02-greedy-regret-heuristics

October 27, 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.

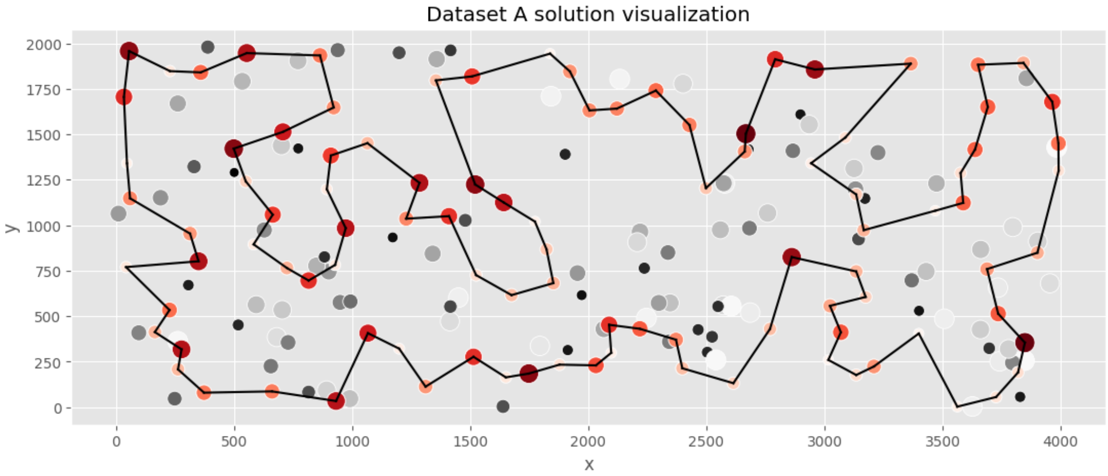
2 Heuristics

2.1 Greedy 2-regret

```
Function Greedy_2_regret_heuristics($dataset, distance\_matrix, start\_node$) :
  size ← determine subset size based on dataset length and a fixed ratio
  num_nodes ← total number of nodes (rows in dataset)
  Copy the distance matrix to avoid modifying the original
  remaining_nodes ← all nodes except the start_node
  solution ← [$start\_node$]
  nearest_node ← find the nearest node to start_node based on distance matrix
  Add nearest_node to solution and remove it from remaining_nodes
  While the solution size is smaller than the subset size :
    best_regret ← $-\infty$
    best_node ← None
    best_insertion ← None
    For each node in remaining_nodes :
      best_cost ← $\infty$
      second_best_cost ← $\infty$
      best_position ← None
      For each position in the current solution :
        Calculate the cost of inserting the node between two positions in solution
        If the current cost < best_cost :
          second_best_cost ← best_cost
          best_cost ← current_cost
          best_position ← current insertion point
        Else If the current cost < second_best_cost :
          second_best_cost ← current_cost
      Calculate the regret ← second_best_cost − best_cost
      If the regret > best_regret :
        best_regret ← regret
        best_node ← current node
        best_insertion ← current insertion point
    Insert best_node into the solution after best_insertion point
    Remove best_node from remaining_nodes
  Returnthe final solution based on dataset order
```

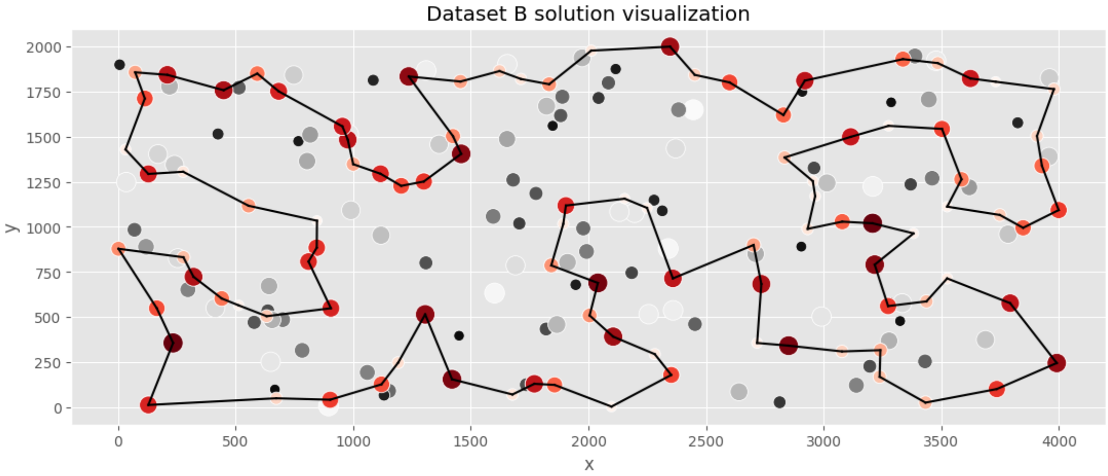
Dataset A

```
Best solution: [196, 157, 188, 113, 171, 16, 78, 25, 44, 120, 82, 129, 92, 57,
172, 2, 75, 86, 26, 121, 182, 53, 158, 154, 6, 135, 194, 127, 123, 24, 156, 4,
190, 177, 104, 54, 48, 34, 192, 181, 146, 22, 20, 134, 18, 69, 67, 140, 68, 110,
142, 41, 96, 42, 43, 77, 65, 197, 115, 198, 46, 60, 118, 109, 151, 133, 79, 80,
176, 66, 141, 0, 153, 183, 89, 23, 186, 114, 15, 148, 9, 61, 73, 132, 21, 14,
49, 178, 52, 185, 119, 165, 39, 95, 7, 164, 71, 27, 90, 81]
Objective function statistics:
minimum = 105692
mean = 115579.335
maximum = 126951
```



