Assignment 2 - Greedy regret heuristics

Authors

- Michał Kamiński 151969
- Jan Indrzejczak 152059

Table of contents

Description of the problem

The travelling salesman problem (TSP) is a classic optimization problem. Given a list of cities and the distances between them, the task is to find the shortest possible route that visits each city exactly once and returns to the origin city. In this version of the problem, each city also has a cost of being visited, and we only need to select half of the cities.

As an input we received a list of coordinates of cities, along with the cost. To calculate the distance between cities we used Euclidean distance, and each city is represented as a number from 0 to n-1 (n-number of cities). The objective function is to find the route that minimizes the sum of distances between cities and the cost of visiting them.

Pseudocode of all implemented algorithms

Greedy 2-regret heuristic

```
Initialize

    Get the size of the problem.
    Compute the half size of the solution (round up).
    Initialize an empty solution list.
    Initialize a visited list with all nodes marked as unvisited.
    Select the starting node (either provided or default to node 0).
    Mark the starting node as visited and add it to the solution.
    Initialize the current cost with the cost of the starting node.

Main Loop
Repeat until the solution contains half of the nodes:

    For each unvisited node:
        Initialize best and second-best insertion costs for the node.
        For each possible insertion position in the solution:
            Compute the total cost of inserting the node at this position.
            If inserting at position 0 or the last position, update the cost by considering the edges connecting the start and end of the solution to the new node.
            If inserting at an intermediate position, update the cost by replacing the edges between two consecutive nodes with edges to the new node.
        Update the best and second-best insertion costs based on the computed total cost.
        Calculate regret as the difference between the second-best and best costs.
        Track the node and position that provide the highest regret.

Add Node with Maximum Regret

    Insert the node with the highest regret at its best position in the solution.
    Update the total cost and mark the node as visited.

End Loop

Return the solution.
```

