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.

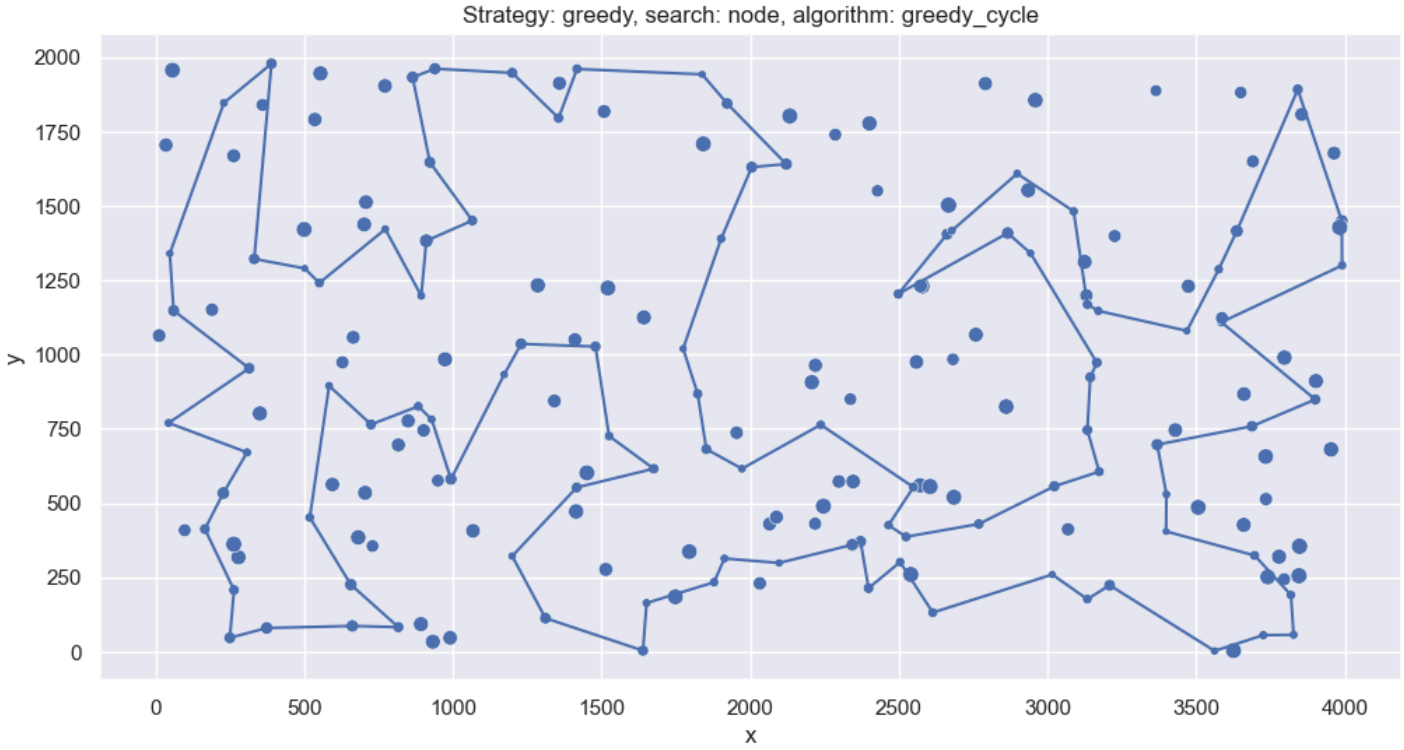
Local Search

Pseudocode:

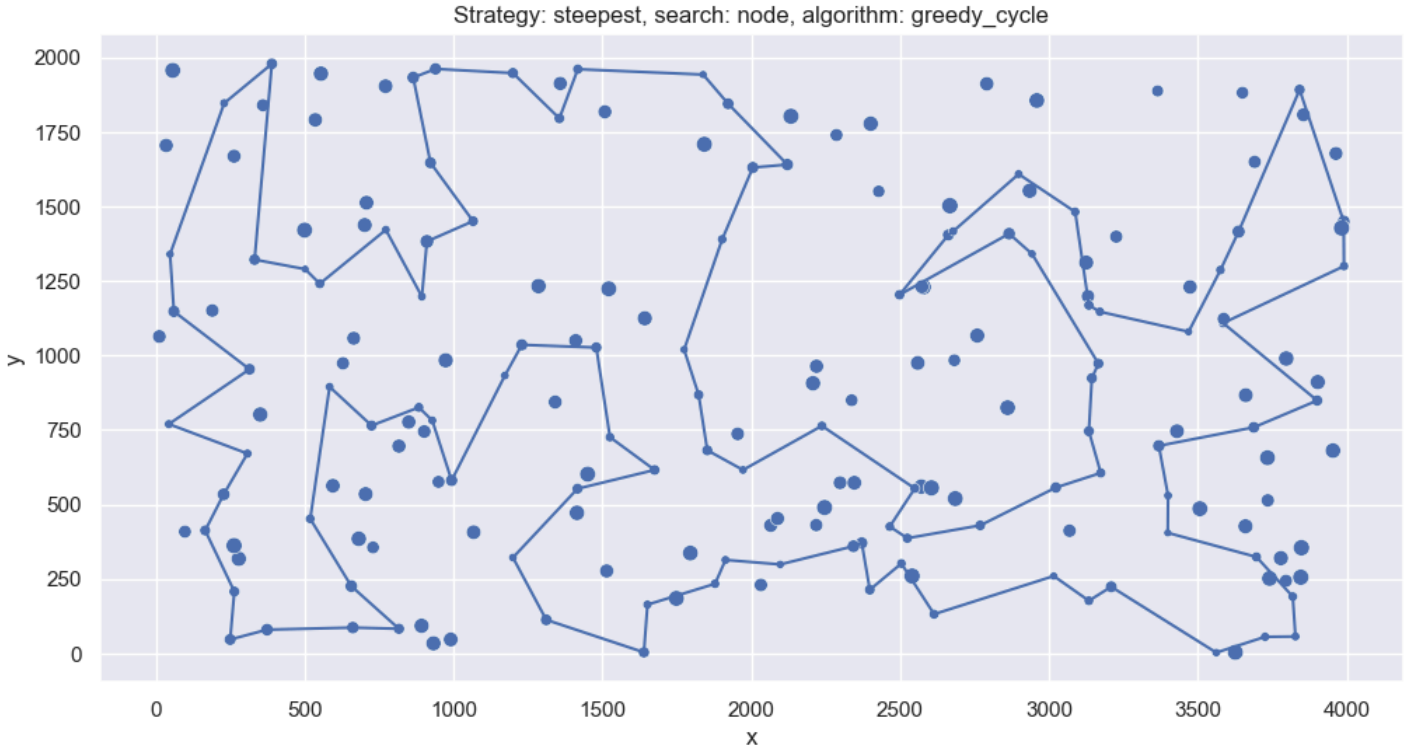
```
    Initialize cost of the initial solution
    Set the solution as the initial solution
    Identify selected nodes and non-selected nodes

    Loop until no improvement can be found:
        Search for intra-route neighbors based on intra_search type (node or edge)
        Search for inter-route neighbors between solution nodes and non-selected nodes
        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 and select the first improving neighbor
        Else If strategy is "steepest":
            Choose the neighbor with the steepest improvement

        Update solution, selected nodes, and non-selected nodes based on the best
neighbor
        Update cost by adding the improvement of the best neighbor

    Return the final solution
```