Dataset B

```
Best solution: [18, 34, 174, 183, 9, 99, 185, 179, 172, 57, 66, 47, 60, 20, 59,
28, 4, 53, 170, 184, 155, 84, 70, 132, 169, 188, 6, 192, 134, 2, 74, 118, 98,
51, 120, 71, 178, 10, 44, 17, 107, 100, 63, 102, 135, 131, 121, 112, 19, 173,
31, 117, 198, 24, 1, 27, 42, 196, 108, 80, 162, 142, 5, 123, 7, 36, 79, 91, 141,
97, 77, 58, 82, 68, 104, 33, 49, 29, 0, 41, 143, 119, 153, 186, 163, 103, 127,
137, 75, 93, 48, 166, 194, 180, 64, 86, 110, 128, 124, 62]
Objective function statistics:
minimum = 67809
mean = 72740.005
maximum = 78406
```

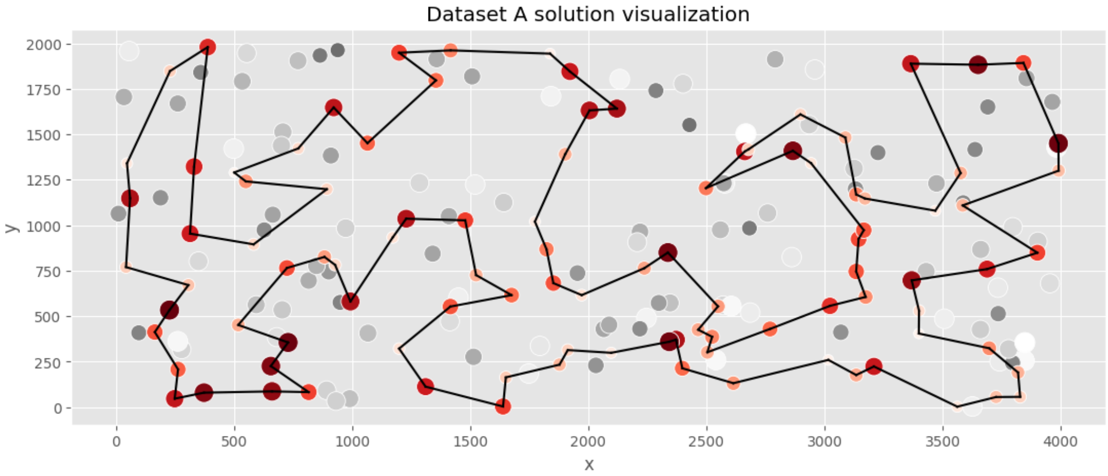

Dataset B solution visualization
2000
1750
1500
1250
1000
750
500
250
0
y
0
500
1000
1500
2000
2500
3000
3500
4000
x

2.2 Greedy heuristics with a weighted sum criterion

```
Function Greedy_heuristics_with_weighted_sum($dataset, distance\_matrix, start\_node, w\_cost, w\_reg$
  size ← determine subset size as half of the dataset length
  num_nodes ← total number of nodes (rows in dataset)
  Copy the distance matrix to avoid modifying the original
  remaining_nodes ← all nodes except the start_node
  solution ← [$start\_node$]
  nearest_node ← find the nearest node to start_node based on distance matrix
  Add nearest_node to solution and remove it from remaining_nodes
  While the solution size is smaller than the subset size :
    best_combined_criterion ← $\infty$
    best_node ← None
    best_insertion ← None
    For each node in remaining_nodes :
      best_cost ← $\infty$
      second_best_cost ← $\infty$
      best_position ← None
      For each position in the current solution :
        Calculate the cost of inserting the node between two positions in solution
        If the current cost < best_cost :
          second_best_cost ← best_cost
          best_cost ← current_cost
          best_position ← current insertion point
        Else If the current cost < second_best_cost :
          second_best_cost ← current_cost
      Calculate the regret ← second_best_cost − best_cost
      combined_criterion ← w_cost × best_cost − w_regret × regret
    If combined_criterion < best_combined_criterion :
      best_combined_criterion ← combined_criterion
      best_node ← current node
      best_insertion ← current insertion point
    Insert best_node into the solution after best_insertion point
    Remove best_node from remaining_nodes
  Returnthe final solution based on dataset order
```