Greedy weighted 2-regret heuristic

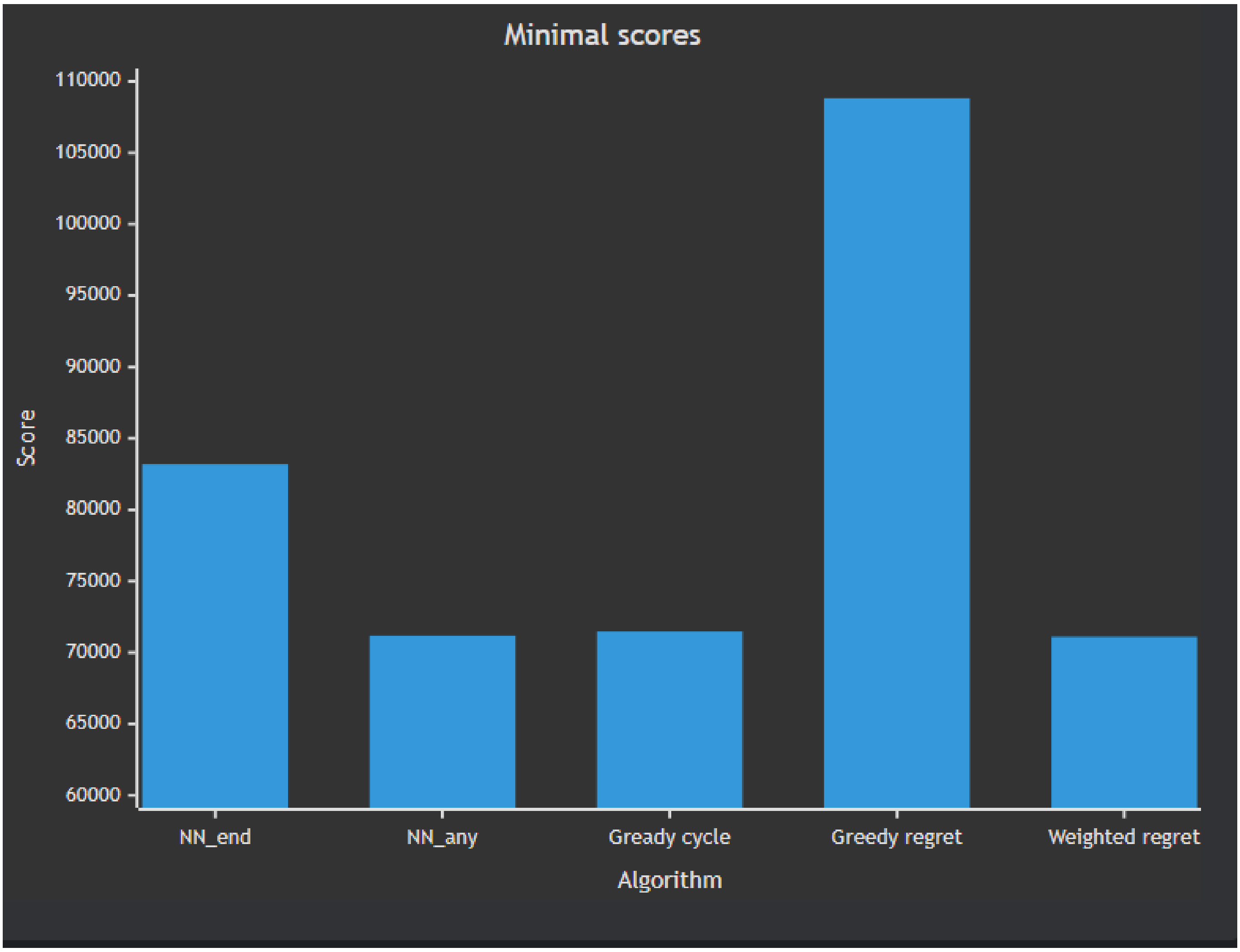
```
Initialize

    Get the size of the problem.
    Compute the half size of the solution (round up).
    Initialize an empty solution list.
    Initialize a visited list with all nodes marked as unvisited.
    Select the starting node (either provided or default to node 0).
    Mark the starting node as visited and add it to the solution.
    Initialize the current cost with the cost of the starting node.

Main Loop
Repeat until the solution contains half of the nodes:

    For each unvisited node:
        Initialize best and second-best insertion costs for the node.
        For each possible insertion position in the solution:
            Compute the total cost of inserting the node at this position.
            If inserting at position 0 or the last position, update the cost by considering the edges connecting the start and end of the solution to the new node.
            If inserting at an intermediate position, update the cost by replacing the edges between two consecutive nodes with edges to the new node.
        Update the best and second-best insertion costs based on the computed total cost.
        Calculate regret as the difference between the second-best and best costs.
        Compute a weighted score using a formula that combines regret and the difference between the best cost and the current cost. Use a predefined constant REGRET_WEIGHT to adjust the importance of regret versus cost improvement:
            score=(regret×REGRET_WEIGHT)−((best cost−current cost)×(1−REGRET_WEIGHT))
        Track the node and position that provide the highest score.

Add Node with Maximum Score

    Insert the node with the highest score at its best position in the solution.
    Update the total cost and mark the node as visited.
```

```
End Loop

Return the solution.
```