Dataset A:

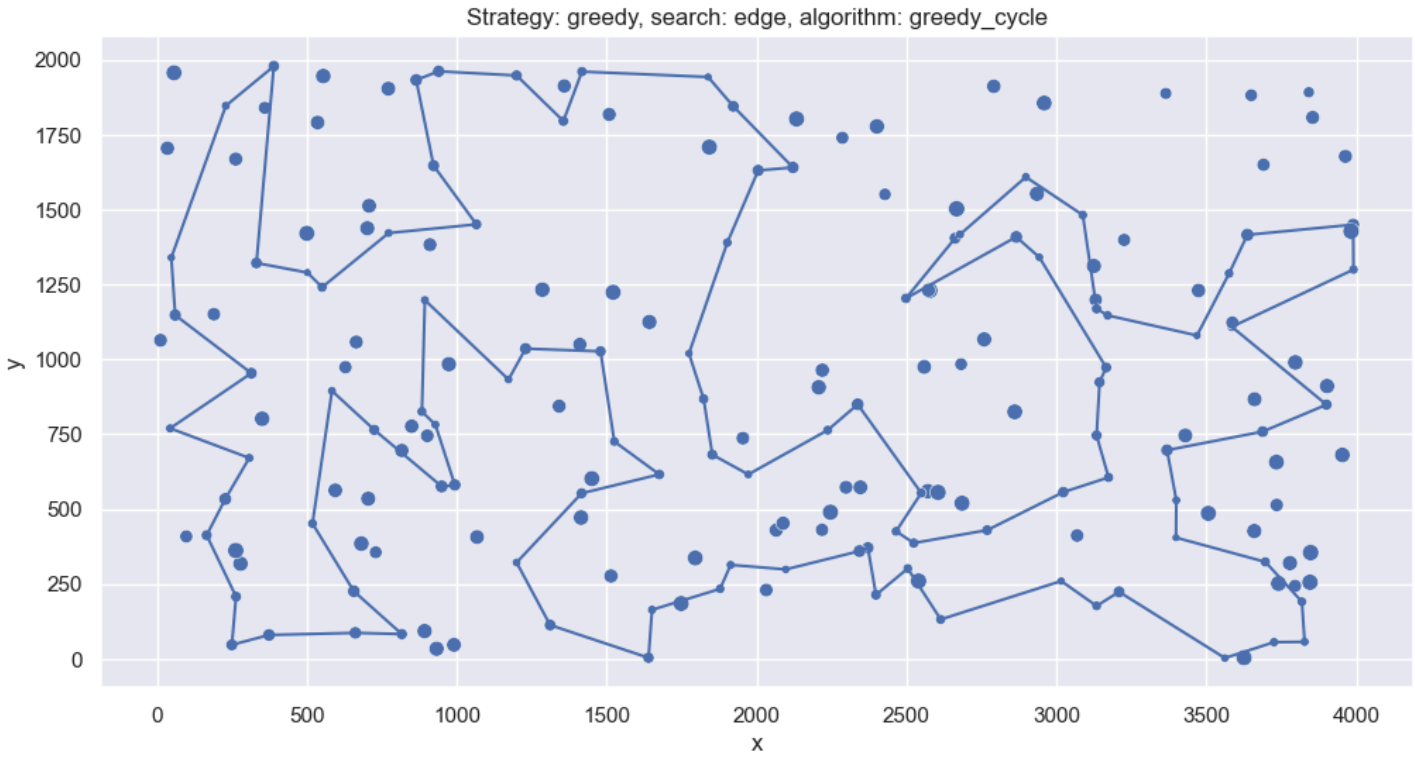
Best solution: [198, 115, 139, 41, 193, 159, 108, 18, 22, 146, 181, 34, 160, 48, 54, 177, 10, 190, 4, 112, 84, 184, 42, 43, 116, 65, 149, 59, 118, 51, 151, 133, 162, 123, 127, 70, 135, 154, 180, 53, 100, 26, 86, 101, 75, 120, 44, 25, 16, 171, 175, 113, 31, 78, 145, 179, 196, 81, 40, 90, 27, 164, 39, 165, 185, 106, 178, 14, 144, 62, 9, 148, 102, 49, 52, 55, 57, 92, 129, 2, 1, 97, 152, 94, 63, 79, 80, 176, 137, 23, 186, 89, 183, 143, 0, 117, 93, 140, 68, 46]



