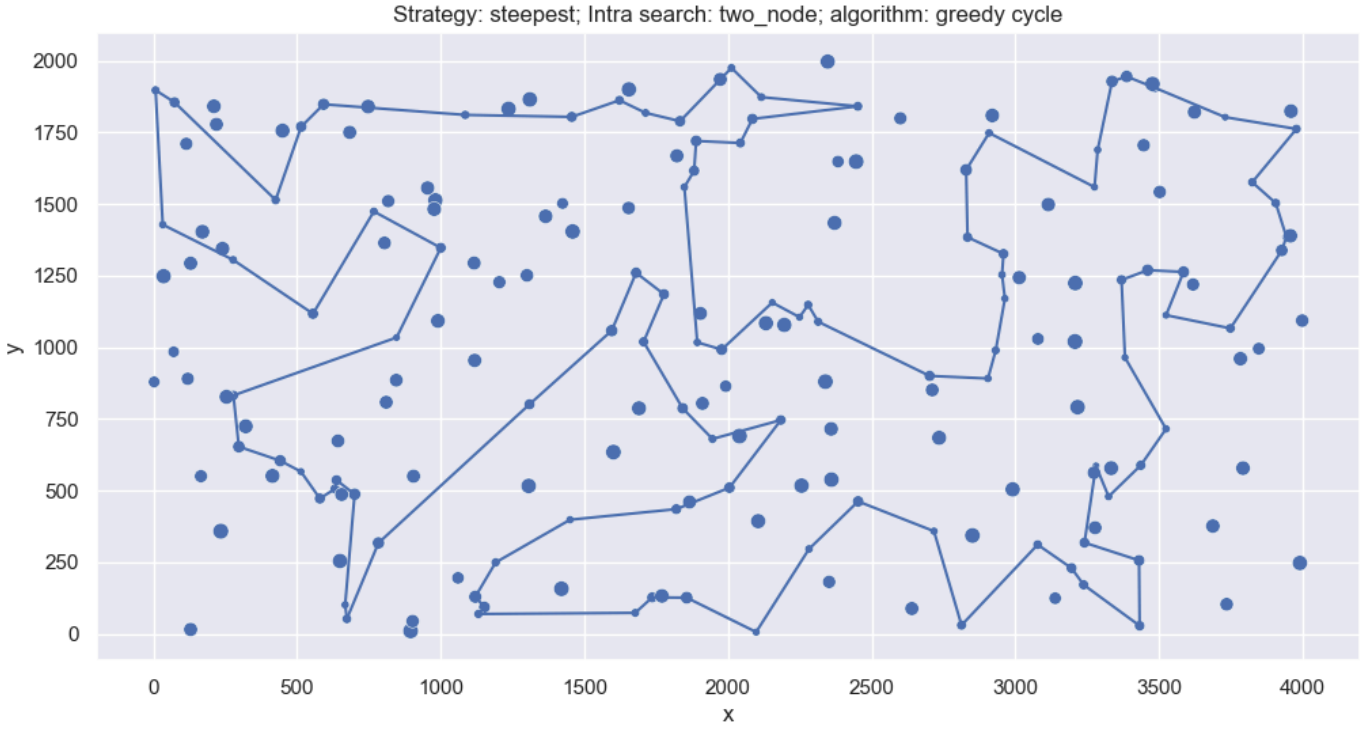


Best solution: [55, 62, 124, 106, 143, 35, 109, 0, 29, 160, 33, 168, 195, 13, 145, 15, 155, 3, 70, 132, 169, 188, 6, 147, 10, 133, 122, 107, 40, 63, 135, 131, 90, 51, 121, 1, 156, 198, 117, 193, 31, 54, 73, 190, 80, 136, 25, 182, 139, 11, 138, 104, 8, 144, 111, 82, 21, 177, 5, 45, 142, 78, 175, 36, 61, 91, 141, 77, 81, 153, 187, 163, 89, 127, 137, 114, 103, 176, 113, 194, 166, 86, 179, 66, 94, 47, 148, 60, 20, 28, 99, 185, 130, 95, 183, 140, 152, 170, 34, 18]