Results of computational experiments

TSPA

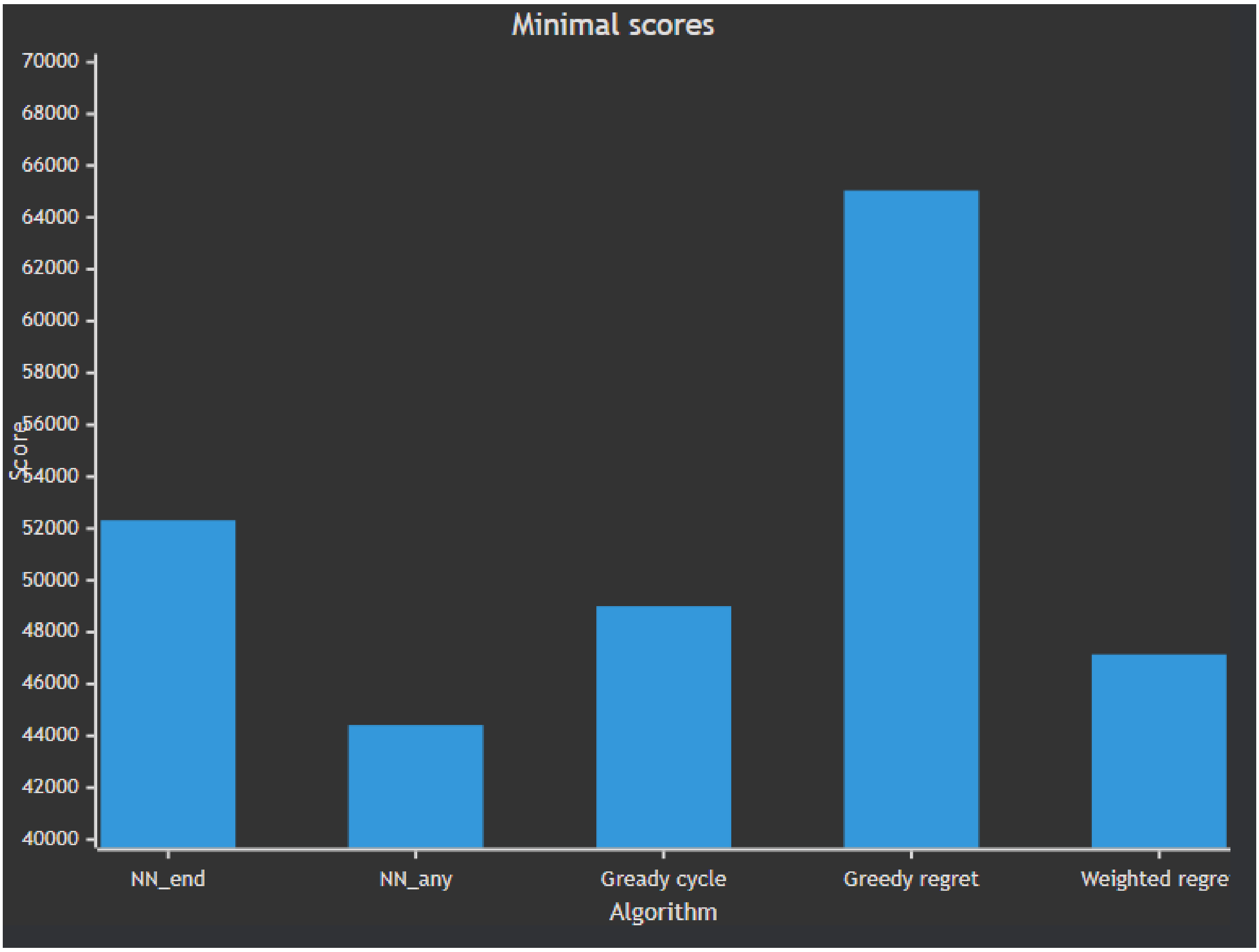
```
Results for Random Algorithm
Min cost: 225467

Results for Nearest Neighbor with adding the node at the end algorithm
Min cost: 83182

Results for Nearest neighbor insert anywhere algorithm
Min cost: 71179

Results for Greedy cycle algorithm
Min cost: 71488
Max cost: 74410
Average cost: 72636

Results for Greedy Regret Heuristic with 2-Regret
Min cost: 108804
Max cost: 123447
Average cost: 116681

Results for Greedy Regret Heuristic with weighted 2-Regret
Min cost: 71108
Max cost: 73718
Average cost: 72148
```