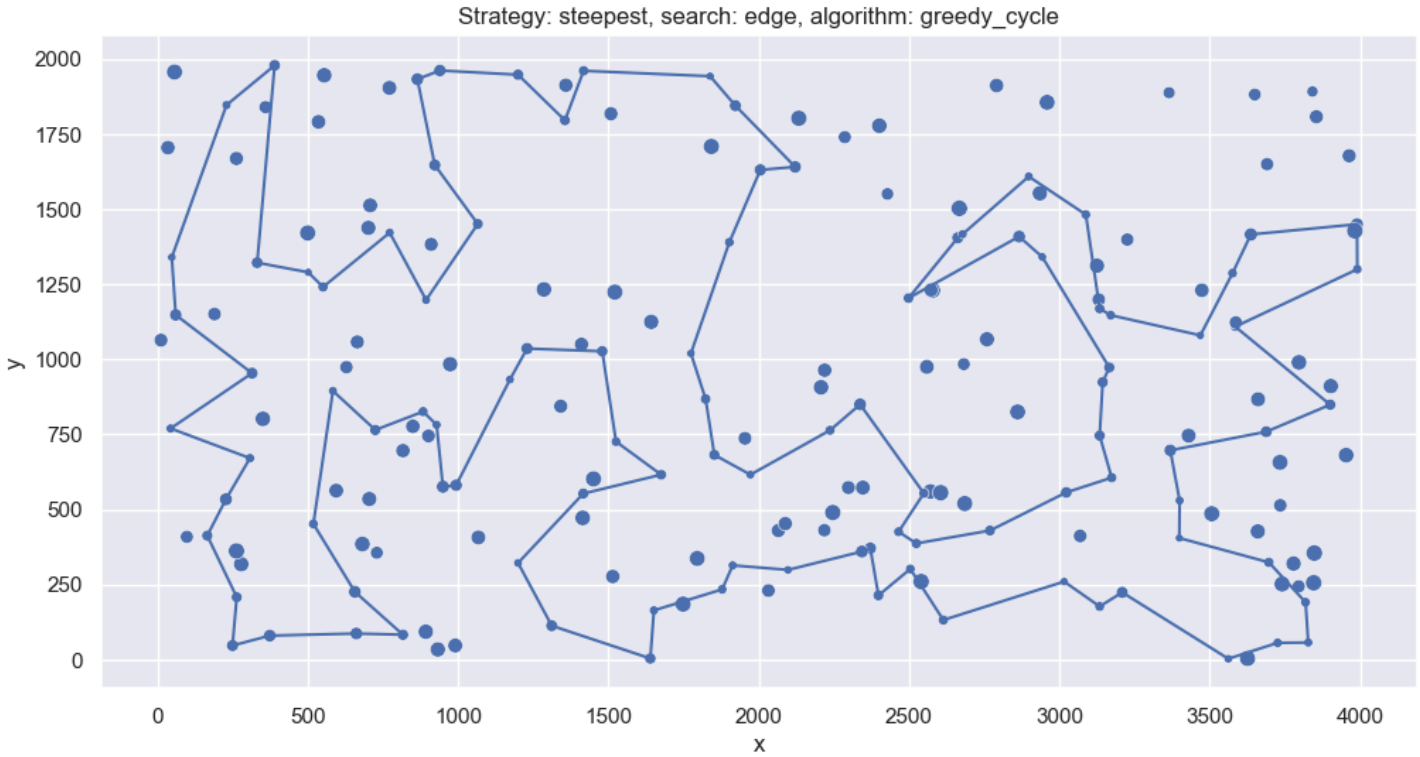
Best solution: [198, 115, 139, 41, 193, 159, 108, 18, 22, 146, 181, 34, 160, 48, 54, 177, 10, 190, 4, 112, 84, 184, 42, 43, 116, 65, 149, 59, 118, 51, 151, 133, 162, 123, 127, 70, 135, 154, 180, 53, 100, 26, 86, 101, 75, 120, 44, 25, 16, 171, 175, 113, 31, 78, 145, 179, 196, 81, 40, 90, 27, 164, 39, 165, 185, 106, 178, 14, 144, 62, 9, 148, 102, 49, 52, 55, 57, 92, 129, 2, 1, 97, 152, 94, 63, 79, 80, 176, 137, 23, 186, 89, 183, 143, 0, 117, 93, 140, 68, 46]



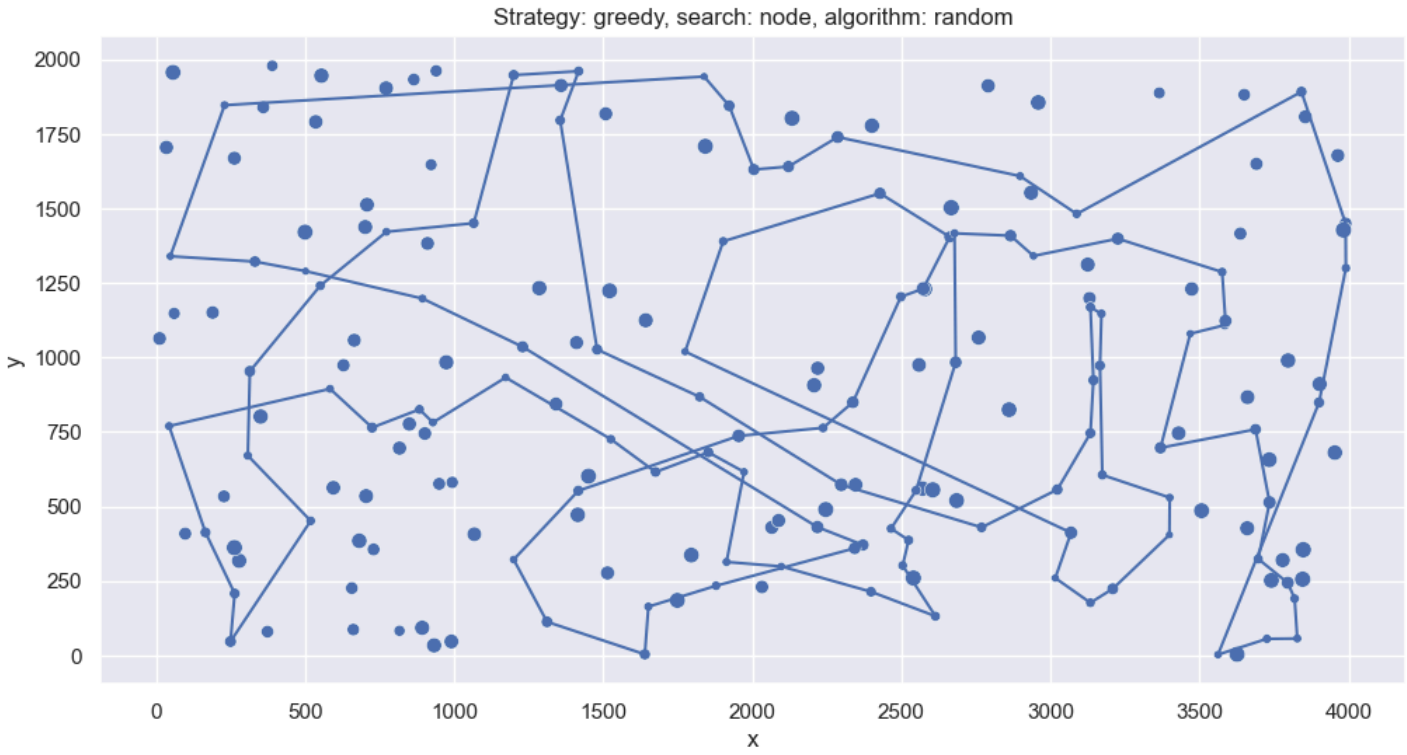
Best solution: [65, 116, 115, 59, 118, 51, 151, 133, 162, 123, 127, 70, 135, 154, 180, 53, 100, 26, 86, 101, 75, 120, 44, 25, 16, 171, 175, 113, 31, 78, 145, 179, 196, 81, 40, 90, 27, 39, 165, 185, 106, 178, 14, 144, 62, 9, 148, 102, 49, 52, 55, 57, 92, 129, 2, 1, 97, 152, 124, 94, 63, 79, 80, 176, 137, 23, 186, 89, 183, 143, 0, 117, 93, 140, 68, 46, 139, 41, 193, 159, 108, 18, 22, 146, 181, 34, 160, 48, 54, 177, 10, 190, 4, 112, 84, 184, 42, 43, 131, 149]