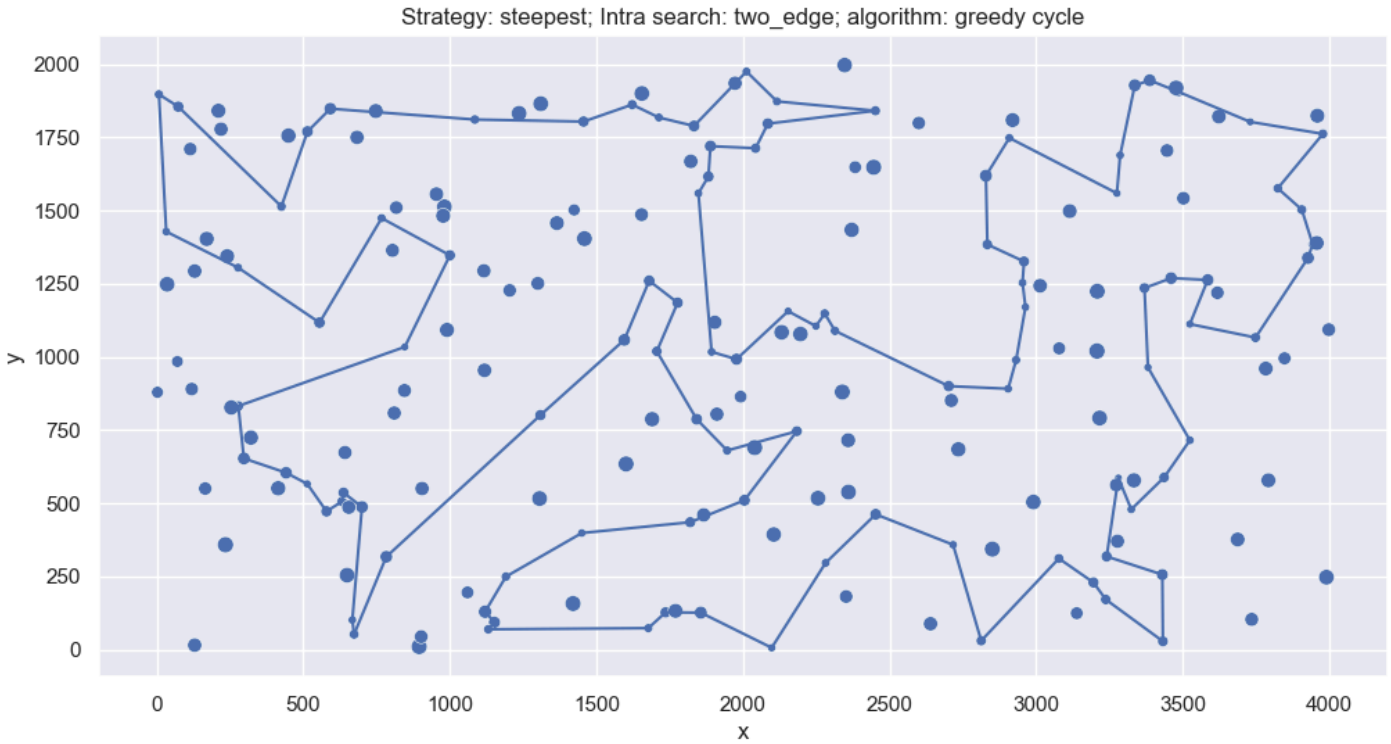
Best solution: [33, 177, 20, 104, 55, 139, 192, 190, 34, 87, 199, 72, 82, 140, 167, 80, 50, 158, 58, 172, 19, 147, 13, 129, 159, 189, 17, 118, 171, 145, 173, 128, 66, 120, 137, 150, 16, 42, 106, 187, 64, 78, 62, 186, 182, 60, 71, 45, 103, 8, 142, 107, 135, 112, 70, 127, 65, 109, 24, 7, 84, 30, 96, 154, 119, 170, 124, 49, 151, 90, 98, 9, 76, 132, 1, 130, 83, 100, 75, 191, 67, 51, 126, 198, 92, 74, 56, 54, 4, 156, 178, 133, 193, 28, 155, 0, 188, 138, 122, 41]