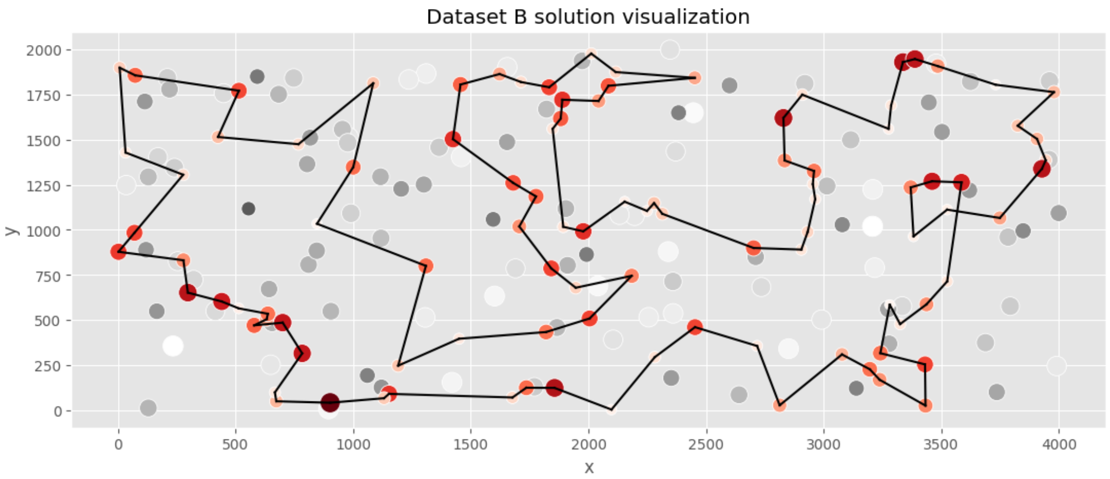
Dataset A

```
Best solution: [0, 117, 143, 183, 89, 186, 23, 137, 176, 80, 79, 63, 94, 124,
152, 97, 1, 101, 2, 129, 92, 57, 55, 52, 49, 102, 148, 9, 62, 144, 14, 178, 106,
185, 165, 21, 7, 164, 27, 90, 40, 81, 196, 179, 145, 78, 31, 113, 175, 171, 16,
25, 44, 120, 75, 86, 26, 100, 53, 180, 154, 135, 70, 127, 123, 162, 133, 151,
51, 118, 59, 149, 65, 116, 43, 184, 35, 84, 112, 4, 190, 10, 177, 54, 48, 160,
34, 146, 22, 18, 108, 159, 181, 42, 115, 41, 193, 139, 68, 46]
Objective function statistics:
minimum = 71057
mean = 72218.32
maximum = 73587
```



Dataset B

```
Best solution: [4, 149, 28, 20, 60, 148, 47, 94, 66, 179, 185, 86, 95, 130, 99,
166, 194, 113, 176, 103, 114, 137, 127, 89, 163, 187, 153, 81, 77, 141, 91, 61,
36, 78, 175, 162, 80, 190, 136, 73, 193, 31, 54, 117, 198, 156, 1, 27, 38, 135,
63, 40, 107, 133, 122, 90, 147, 51, 121, 25, 5, 177, 21, 82, 111, 8, 104, 138,
11, 139, 134, 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]
Objective function statistics:
minimum = 45453
mean = 46252.105
maximum = 47884
```



3 Joint Results

Dataset A

	min	mean	max
greedy_cycle	71263.0	72071.915	73154.0
greedy_weighted	71057.0	72218.320	73587.0
nn_best	71114.0	72871.870	74875.0
nn_last	83182.0	85108.510	89433.0
greedy_2-regret	105692.0	115579.335	126951.0
random_solution	236601.0	262859.735	297066.0

Dataset B

	min	mean	max
greedy_weighted	45453.0	46252.105	47884.0
greedy_cycle	45312.0	46903.730	48623.0
nn_best	44762.0	47575.555	49919.0
nn_last	52319.0	54390.430	59030.0
greedy_2-regret	67809.0	72740.005	78406.0
random_solution	187699.0	212675.575	244471.0

4 Conclusion

Pure regret didn't help to improve the score, while the weighted version worked situatively: on one dataset it improved mean, but got worse min solution and vv on the other.