Best solution: [162, 133, 151, 51, 118, 59, 149, 131, 65, 116, 43, 42, 184, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 181, 146, 22, 18, 108, 159, 193, 41, 139, 115, 46, 68, 140, 93, 117, 0, 143, 183, 89, 186, 23, 137, 176, 80, 79, 63, 94, 124, 152, 97, 1, 2, 129, 92, 57, 55, 52, 49, 102, 148, 9, 62, 144, 14, 178, 106, 185, 165, 39, 27, 90, 40, 81, 196, 179, 145, 78, 31, 113, 175, 171, 16, 25, 44, 120, 75, 101, 86, 26, 100, 53, 180, 154, 135, 70, 127, 123]

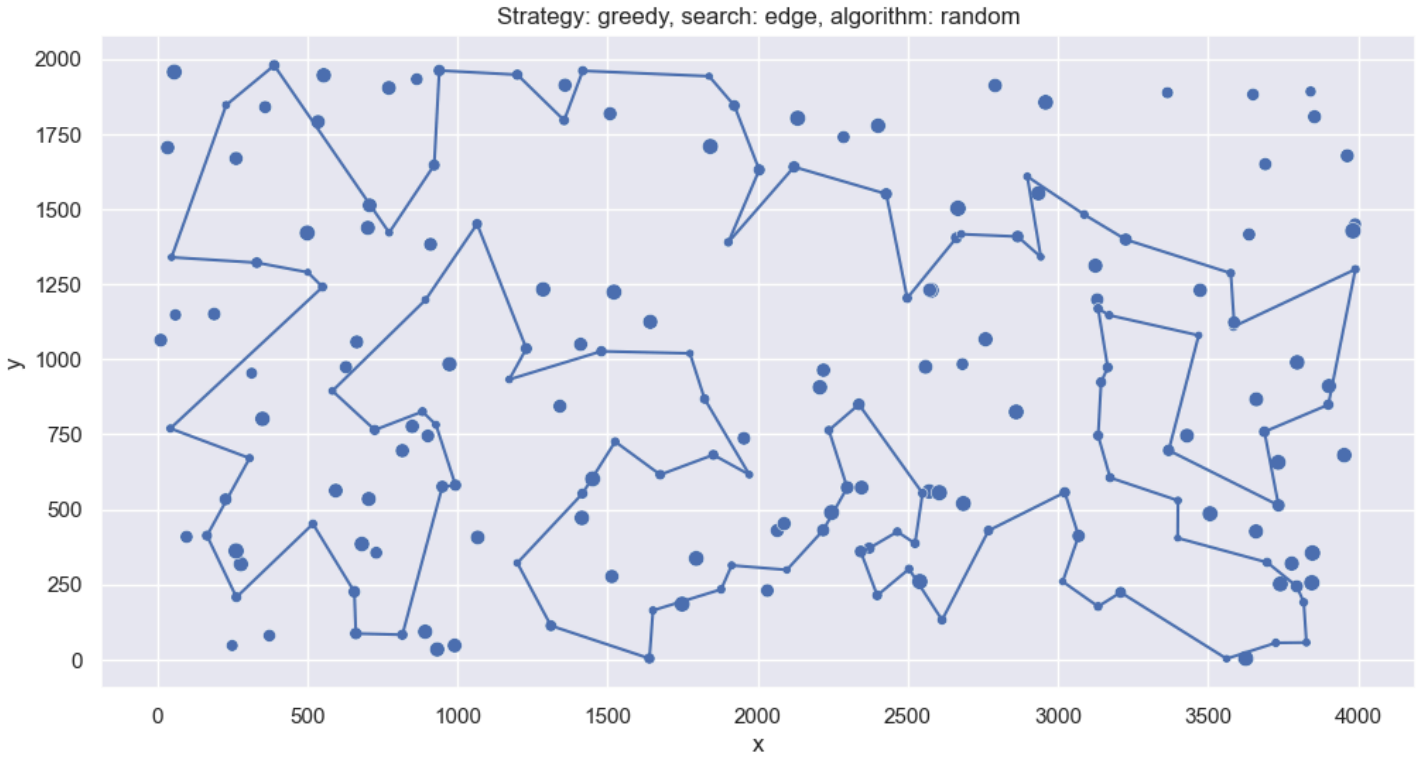


Best solution: [18, 183, 89, 23, 186, 114, 144, 14, 164, 27, 90, 81, 16, 171, 175, 113, 56, 31, 157, 196, 179, 185, 40, 119, 165, 138, 49, 102, 62, 167, 152, 97, 1, 101, 75, 86, 53, 180, 63, 79, 133, 151, 59, 65, 116, 43, 42, 34, 54, 177, 10, 184, 160, 181, 41, 139, 46, 117, 143, 0, 51, 80, 189, 2, 129, 57, 55, 178, 106, 52, 92, 145, 78, 25, 44, 120, 82, 176, 137, 15, 9, 37, 148, 124, 94, 122, 162, 123, 127, 70, 135, 154, 100, 26, 121, 118, 115, 193, 159, 22]



Best solution: [68, 41, 193, 159, 22, 146, 195, 181, 42, 43, 116, 65, 131, 149, 123, 135, 133, 151, 162, 35, 84, 4, 112, 59, 118, 115, 139, 46, 176, 80, 100, 26, 97, 1, 152, 2, 129, 178, 106, 52, 55, 57, 92, 145, 78, 25, 44, 120, 75, 101, 86, 63, 79, 51, 184, 160, 48, 177, 54, 30, 34, 18, 69, 108, 140, 93, 117, 0, 143, 183, 89, 186, 23, 137, 180, 154, 53, 121, 94, 124, 148, 9, 62, 49, 16, 171, 175, 113, 31, 196, 40, 90, 27, 39, 165, 119, 185, 14, 144, 15]

