Estimating TSP with Self organizing maps

Parsa Tasbihgou
Prof Babaali
February 5, 2022

I have used the SOM clustering method to break an instance of TSP into smaller subproblems and solving each subproblem with brute force method, then joining the solved subproblems together to make an approximate solution to the original problem.

One reason that SOM is specially suitable for this application is that not only it divides the nodes into clusters but unlike other clustering algorithms by applying a proper neighborhood map, the clusters will form a decant ordering among them selves, therefor joining the subproblems is almost trivial.

In this implementation I have used a circular neighborhood (linear with the ends tied together) so that the an order is formed on the clusters.
The algorithm starts with the radius of neighborhood set to 3 and gradually decreases it until it reaches 0.

The number of candidate clusters is set to N (number of the cities) and for larger instances set to N/2, since the training process of the network is has computational complexity of O (N * M) where M is the number of candidate clusters for each iteration, so if M = N, O(N^2).
So I expect that most non-trivial clusters have 1 to 3 and in rare cases 4 to 5 nodes.I have used exhaustive backtrack search to solve each subproblem, but I do not expect that any subproblem will be larger than 4 or 5 nodes which is negligible.

The ordering of the clusters generated by the training process has shown to be acceptable hence no further optimization is applied to it. For joining the subproblems we start from the first non-empty cluster and move forward, if the cluster has more than 1 node, the “sub” function finds the optimal ordering of this cluster and we greedily attach on end of the found path to the last node of the previous cluster. The greedy method compares which endpoint is closest to the previous node. A possible optimization is to include information from the last cluster in calculating the best ordering of each cluster, however since the

subproblems are relatively small it is unlikely that much improvement is achieved by this method.

Another possible optimization that based on the results, I believe is more effective than the previous one, is to use a more accurate method to join the subproblems, however it appears that solving this problem optimally is quit difficult and I have not tried any suboptimal methods. Also the current method is achieving acceptable results, however it is not much more accurate than joining the randomly.

Below is an example of the type of output that is generated by this program and the best answers that I have found.

1.tsp: 29 [27, 5, 11, 8, 4, 25, 2, 28, 1, 20, 0, 23, 15, 12, 9, 19, 3, 18, 14, 17, 13, 16, 21, 10, 24, 6, 22, 26, 7]

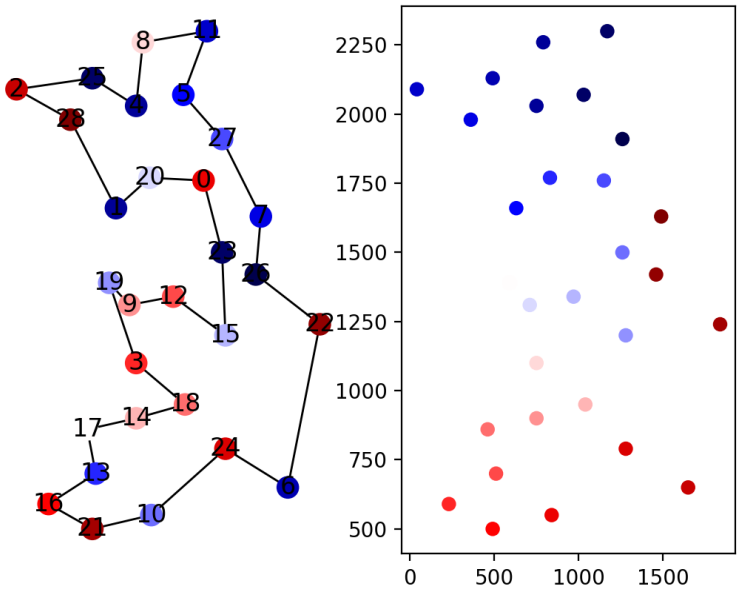
Cost: 9383.362662506262

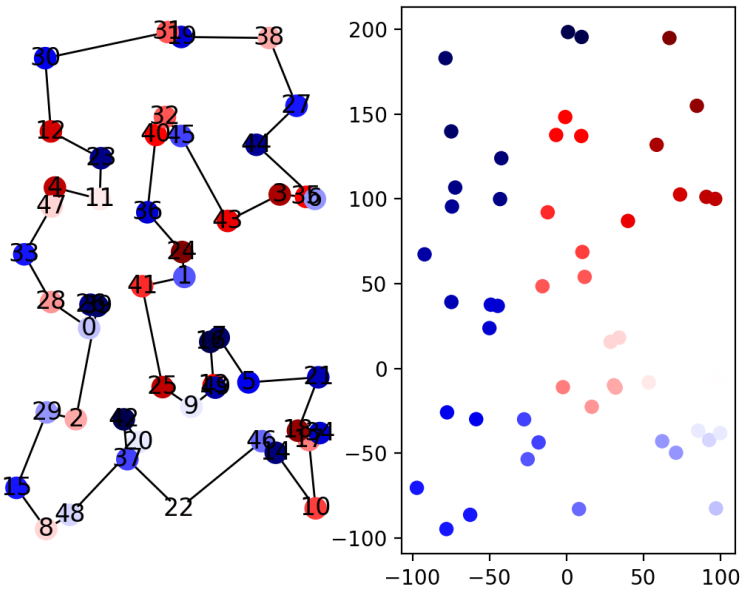
2.tsp: 50 [19, 31, 30, 12, 23, 11, 4, 47, 33, 28, 0, 26, 39, 2, 29, 15, 8, 48, 20, 42, 37, 22, 46, 14, 10, 17, 34, 18, 21, 5, 7, 16, 13, 49, 9, 25, 41, 1, 24, 36, 40, 32, 45, 43, 3, 35, 6, 44, 27, 38]

