# Evolutionary Computation - lab assignment 5

Szymon Bujowski, 148050, source: https://github.com/bujowskis/put-evolutionary-computation

## Problem description

The goal of the task is to improve the time efficiency of the steepest local search with the use move evaluations (deltas) from previous iterations (list of improving moves) using the neighborhood, which turned out to be the best in assignment 3. Both inter-route and intra-route moves should be included in the list. In the case of inter-route moves of the exchange of two edges, you should carefully read the description of the lectures on the traveling salesman problem, in particular:

• Consider 3 situations (when we browse moves from LM):

    • Removed edges (defining the saved move) no longer exist in the current solution (at least one of them)

        • -> remove the move from LM

    • Removed edges occur in the current solution in a different relative direction from the saved one – not applicable now but the move can be applied in the future

        • -> leave the move in LM but do not apply it browse LM further

    • Removed edges appear in the current solution in the same relative direction (also both reversed)

        • -> perform (apply) the move and remove from LM

    • When evaluating new moves we need to consider also moves with inverted edges

This mechanism should be used separately from candidate moves. Optionally, you can try to implement them both together.

As starting solutions use random solutions.

As baseline report also results of the steepest local search with random starting solutions without these mechanisms.

Computational experiment: Run each of the methods 200 times.

Reporting results: Use tables as in the previous assignment.

The outline of the report as previously.

## Implemented algorithms and pseudocodes

### Notes

- 50% was the required number of nodes in a solution, and "required number of nodes" will be referred to this way in the pseudocode
- "**Total move cost**" is the total change in objective function after adding a node, that is, its edge distance and additional cost
- Pseudocode is simplified to the required configuration of Local Search - that is: Steepest LS, Random starting solution, two edges exchange intra moves

### Local Search with deltas

```
1. Generate random initial solution
2. Initialize list of all possible improving moves
2.Ad.1. In case of inter nodes exchanges - moves that put each node outside of cycle in place of each node within the cycle
2.Ad.2. In case of intra edge exchanges - 2-edge exchange; e.g. given edges (1,2) and (3,4), break them and connect (1,3) and (2,4)
3. WHILE there are improving moves
3.1. Choose a move yielding the best improvement
3.2. Check if the move is still valid; act depending on the conditions
3.2.Ad.1. In case of inter nodes exchange, conditions are:
   1. Node from the cycle is still within it
   2. Node from outside cycle is still outside it
   3. Node from the cycle still forms the same edges with its neighbors
If all conditions are met, apply the move. Otherwise, remove it from possible improving moves.
3.2.Ad.2. In case of intra edge exchange, conditions are:
   1. Removed edges are in the cycle in the same relative direction
      (both original or both reversed) -> apply the move
   2. Removed edges are in the cycle in different relative direction
      (one original and one reversed) -> store the move for later
   3. At least one edge no longer exists in the cycle -> remove move
      from possible improving moves
3.3. Reintroduce moves stored for later (for next iterations)
3.4. Add new moves resulting from making the move into list of all possible improving moves
4. Return the cycle
```

## Results

Note: All best solutions were checked with the solution checker

### Statistics

| **Method** | **TSPA** avg (min - max) | **TSPB** avg (min - max) |
|---|---|---|
| Local Search (no deltas) | 74222 (71725 - 78624) | 48627 (45619 - 52245) |
| Local Search with deltas | 74222 (71725 - 78624) | 48627 (45619 - 52245) |

### Running times (s)

| **Method** | **TSPA** avg (min - max) | **TSPB** avg (min - max) |
|---|---|---|
| Local Search (no deltas) | 2.06 (1.82 - 2.31) | 2.08 (1.83 - 2.38) |
| Local Search with deltas | 0.22 (0.18 - 0.42) | 0.22 (0.17 - 0.32) |