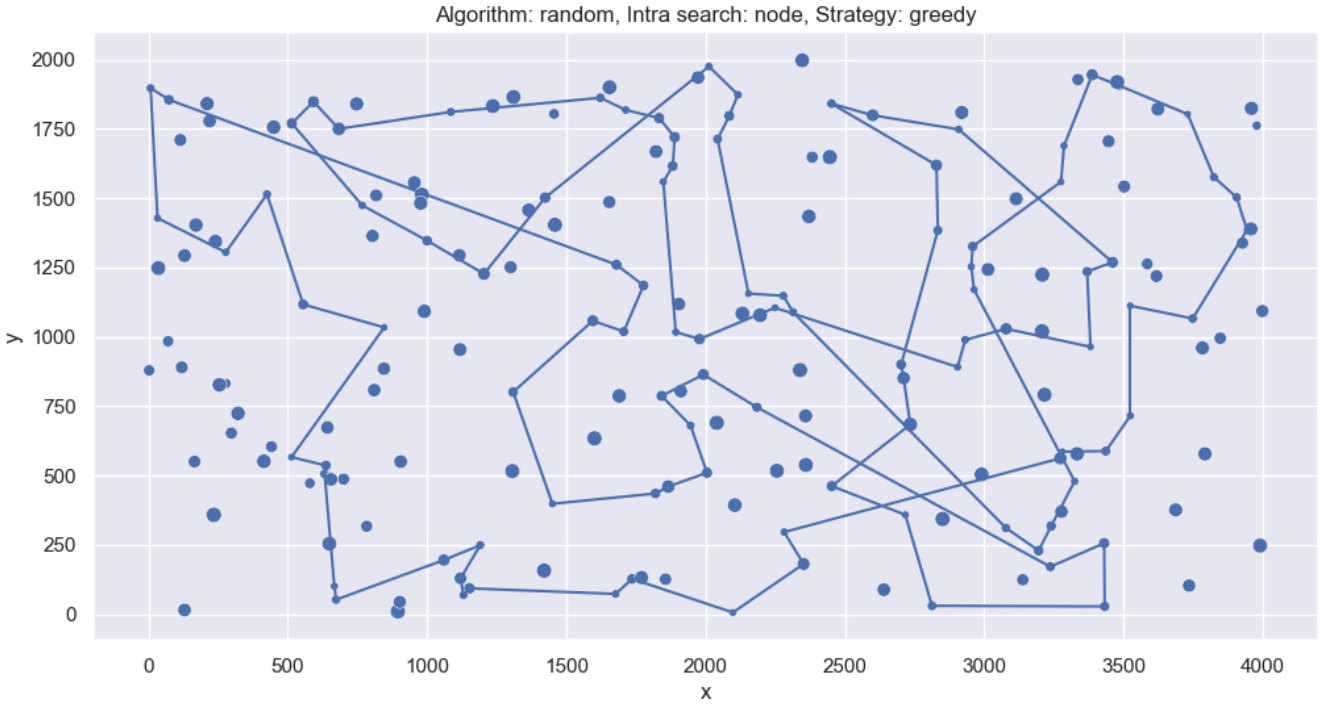
Best solution: [152, 1, 97, 26, 100, 86, 101, 75, 2, 129, 82, 120, 44, 25, 16, 171, 175, 113, 56, 31, 78, 145, 92, 57, 55, 52, 178, 106, 185, 179, 157, 196, 81, 90, 40, 119, 165, 138, 14, 144, 49, 102, 62, 9, 148, 15, 186, 137, 23, 89, 183, 143, 0, 117, 93, 68, 139, 108, 18, 22, 159, 193, 41, 34, 160, 54, 177, 184, 84, 4, 112, 131, 149, 65, 116, 43, 42, 115, 46, 118, 59, 51, 176, 80, 63, 79, 133, 151, 162, 123, 127, 70, 135, 154, 180, 53, 121, 189, 94, 124]



Best solution: [176, 51, 151, 162, 133, 80, 79, 63, 94, 97, 152, 148, 9, 62, 102, 144, 14, 49, 178, 106, 52, 55, 185, 40, 165, 27, 90, 81, 196, 31, 113, 175, 171, 16, 78, 145, 92, 57, 129, 82, 25, 44, 120, 2, 75, 86, 101, 1, 26, 100, 121, 53, 158, 180, 154, 135, 70, 127, 123, 149, 131, 35, 184, 84, 112, 4, 190, 10, 177, 54, 48, 160, 34, 146, 22, 181, 42, 5, 43, 116, 65, 59, 118, 115, 46, 139, 41, 193, 159, 18, 108, 93, 117, 0, 143, 183, 89, 186, 23, 137]

Dataset B:

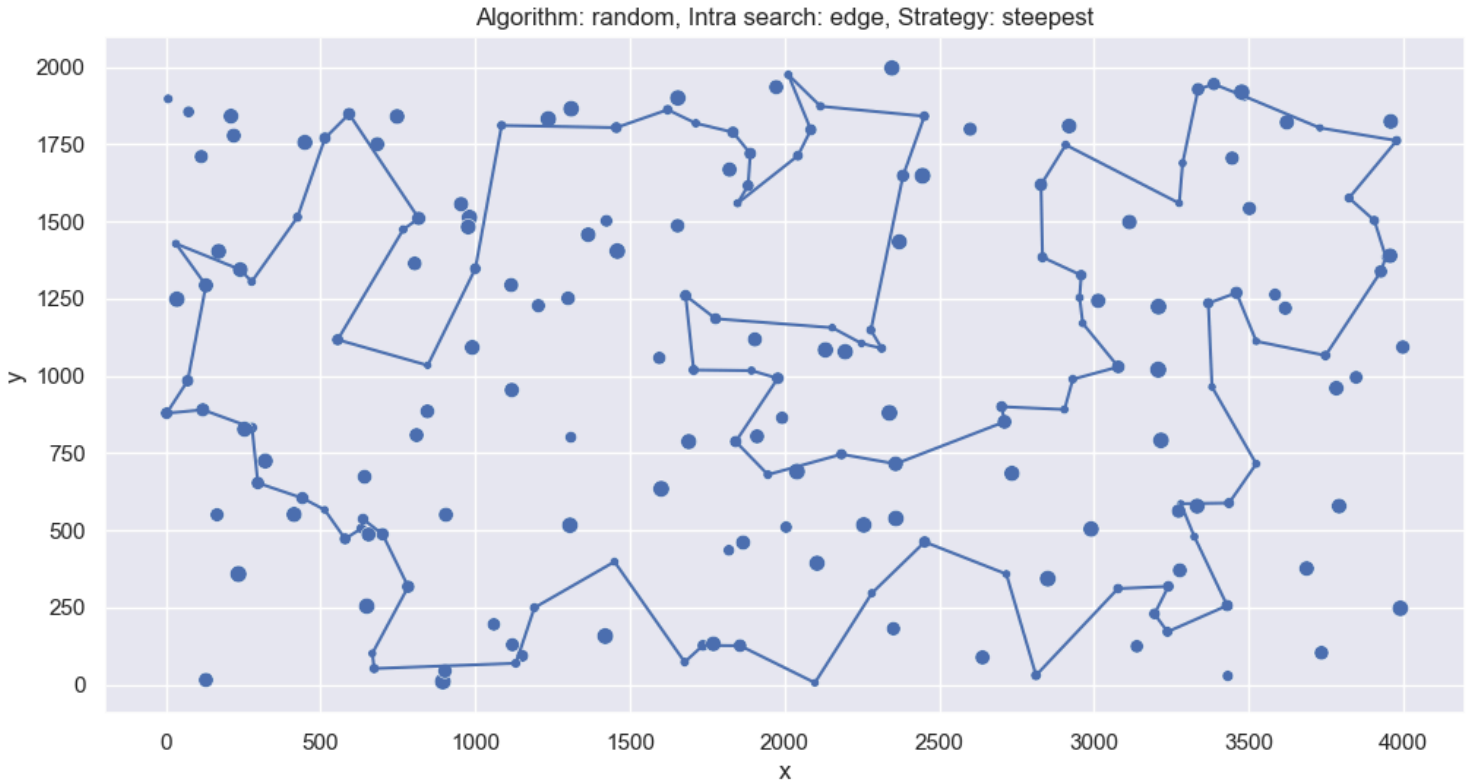
Best solution: [143, 159, 119, 81, 153, 187, 137, 114, 127, 111, 144, 104, 8, 82, 87, 21, 177, 25, 182, 138, 11, 139, 107, 40, 63, 135, 122, 131, 121, 117, 54, 31, 190, 80, 45, 5, 142, 175, 78, 36, 61, 141, 97, 77, 180, 176, 194, 166, 185, 179, 94, 47, 148, 20, 28, 149, 140, 183, 55, 18, 62, 113, 103, 89, 163, 35, 109, 29, 145, 15, 3, 70, 134, 118, 51, 90, 133, 10, 178, 147, 188, 169, 132, 13, 195, 168, 33, 160, 0, 106, 124, 128, 86, 95, 130, 152, 184, 155, 170, 34]