Best solution: [55, 62, 124, 106, 143, 35, 109, 0, 29, 160, 33, 168, 195, 13, 145, 15, 155, 3, 70, 132, 169, 188, 6, 147, 10, 133, 122, 107, 40, 63, 135, 131, 90, 51, 121, 1, 156, 198, 117, 193, 31, 54, 73, 190, 80, 136, 25, 182, 139, 11, 138, 104, 8, 144, 111, 82, 21, 177, 5, 45, 142, 78, 175, 36, 61, 91, 141, 77, 81, 153, 187, 163, 89, 127, 137, 114, 103, 176, 113, 194, 166, 86, 179, 66, 94, 47, 148, 60, 20, 28, 99, 185, 130, 95, 183, 140, 152, 170, 34, 18]



Best solution: [130, 76, 166, 189, 9, 122, 1, 80, 157, 85, 168, 20, 112, 47, 124, 32, 119, 198, 125, 196, 79, 3, 159, 48, 178, 52, 195, 142, 43, 63, 73, 82, 167, 107, 104, 4, 177, 87, 102, 138, 117, 139, 31, 141, 86, 44, 128, 140, 149, 158, 98, 7, 108, 51, 188, 83, 38, 34, 127, 39, 194, 91, 96, 175, 30, 23, 11, 78, 153, 74, 12, 165, 77, 155, 120, 46, 42, 71, 36, 118, 37, 180, 57, 16, 197, 101, 53, 150, 55, 183, 151, 13, 26, 116, 106, 123, 135, 92, 6, 33]BB

Best solution: [55, 62, 124, 106, 143, 35, 109, 0, 29, 160, 33, 168, 195, 13, 145, 15, 155, 3, 70, 132, 169, 188, 6, 147, 10, 133, 122, 107, 40, 63, 135, 131, 90, 51, 121, 1, 156, 198, 117, 193, 31, 54, 73, 190, 80, 136, 25, 182, 139, 11, 138, 104, 8, 144, 111, 82, 21, 177, 5, 45, 142, 78, 175, 36, 61, 91, 141, 77, 81, 153, 187, 163, 89, 127, 137, 114, 103, 176, 113, 194, 166, 86, 179, 66, 94, 47, 148, 60, 20, 28, 99, 185, 130, 95, 183, 140, 152, 170, 34, 18]



Final table:

Method	Dataset A	Dataset B
Random generation, node, greedy	106894.84(95970-118282)	72435.175(60732-81134)
Greedy cycle, node, greedy	71869.84(71239-72849)	46751.435(45290-48316)
Random generation, edge, greedy	106568.28(95193-119478)	72420.31(61679-86503)
Greedy cycle, edge, greedy	71867.985(71239-72849)	46751.03(45290-48316)
Random generation, node, steepest	108506.295(96433-120153)	75609.58(67063-85889)
Greedy cycle, node, steepest	71867.735(71239-72849)	46751.42(45290-48316)
Random generation, edge, steepest	108418.825(95992-120869)	75185.49(64652-84818)
Greedy cycle, edge, steepest	71867.735(71239-72849)	46751.42(45290-48316)
Greedy cycle	72071.915(71263-73154)	46903.73(45312-48623)
Random generation	265009.32(243393-296391)	212375.76(190669-241913)

Conclusion:

Local search does not increase the performance of Greedy cycle a lot. There seems to be the ceiling of 71263/45920 for Dataset A/B respectively as min value, which is a better objective function than in Greedy cycle alone, and the solution for greedy in each dataset seems the same whatever the method is.

Local search drastically improves the performance of Random generation. The objective function is two times better than the original random generation approach.

Running the algorithm takes a while, the greedy cycle solution converges faster than random solution, since there are less changes to be applied.