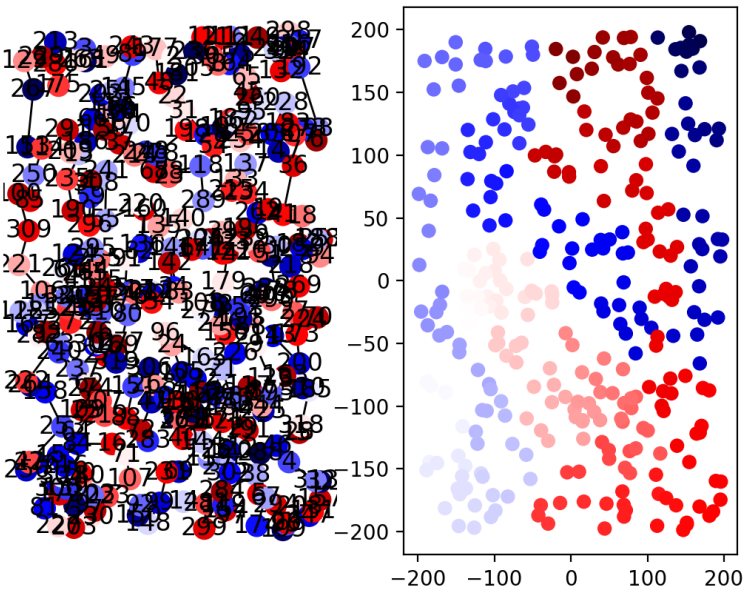
Cost: 1563.5807457086858

3.tsp: 321 [259, 182, 218, 269, 27, 224, 70, 173, 200, 163, 185, 310, 15, 28, 318, 123, 195, 87, 225, 244, 117, 287, 311, 171, 194, 125, 271, 166, 165, 21, 226, 156, 246, 33, 108, 279, 83, 34, 142, 146, 135, 160, 220, 82, 136, 91, 55, 241, 59, 1, 308, 235, 190, 296, 295, 255, 44, 115, 43, 265, 124, 61, 266, 320, 121, 272, 233, 315, 110, 168, 127, 245, 210, 76, 0, 301, 270, 169, 277, 80, 176, 317, 202, 134, 52, 212, 24, 96, 306, 47, 268, 104, 57, 77, 40, 201, 261, 130, 189, 105, 152, 292, 35, 56, 144, 119, 126, 275, 120, 312, 29, 191, 288, 46, 4, 32, 313, 157, 131, 247, 208, 20, 93, 199, 294, 174, 5, 67, 38, 302, 284, 149, 154, 297, 281, 299, 14, 69, 231, 99, 148, 150, 92, 107, 16, 71, 72, 283, 88, 151, 319, 237, 62, 274, 207, 219, 25, 109, 64, 253, 84, 50, 180, 303, 223, 7, 230, 79, 203, 227, 81, 139, 172, 300, 216, 304, 155, 276, 42, 256, 222, 264, 158, 240, 63, 53, 282, 167, 112, 11, 285, 106, 221, 309, 85, 100, 250, 114, 153, 129, 267, 229, 213, 286, 175, 291, 209, 13, 186, 66, 3, 161, 263, 49, 243, 89, 145, 251, 113, 293, 159, 138, 86, 10, 184, 170, 37, 214, 248, 73, 278, 68, 23, 78, 48, 177, 2, 236, 205, 181, 22, 31, 198, 128, 54, 118, 289, 140, 17, 204, 141, 26, 39, 74, 122, 179, 252, 193, 58, 9, 211, 143, 178, 290, 103, 238, 280, 60, 30, 51, 101, 239, 196, 232, 19, 262, 234, 215, 133, 137, 254, 258, 102, 187, 162, 95, 260, 45, 65, 164, 8, 305, 12, 116, 111, 75, 249, 242, 90, 97, 314, 132, 192, 197, 316, 298, 217, 98, 6, 273, 307, 183, 228, 206, 147, 36, 41, 18, 188, 257, 94]

Cost: 6004.121205164612


2250
2000
1750
1500
1250
1000
750
500
0
500
1000
1500


200
150
100
50
0
−50
−100
−100
−50
0
50
100


200
150
100
50
0
−50
−100
−150
−200
−200
−100
0
100
200