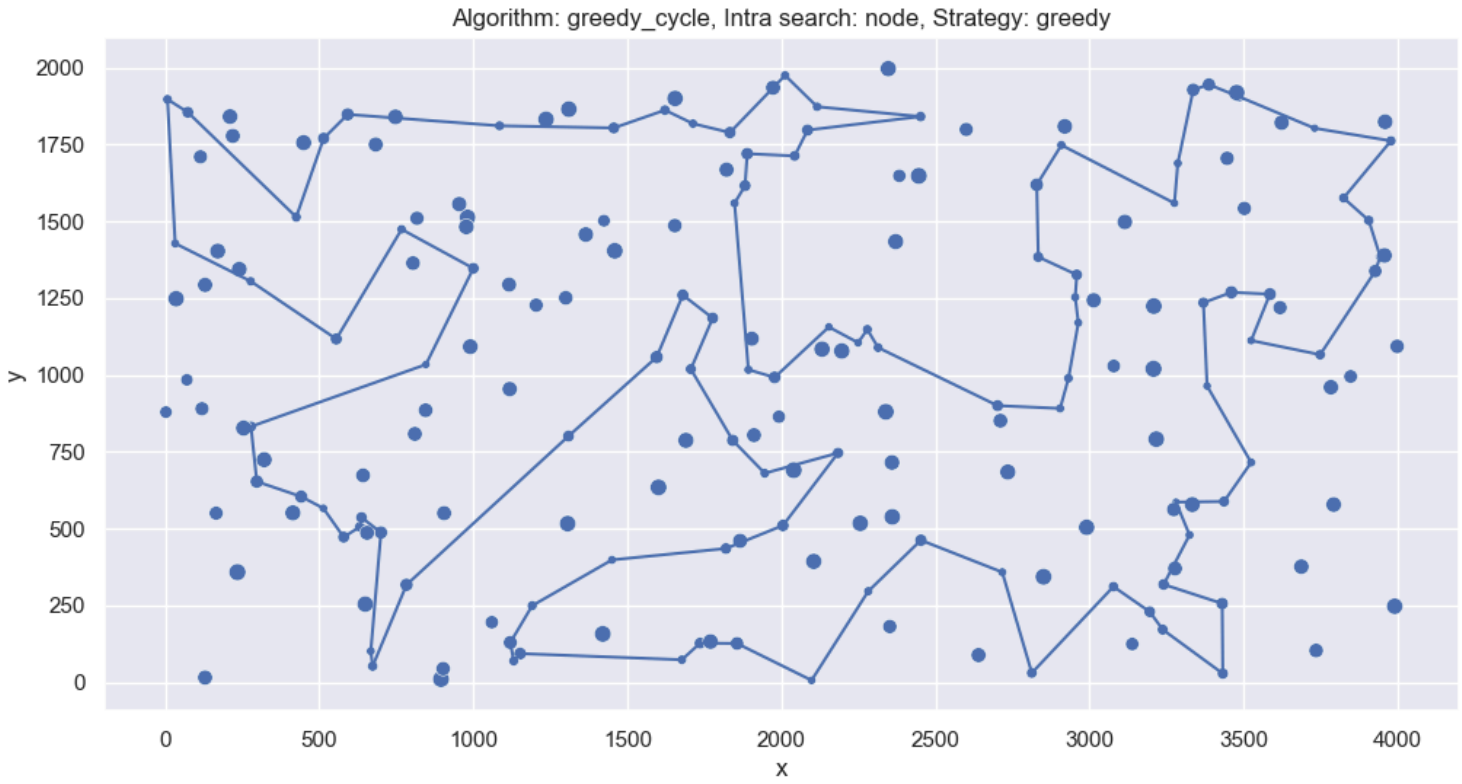
Best solution: [107, 133, 10, 147, 6, 70, 3, 15, 145, 195, 168, 139, 11, 182, 138, 33, 160, 144, 104, 177, 5, 142, 78, 36, 91, 61, 54, 31, 193, 117, 121, 112, 73, 136, 190, 80, 175, 141, 77, 97, 187, 165, 137, 127, 153, 81, 111, 189, 155, 152, 149, 28, 140, 183, 103, 89, 163, 18, 55, 34, 8, 21, 87, 82, 62, 95, 86, 176, 180, 166, 185, 99, 20, 60, 148, 47, 94, 179, 194, 113, 106, 124, 143, 35, 109, 0, 29, 13, 132, 169, 188, 134, 74, 118, 51, 90, 122, 135, 63, 40]

Best solution: [80, 45, 175, 78, 5, 177, 104, 8, 111, 82, 21, 61, 36, 91, 141, 77, 153, 163, 89, 127, 103, 113, 180, 176, 194, 166, 86, 185, 95, 130, 99, 22, 179, 66, 94, 47, 148, 60, 23, 20, 28, 149, 140, 183, 152, 184, 155, 170, 34, 55, 18, 62, 124, 106, 143, 35, 109, 0, 29, 144, 160, 33, 138, 182, 11, 139, 168, 195, 13, 145, 15, 3, 70, 132, 169, 188, 6, 134, 147, 51, 121, 131, 90, 122, 135, 63, 38, 27, 16, 197, 1, 156, 198, 117, 193, 54, 31, 164, 73, 190]

