Assignment 5: The use of move evaluations (deltas) from previous iterations in 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 with move evaluations

Pseudocode:

```
Initialize cost of the initial solution
Set the solution as the initial solution
Identify selected nodes and non-selected nodes
Initiate a list of moves(LM)

Loop until no move can be found after searching the whole LM:
    Choose a move yielding the best improvement

    Check if the move is still valid:
        If node is inter-route:
            If (node from the cycle is still within it AND
                node from outside cycle is still outside it AND
                node from the cycle still forms the same edges with its
                neighbours):
               Apply the move
             Else:
               Remove the move from LM

        If node is intra-route:
            If (node from the cycle is still within it AND
                removed edges are in the cycle in the same relative direction
```

```
                        (both original or both reversed)):
                      Apply the move
                    Else:
                      Store the move for later
                   If at least one edge no longer exists in the cycle:
                      Remove move from possible improving moves

       Reintroduce the stored moves for later for next iterations.

        Add new moves resulting from making the move into the list of all possible
improving moves.

    Return the final solution
```

Function performance

Method	Dataset A	Dataset B
Random generation, edge, steepest	73855.835(70939-77610)	48296.625(45319-50992)
TRandom, edge, steepest with move evaluations	73627.1(71082-75503)	48288.1(46560-49845)

Average running time

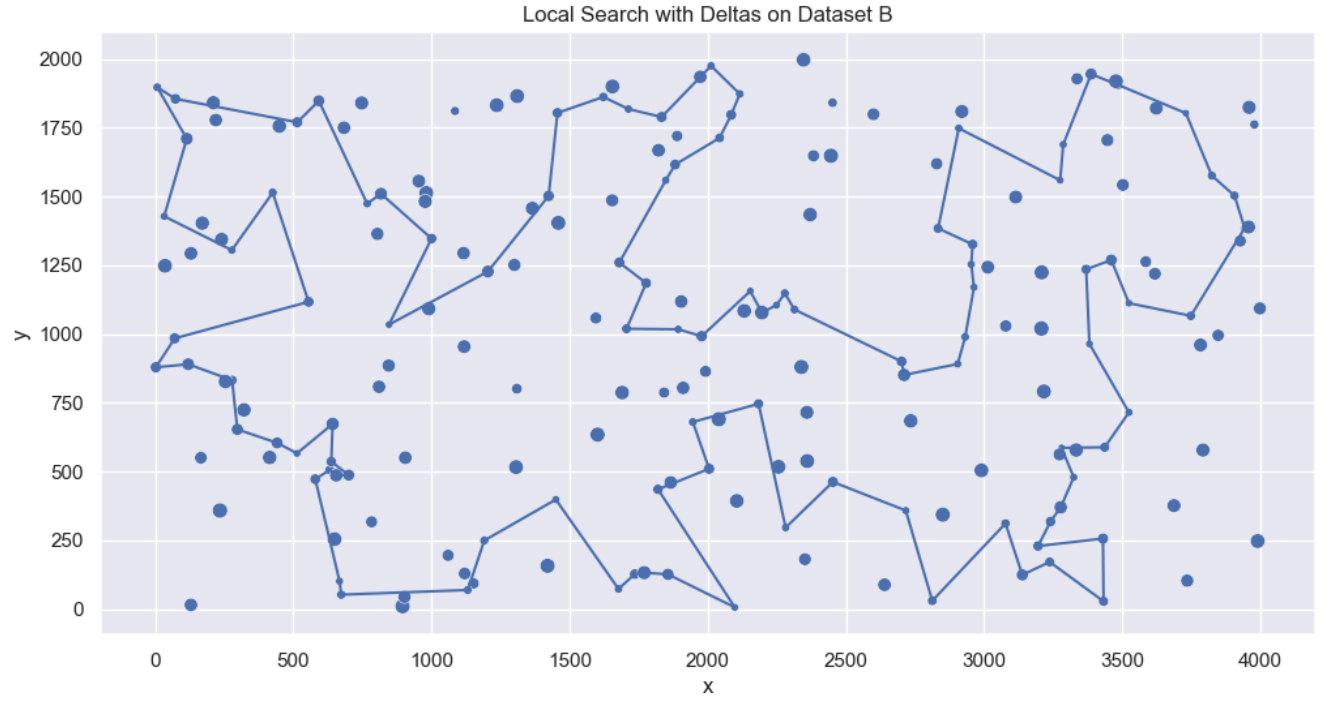
Method	Dataset A	Dataset B
Random generation, edge, steepest	3.3 s	3.17 s
Random, edge, steepest with move evaluations	0.13 s	0.12 s

Dataset A:

Best solution: [78, 145, 25, 44, 120, 129, 2, 75, 86, 101, 1, 152, 97, 26, 53, 180, 154, 135, 70, 127, 123, 131, 149, 162, 133, 151, 51, 118, 59, 116, 65, 47, 43, 184, 35, 84, 112, 4, 190, 10, 177, 54, 48, 34, 160, 42, 5, 115, 46, 139, 41, 193, 159, 181, 146, 22, 18, 108, 117, 0, 143, 183, 89, 186, 23, 137, 176, 80, 79, 63, 94, 148, 9, 62, 102, 49, 14, 144, 21, 164, 27, 90, 165, 40, 185, 106, 178, 52, 55, 57, 92, 179, 196, 81, 157, 31, 113, 175, 171, 16]

Dataset B:

Best solution: [168, 139, 11, 138, 33, 160, 29, 12, 0, 109, 35, 143, 159, 106, 124, 62, 18, 55, 34, 152, 183, 140, 149, 28, 20, 148, 47, 94, 179, 185, 130, 95, 86, 166, 194, 176, 113, 26, 103, 89, 114, 137, 127, 165, 163, 187, 153, 81, 77, 111, 8, 82, 21, 141, 91, 79, 61, 36, 177, 5, 175, 80, 190, 193, 31, 164, 73, 54, 151, 117, 198, 156, 1, 16, 27, 38, 131, 122, 135, 63, 100, 40, 107, 133, 10, 90, 191, 51, 121, 118, 134, 6, 188, 169, 132, 70, 3, 15, 145, 195]



Conclusion:

With use of move evaluations from previous iterations we can see that we get really good results of time, also getting really close to random generation, edge exchange, steepest algorithm.