TSPB

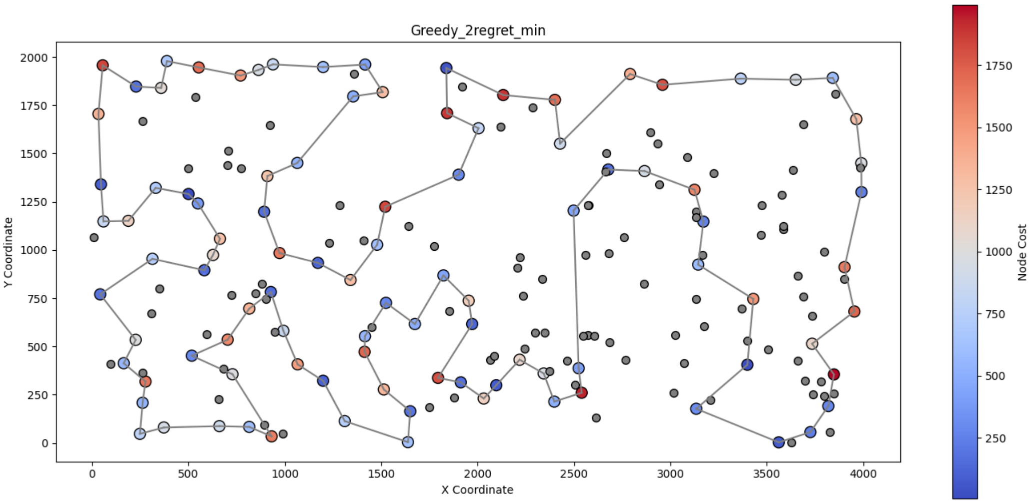
```
Results for Random Algorithm
Min cost: 193417

Results for Nearest Neighbor with adding the node at the end algorithm
Min cost: 52319

Results for Nearest neighbor insert anywhere algorithm
Min cost: 44417

Results for Greedy cycle algorithm
Min cost: 49001
Max cost: 57324
Average cost: 51401

Results for Greedy Regret Heuristic with 2-Regret
Min cost: 65043
Max cost: 76325
Average cost: 70265

Results for Greedy Regret Heuristic with weighted 2-Regret
Min cost: 47144
```

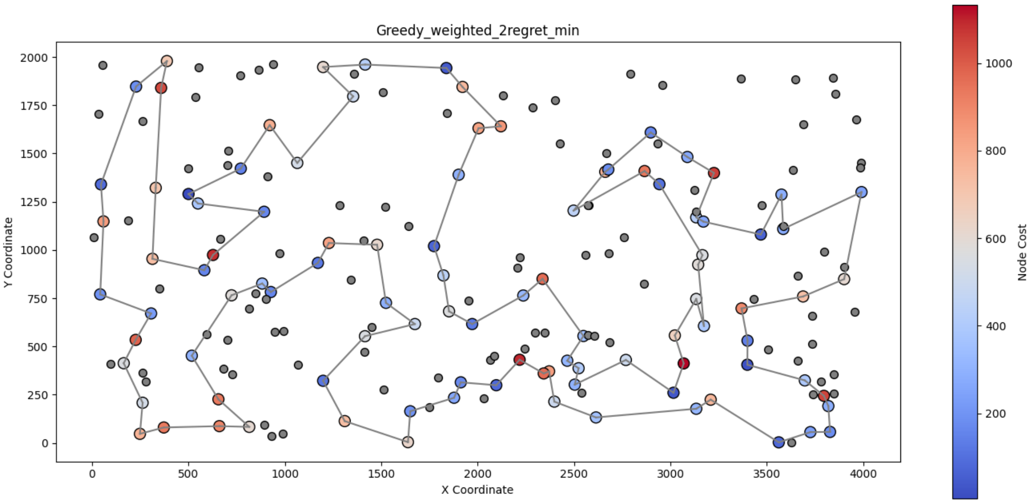
```
Max cost: 56747
Average cost: 50997
```