Best solution: [5, 175, 80, 190, 136, 73, 54, 31, 193, 117, 198, 156, 1, 16, 27, 38, 102, 63, 32, 135, 122, 133, 10, 191, 90, 131, 121, 51, 147, 6, 188, 169, 132, 13, 195, 168, 145, 15, 70, 3, 155, 189, 109, 35, 0, 29, 11, 139, 138, 33, 160, 104, 8, 111, 41, 159, 143, 106, 124, 128, 62, 18, 55, 34, 170, 152, 183, 140, 4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 185, 130, 95, 86, 166, 194, 176, 113, 114, 127, 89, 103, 163, 187, 153, 81, 77, 141, 91, 61, 36, 177]

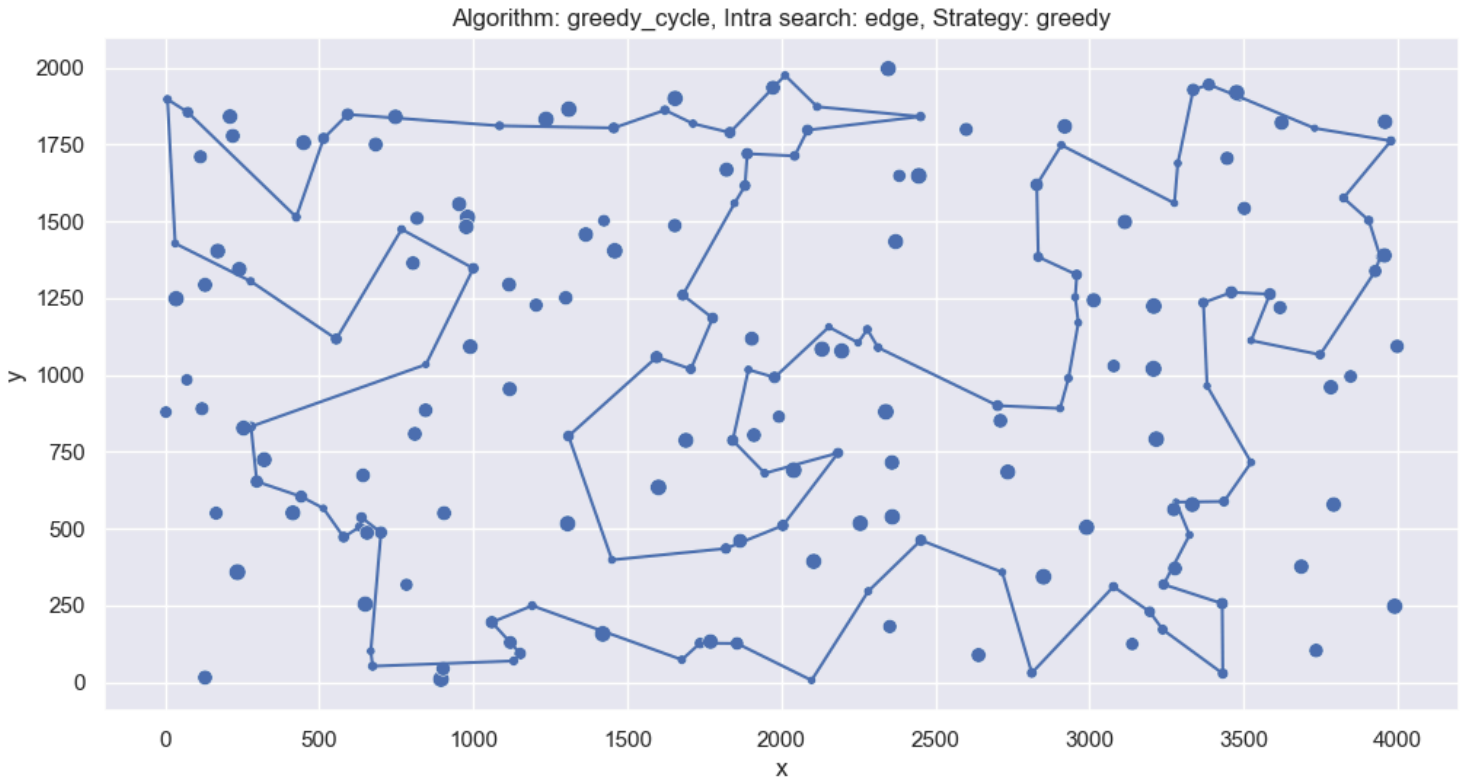


Best solution: [4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 185, 99, 130, 95, 86, 166, 194, 176, 113, 103, 114, 137, 127, 89, 163, 187, 153, 81, 77, 141, 91, 61, 36, 78, 175, 142, 5, 177, 21, 82, 111, 8, 104, 138, 11, 139, 182, 25, 136, 80, 190, 73, 54, 31, 193, 117, 198, 156, 1, 121, 51, 90, 131, 135, 63, 40, 107, 122, 133, 10, 147, 6, 188, 169, 132, 70, 3, 155, 15, 145, 13, 195, 168, 33, 160, 29, 0, 109, 35, 143, 106, 124, 62, 18, 55, 34, 170, 152, 183, 140]

