Michał Kałmucki 151944 Dominik Ludwiczak 151948 (All solutions were checked with solution checker)

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.

Methods to solve:

Greedy 2-regret

Pseudo code:

```
FUNCTION FindRegret2(solution, objective):
  SET initial variables for minimum cost, adding node, and best regret

  FOR each node not already in the solution:
      SET variables for best cost, second-best cost, and best position

      FOR each possible position in the solution:
          CALCULATE cost of inserting the node

          UPDATE best and second-best costs for the node

      IF regret (calculated as secondBestCost - bestCost) is better than the
current best regret:
          UPDATE best regret with the node's best and second-best costs, and its
position
          SET this node as the adding node

  INSERT adding node into the solution at the best position
  UPDATE the objective with the best cost
  MARK the node as taken
  RETURN updated solution
```

```
FUNCTION findFirst2(solution, objective):
  SET initial variables for minimum cost and adding node

  FOR each node not already in the solution:
```

```
        CALCULATE cost of inserting the node at the beginning of the solution

        IF the cost is lower than the current minimum:
            UPDATE minimum cost and selected node

    ADD the best node to the solution
    UPDATE the solution's cost and mark the node as used
    RETURN updated solution
```

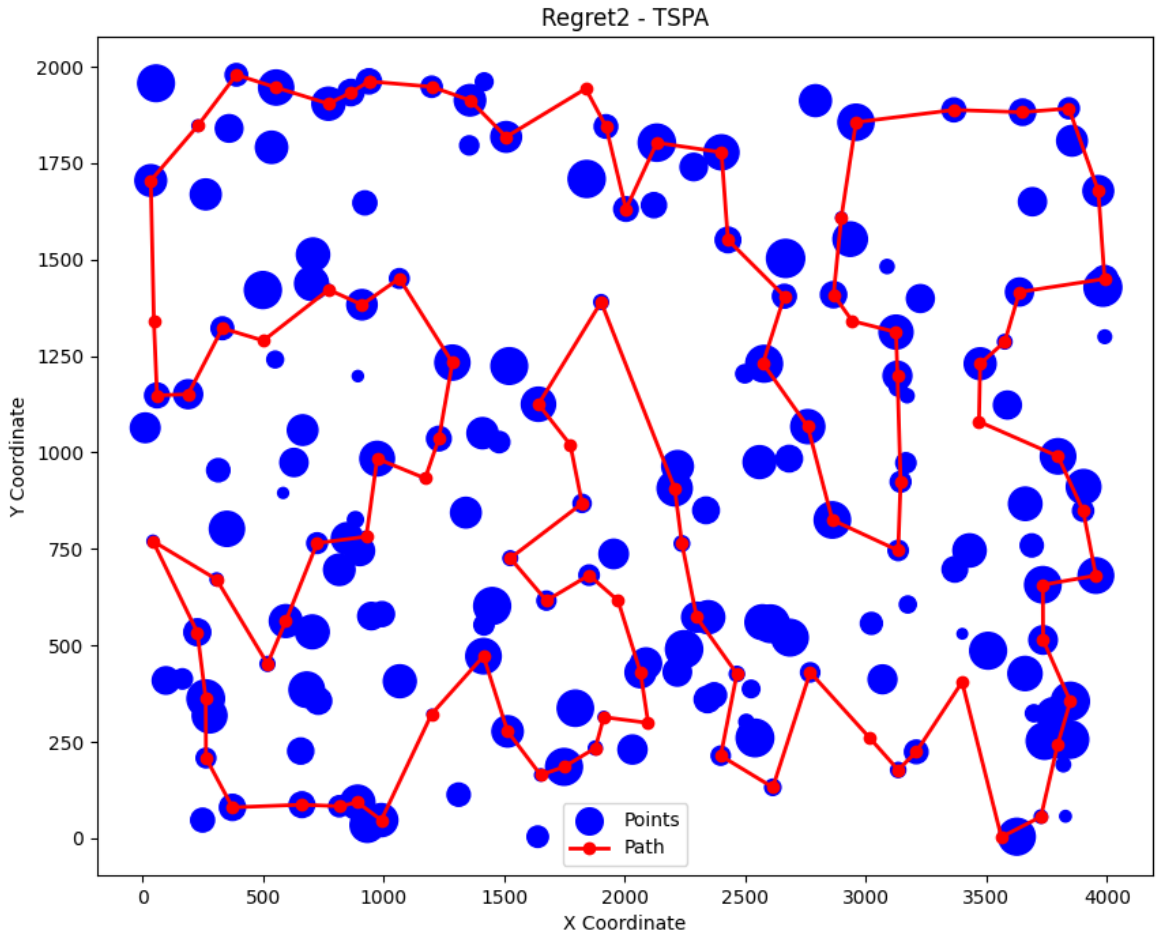
Results:

Problem Instance	Min Distance	Max Distance	Average Distance
TSPA	105,852	123,171	115,630.16
TSPB	67,568	77,329	72,656.19

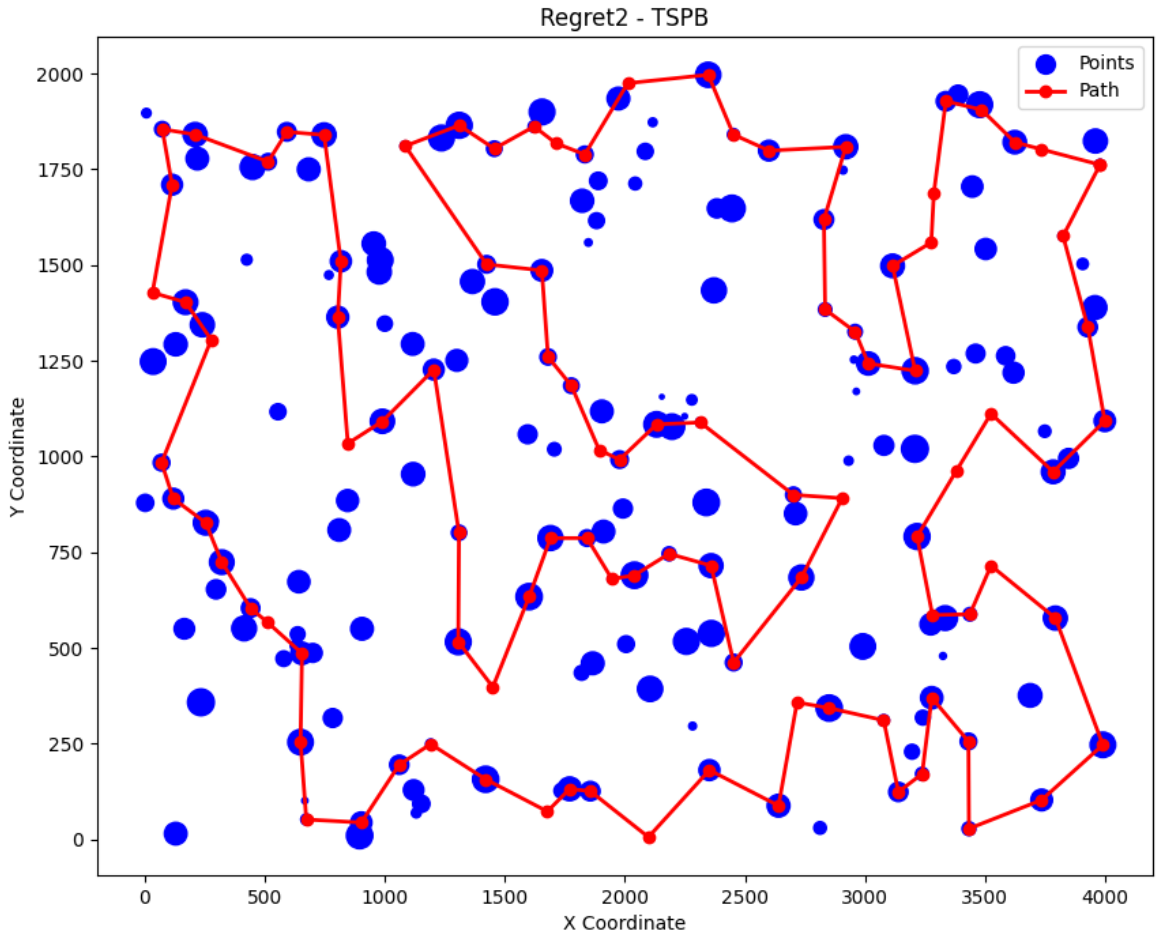
Best Solutions:

- TSPA: 93, 117, 170, 153, 183, 89, 23, 83, 64, 15, 9, 37, 128, 172, 57, 55, 3, 32, 49, 102, 144, 132, 21, 7, 164, 71, 27, 39, 165, 8, 185, 174, 81, 98, 17, 157, 188, 56, 171, 16, 78, 25, 44, 120, 2, 75, 86, 97, 189, 94, 130, 137, 66, 176, 80, 151, 133, 79, 63, 136, 53, 180, 154, 6, 135, 194, 161, 123, 29, 126, 112, 4, 190, 177, 147, 48, 34, 160, 184, 28, 43, 65, 197, 59, 118, 60, 46, 198, 139, 193, 159, 195, 146, 22, 20, 18, 108, 67, 36, 140
- TSPB: 60, 20, 59, 28, 4, 140, 183, 174, 181, 83, 55, 34, 170, 53, 184, 155, 84, 70, 132, 169, 188, 6, 150, 147, 134, 43, 139, 11, 33, 160, 39, 35, 143, 106, 119, 81, 41, 111, 68, 8, 104, 157, 171, 177, 123, 25, 118, 116, 121, 125, 191, 115, 10, 133, 17, 107, 100, 63, 96, 135, 38, 16, 197, 24, 198, 117, 164, 105, 80, 162, 45, 5, 7, 36, 79, 91, 141, 97, 146, 153, 186, 163, 165, 127, 26, 114, 137, 75, 93, 48, 166, 194, 176, 64, 86, 185, 52, 57, 66, 148

Instance A visualization:



Instance B visualization:



Greedy heuristics with a weighted sum criterion: 2-regret + best change of the objective function

Pseudo code:

```
FUNCTION FindRegret2(solution, objective):
  SET initial variables for minimum cost, adding node, and best regret

  FOR each node not already in the solution:
      SET variables for best cost, second-best cost, and best position

      FOR each possible position in the solution:
          CALCULATE cost of inserting the node

          UPDATE best and second-best costs for the node

      IF regret cost ratio (calculated as (secondBestCost - bestCost) * 0.5 -
bestCost * 0.5)
      is better than the current best regret:
          UPDATE best regret with the node's best and second-best costs, and its
position
          SET this node as the adding node
```

```
    INSERT adding node into the solution at the best position
    UPDATE the objective with the best cost
    MARK the node as taken
    RETURN updated solution
```

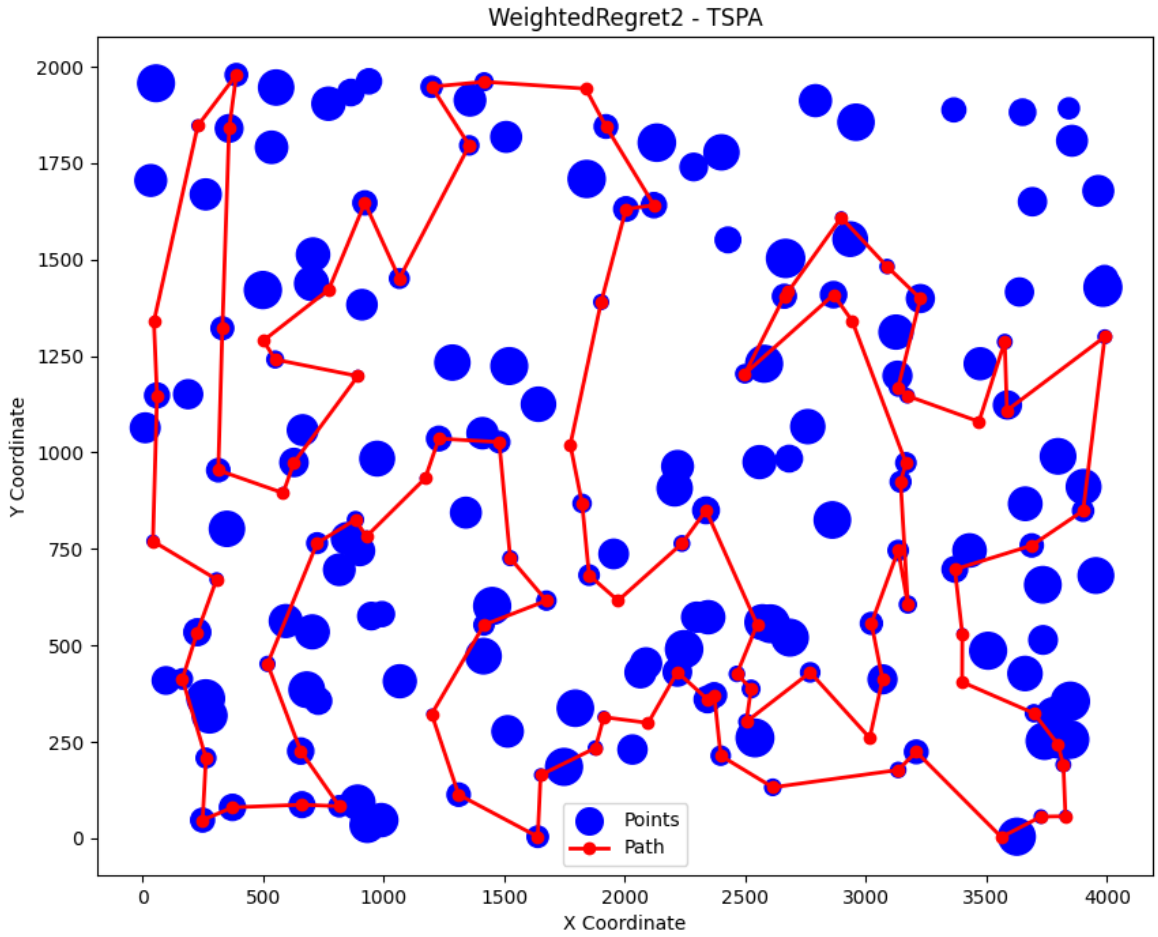
Results:

Type	Min Distance	Max Distance	Average Distance
TSPA	71,108	73,395	72,130.045
TSPB	47,144	55,700	50,919.565

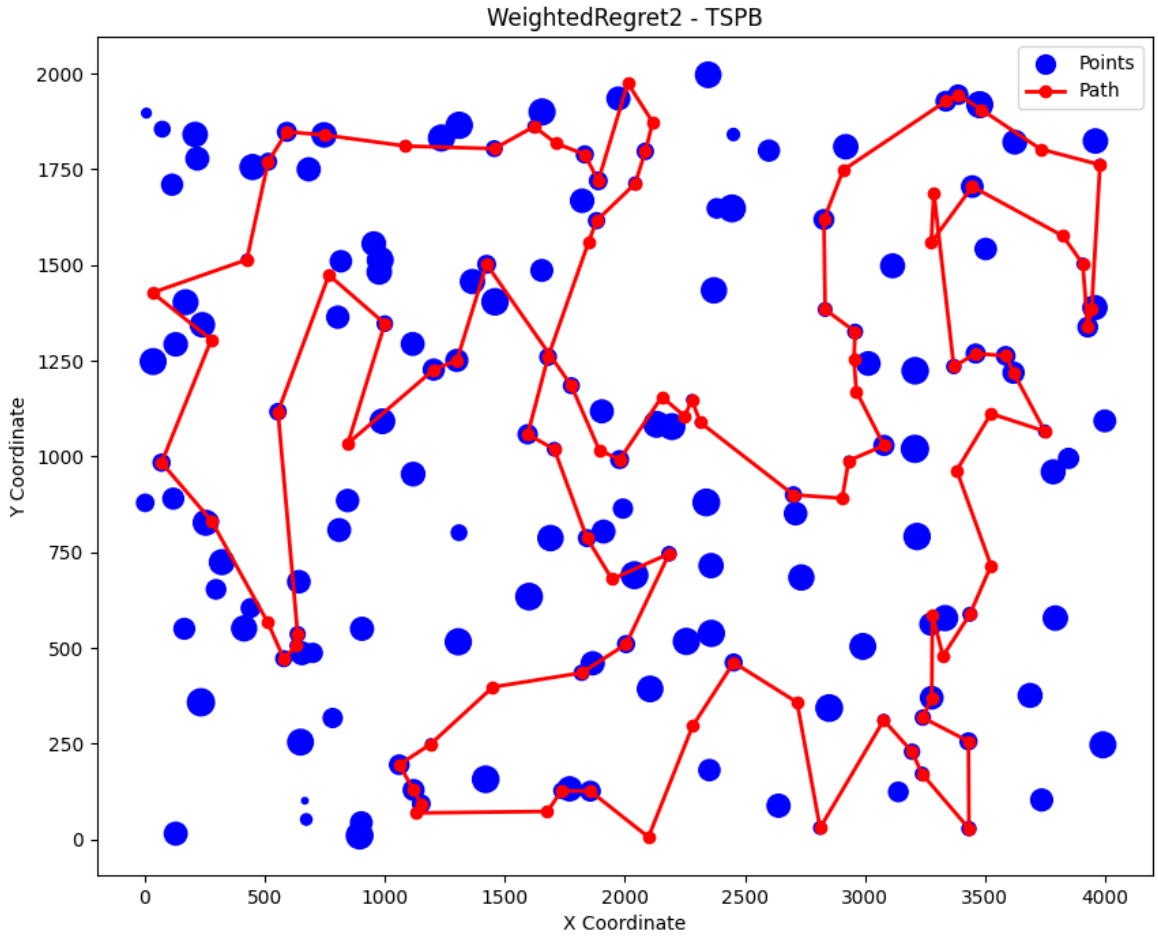
Best Solutions:

- TSPA: 0, 46, 68, 139, 193, 41, 115, 5, 42, 181, 159, 69, 108, 18, 22, 146, 34, 160, 48, 54, 177, 10, 190, 4, 112, 84, 184, 43, 116, 65, 59, 118, 51, 151, 133, 162, 123, 127, 70, 135, 154, 180, 53, 121, 100, 26, 86, 75, 44, 25, 16, 171, 175, 113, 56, 31, 78, 145, 179, 196, 81, 90, 40, 165, 185, 106, 178, 138, 14, 144, 62, 9, 148, 102, 49, 52, 55, 92, 57, 129, 82, 120, 2, 101, 1, 97, 152, 124, 94, 63, 79, 80, 176, 137, 23, 186, 89, 183, 143, 117
- TSPB: 199, 148, 47, 66, 94, 60, 20, 28, 149, 4, 152, 170, 34, 55, 18, 62, 128, 124, 106, 143, 35, 109, 0, 29, 160, 33, 11, 134, 74, 118, 121, 51, 90, 131, 54, 31, 193, 117, 1, 38, 135, 63, 122, 133, 10, 115, 147, 6, 188, 169, 132, 13, 70, 3, 15, 145, 195, 168, 139, 182, 138, 104, 8, 111, 82, 21, 177, 5, 45, 142, 78, 175, 36, 61, 91, 141, 77, 81, 153, 187, 163, 89, 127, 137, 114, 103, 26, 176, 113, 194, 166, 86, 185, 179, 22, 99, 130, 95, 140, 183

Instance A visualization:



Instance B visualization:



Source code link: Github

Summary

Comparison of all methods

Method	Instance	Min Distance	Max Distance	Average Distance
RANDOM	TSPA	236,372	291,536	264,513.735
	TSPB	194,771	252,218	214,996.51
NEARESTNEIGHBOR	TSPA	83,182	89,433	85,108.51
	TSPB	52,319	59,030	54,390.43
ANYNEARESTNEIGHBOR	TSPA	71,179	75,450	73,178.435
	TSPB	44,417	53,438	45,870.255
GREEDYCYCLE	TSPA	71,488	74,924	72,775.67
	TSPB	49,001	57,294	51,509.075
GREEDY 2-REGRET	TSPA	105,852	123,171	115,630.16

Method	Instance	Min Distance	Max Distance	Average Distance
	TSPB	67,568	77,329	72,656.19
GREEDY 2-REGRET WEIGHTED	TSPA	71,108	73,395	72,130.045
	TSPB	47,144	55,700	50,919.565

Conclusion

The Greedy 2-Regret algorithm showed not bad performance by minimizing regret across instances, achieving efficient results by focusing on balancing short-term and long-term costs. Its ability to consider the second-best insertion option allows for more flexible decision-making and reduced overall costs compared to simpler methods.

The Greedy 2-Regret Weighted algorithm performed even better, thanks to its emphasis on avoiding costly insertions through a weighted regret calculation. By prioritizing nodes with higher insertion costs, it consistently achieved lower overall costs, making it the most effective approach in this comparison.

Weighted greedy 2-regret approach demonstrated a bit improvement over basic methods, while normal greedy 2-regret performed werse than methods considered in previous assignment. Greedy 2-Regret Weighted standing out as the superior option for tasks involving cost-sensitive decision-making.