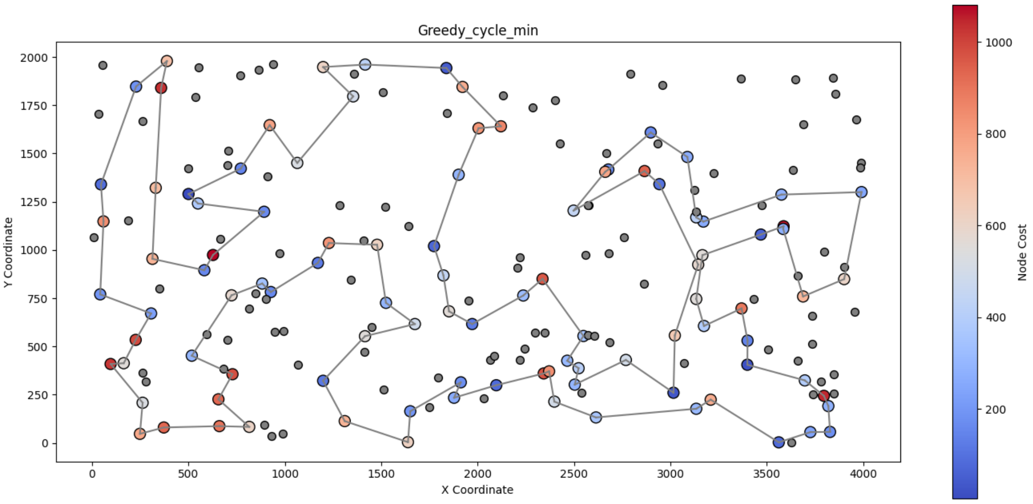


Plots of the results

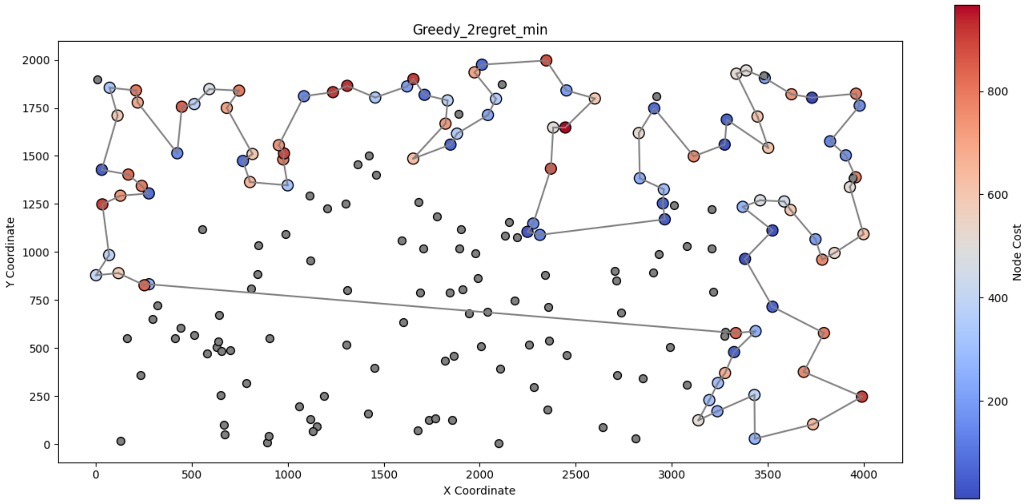
TSPA

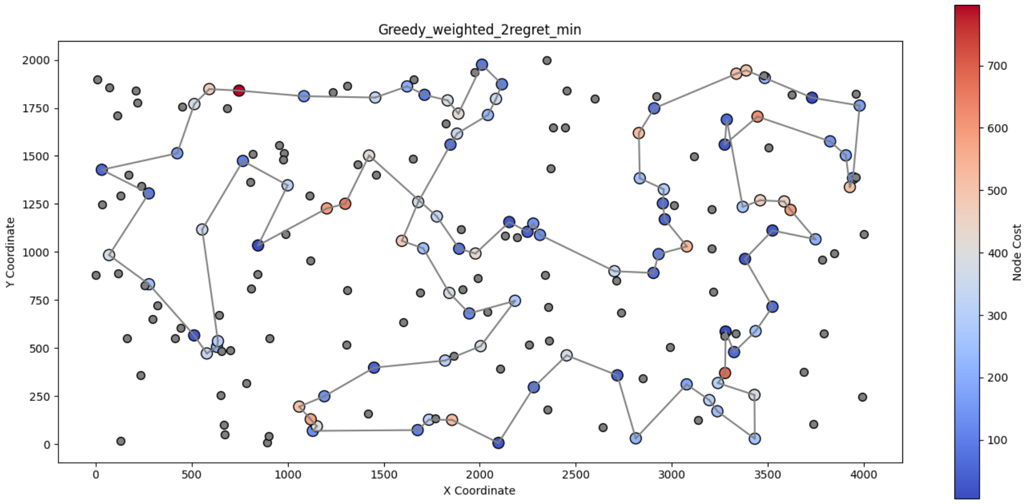

Greedy_2regret_min
Y Coordinate
X Coordinate
Node Cost
0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500
2000
2500
3000
3500
4000
250
500
750
1000
1250
1500
1750