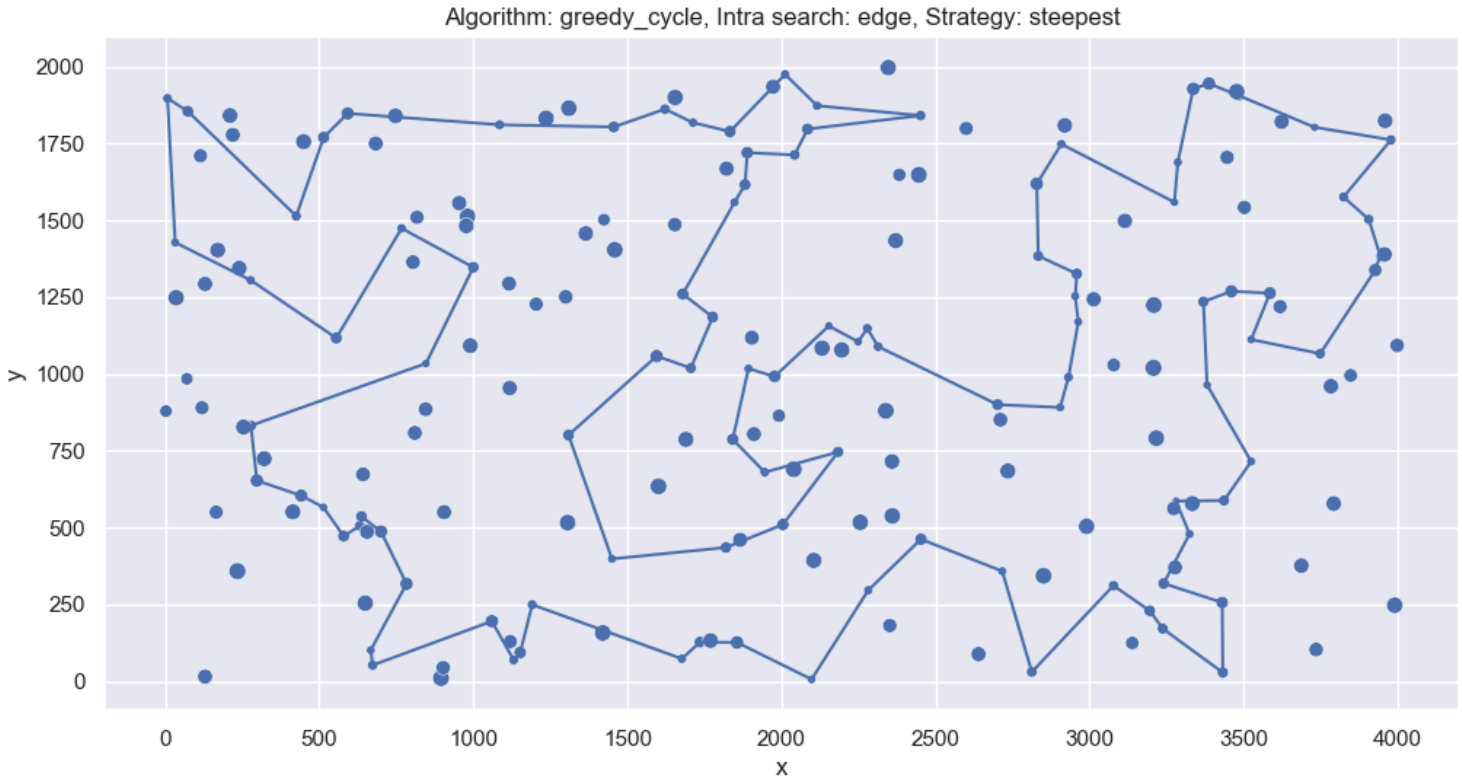


Best solution: [4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 185, 99, 130, 95, 86, 166, 194, 176, 113, 103, 114, 137, 127, 89, 163, 187, 153, 81, 77, 141, 91, 61, 36, 78, 175, 142, 5, 177, 21, 82, 111, 8, 104, 138, 11, 139, 182, 25, 136, 80, 190, 73, 54, 31, 193, 117, 198, 156, 1, 121, 51, 90, 131, 135, 63, 40, 107, 122, 133, 10, 147, 6, 188, 169, 132, 70, 3, 155, 15, 145, 13, 195, 168, 33, 160, 29, 0, 109, 35, 143, 106, 124, 62, 18, 55, 34, 170, 152, 183, 140]

Best solution: [47, 94, 66, 179, 185, 99, 130, 95, 86, 166, 194, 176, 113, 103, 114, 137, 127, 89, 163, 187, 153, 81, 77, 141, 91, 61, 36, 5, 45, 142, 78, 175, 80, 190, 73, 54, 31, 193, 117, 198, 156, 1, 121, 51, 90, 131, 135, 63, 40, 107, 122, 133, 10, 147, 6, 188, 169, 132, 70, 3, 155, 15, 145, 13, 195, 168, 139, 11, 138, 182, 25, 177, 21, 82, 111, 8, 104, 33, 160, 29, 0, 109, 35, 143, 106, 124, 62, 18, 55, 34, 170, 152, 183, 140, 4, 149, 28, 20, 60, 148]



Best solution:[4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 185, 99, 130, 95, 86, 166, 194, 176, 113, 103, 114, 137, 127, 89, 163, 187, 153, 81, 77, 141, 91, 61, 36, 5, 78, 175, 45, 80, 190, 136, 73, 54, 31, 193, 117, 198, 156, 1, 121, 51, 90, 131, 135, 63, 40, 107, 122, 133, 10, 147, 6, 188, 169, 132, 70, 3, 155, 15, 145, 13, 195, 168, 139, 11, 138, 182, 25, 177, 21, 82, 111, 8, 104, 33, 160, 29, 0, 109, 35, 143, 106, 124, 62, 18, 55, 34, 170, 152, 183, 140]



Final tables:

Function performance

Method	Dataset A	Dataset B
Random generation, node, greedy	85999.455(80880-92908)	61198.88(54698-68702)
Greedy cycle, node, greedy	71410.845(70564-72444)	46542.57(45193-47682)
Random generation, edge, greedy	73976.24(71911-77565)	48446.135(45466-52363)
Greedy cycle, edge, greedy	71292.35(70397-72378)	45790.74(44532-47373)
Random generation, node, steepest	88103.225(79540-97655)	62862.19(55813-73074)
Greedy cycle, node, steepest	71408.27(70564-72444)	46537.475(45193-47682)
Random generation, edge, steepest	73855.835(70939-77610)	48296.625(45319-50992)
Greedy cycle, edge, steepest	71259.05(70174-72378)	45564.175(44472-47045)
Greedy cycle	72071.915(71263-73154)	46903.73(45312-48623)
Random generation	265009.32(243393-296391)	212375.76(190669-241913)

Average running time

Method	Dataset A	Dataset B
Random generation, node, greedy	14.11 s	14.9 s
Greedy cycle, node, greedy	0.57 s	0.46 s
Random generation, edge, greedy	12.4 s	12.1 s
Greedy cycle, edge, greedy	0.49 s	0.58 s
Random generation, node, steepest	7.51 s	5.18 s
Greedy cycle, node, steepest	0.51 s	0.43 s
Random generation, edge, steepest	3.3 s	3.17 s
Greedy cycle, edge, steepest	0.46 s	0.56 s
Greedy cycle	0.006 s	0.006 s
Random generation	0.00275 s	0.00275 s

Conclusion:

The Greedy Cycle variants outperformed Random Generation variants on both datasets A and B, getting significantly better solution quality with lower variability in results.

Greedy Cycle with edge-based steepest descent showed the best overall performance, achieving the lowest average costs (71,259 for Dataset A and 45,564 for Dataset B). While Random Generation methods occasionally found good solutions, they showed much higher variability and worse average performance, with costs up to 3-4 times higher than Greedy Cycle approaches.

In terms of computational efficiency, the basic Greedy Cycle and Random Generation methods were the fastest (0.006s and 0.003s respectively), but their solution quality was inferior. The local search variants with Greedy Cycle initialization provided an excellent balance, requiring under 1 second per run while maintaining high solution quality. In contrast, Random Generation with local search required significantly more time (up to 14.9 seconds) while producing worse results.