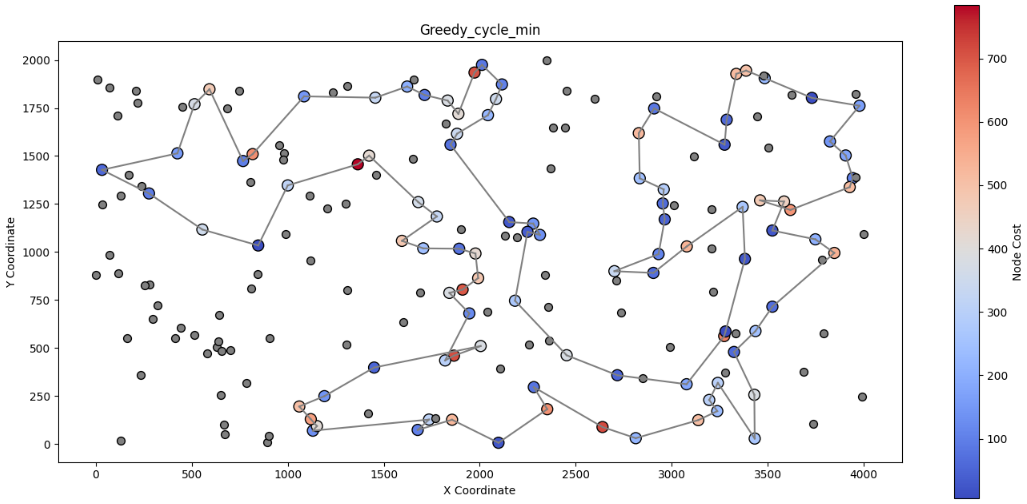

Greedy_weighted_2regret_min
Y Coordinate
X Coordinate
Node Cost
0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500
2000
2500
3000
3500
4000
200
400
600
800
1000


Greedy_cycle_min
Y Coordinate
X Coordinate
Node Cost
0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500
2000
2500
3000
3500
4000
200
400
600
800
1000

TSPB







Best solutions as a list of nodes

TSPA

	Greedy 2-regret heuristic	Greedy cycle	Greedy weighted 2-regret heuristic
0	1	46	137
1	150	68	176
2	86	139	80
3	100	193	79
4	121	41	63
5	53	115	94
6	158	5	124
7	180	42	152
8	173	181	97
9	63	159	1
10	122	69	101
11	80	108	2
12	133	18	120
13	151	22	82
14	162	146	129
15	161	34	57
16	194	160	92
17	135	48	55
18	70	54	52
19	127	30	49
20	123	177	102
21	24	10	148
22	149	190	9
23	65	4	62
24	77	112	144
25	166	84	14
26	184	35	138
27	35	184	178
28	156	43	106

	Greedy 2-regret heuristic	Greedy cycle	Greedy weighted 2-regret heuristic
29	112	116	185
30	4	65	165
31	190	59	40
32	10	118	90
33	177	51	81
34	104	151	196
35	54	133	179
36	48	162	145
37	34	123	78
38	181	127	31
39	42	70	56
40	5	135	113
41	96	180	175
42	41	154	171
43	193	53	16
44	159	100	25
45	195	26	44
46	146	86	75
47	22	75	86
48	20	44	26
49	134	25	100
50	18	16	121
51	69	171	53
52	108	175	180
53	67	113	154
54	36	56	135
55	140	31	70
56	93	78	127
57	117	145	123
58	143	179	162

	Greedy 2-regret heuristic	Greedy cycle	Greedy weighted 2-regret heuristic
59	153	92	133
60	0	57	151
61	46	52	51
62	198	185	118
63	115	119	59
64	197	40	65
65	59	196	116
66	72	81	43
67	51	90	184
68	141	165	84
69	137	106	112
70	23	178	4
71	76	14	190
72	183	144	10
73	83	62	177
74	64	9	54
75	15	148	48
76	73	102	160
77	132	49	34
78	21	55	146
79	7	129	22
80	164	120	18
81	71	2	108
82	27	101	69
83	90	1	159
84	187	97	181
85	98	152	42
86	157	124	5
87	188	94	115
88	113	63	41

	Greedy 2-regret heuristic	Greedy cycle	Greedy weighted 2-regret heuristic
89	171	79	193
90	16	80	139
91	44	176	68
92	78	137	46
93	91	23	0
94	55	186	117
95	106	89	143
96	32	183	183
97	102	143	89
98	62	117	186
99	148	0	23

TSPB

	Greedy 2-regret heuristic	Greedy cycle	Greedy weighted 2-regret heuristic
0	1	51	183
1	197	121	140
2	16	131	95
3	27	135	130
4	38	63	99
5	92	122	22
6	102	133	179
7	135	10	185
8	32	90	86
9	96	191	166
10	63	147	194
11	100	6	113
12	107	188	176
13	17	169	26
14	72	132	103
15	122	13	114

	Greedy 2-regret heuristic	Greedy cycle	Greedy weighted 2-regret heuristic
16	44	161	137
17	133	70	127
18	10	3	89
19	115	15	163
20	178	145	187
21	191	195	153
22	90	168	81
23	125	29	77
24	51	109	141
25	120	35	91
26	67	0	61
27	71	111	36
28	147	81	175
29	192	153	78
30	150	163	142
31	6	180	45
32	188	176	5
33	65	86	177
34	169	95	21
35	132	128	82
36	126	106	111
37	43	143	8
38	168	124	104
39	195	62	138
40	145	18	182
41	15	55	139
42	161	34	168
43	70	170	195
44	84	152	145
45	155	183	15

	Greedy 2-regret heuristic	Greedy cycle	Greedy weighted 2-regret heuristic
46	184	140	3
47	167	4	70
48	189	149	13
49	69	28	132
50	109	20	169
51	0	60	188
52	35	148	6
53	62	47	147
54	18	94	115
55	55	66	10
56	34	22	133
57	170	130	122
58	152	99	63
59	174	185	135
60	183	179	38
61	140	172	1
62	9	166	117
63	199	194	193
64	4	113	31
65	149	114	54
66	28	137	131
67	59	103	90
68	20	89	51
69	23	127	121
70	60	165	118
71	148	187	74
72	47	146	134
73	154	77	11
74	66	97	33
75	57	141	160

	Greedy 2-regret heuristic	Greedy cycle	Greedy weighted 2-regret heuristic
76	172	91	29
77	52	36	0
78	179	61	109
79	22	175	35
80	99	78	143
81	130	142	106
82	95	45	124
83	185	5	128
84	86	177	62
85	166	82	18
86	48	87	55
87	76	21	34
88	93	8	170
89	75	104	152
90	137	56	4
91	114	144	149
92	127	160	28
93	165	33	20
94	89	138	60
95	103	182	94
96	26	11	66
97	113	139	47
98	194	134	148
99	88	85	199

Source code

- Github repository

Conclusions

The best solutions have been checked with the solution checker.

Using a greedy 2-regret heuristic can lead to unambiguous results. We believe that this is due to the fact that this algorithm only looks at regret as a criterion for selecting next node. It can lead to a very bad choice, only because two best options for this node are very close to each other. This is partially solved by using the weighted 2-regret method. In our experiments, the best results were achieved when the weights were equal. Taking into consideration both the regret and the change in the cost of the objective function, mitigates the previous downside. After our experiments, we would probably run greedy cycle and greedy weighted 2-regret and choose the best result, since they seem to be very close to each other.