### Local Search statistics

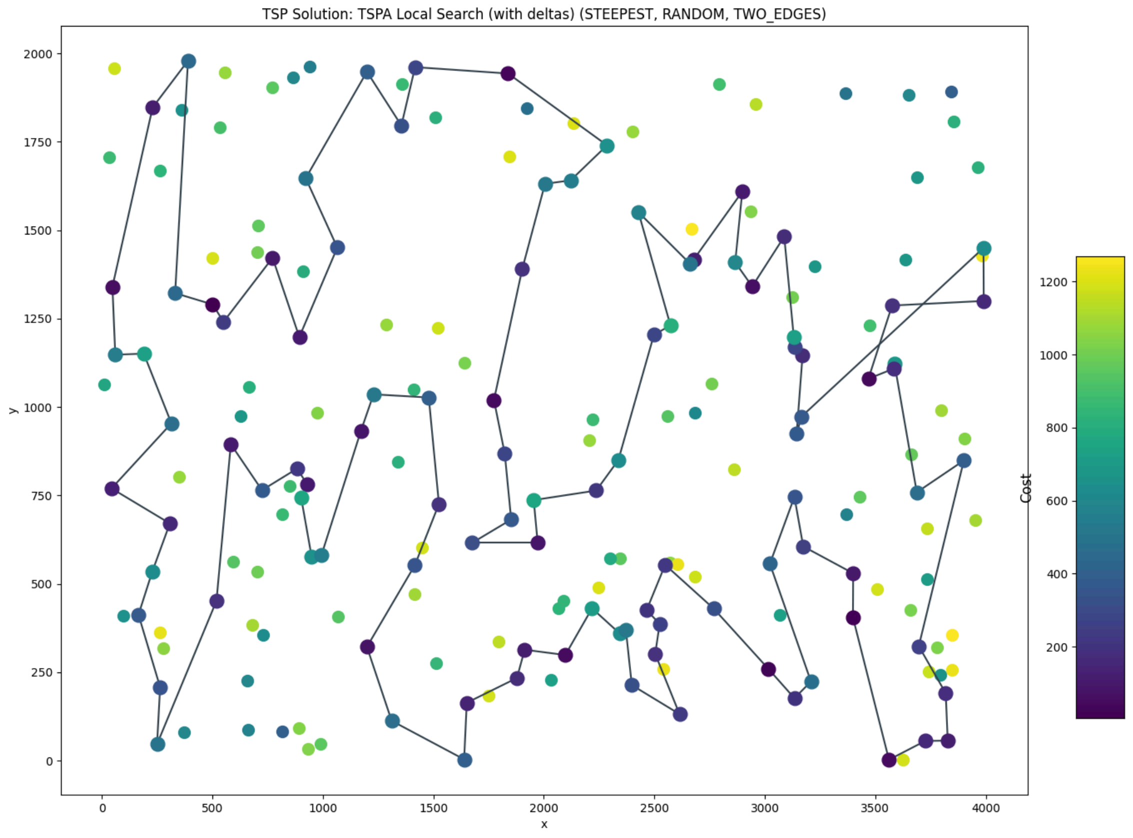
Note: additional statistics I thought may be interesting to investigate

| **Method** | **Moves evaluated** TSPA avg (min - max) TSPB avg (min - max) | **Total moves** TSPA avg (min - max) TSPB avg (min - max) | **Inter moves %** TSPA avg (min - max) TSPB avg (min - max) | **Intra moves %** TSPA avg (min - max) TSPB avg (min - max) |
|---|---|---|---|---|
| Local Search (no deltas) | 1995750 (1779050 - 2272400) 2005617 (1764100 - 2272400) | 197 (161 - 241) 196 (163 - 230) | 28 (22 - 34) 32 (24 - 41) | 70 (61 - 77) 68 (59 - 76) |
| Local Search with deltas | 115445 (94123 - 137768) 113601 (94418 - 134192) | 197 (161 - 241) 196 (163 - 230) | 28 (22 - 34) 32 (24 - 41) | 70 (61 - 77) 68 (59 - 76) |

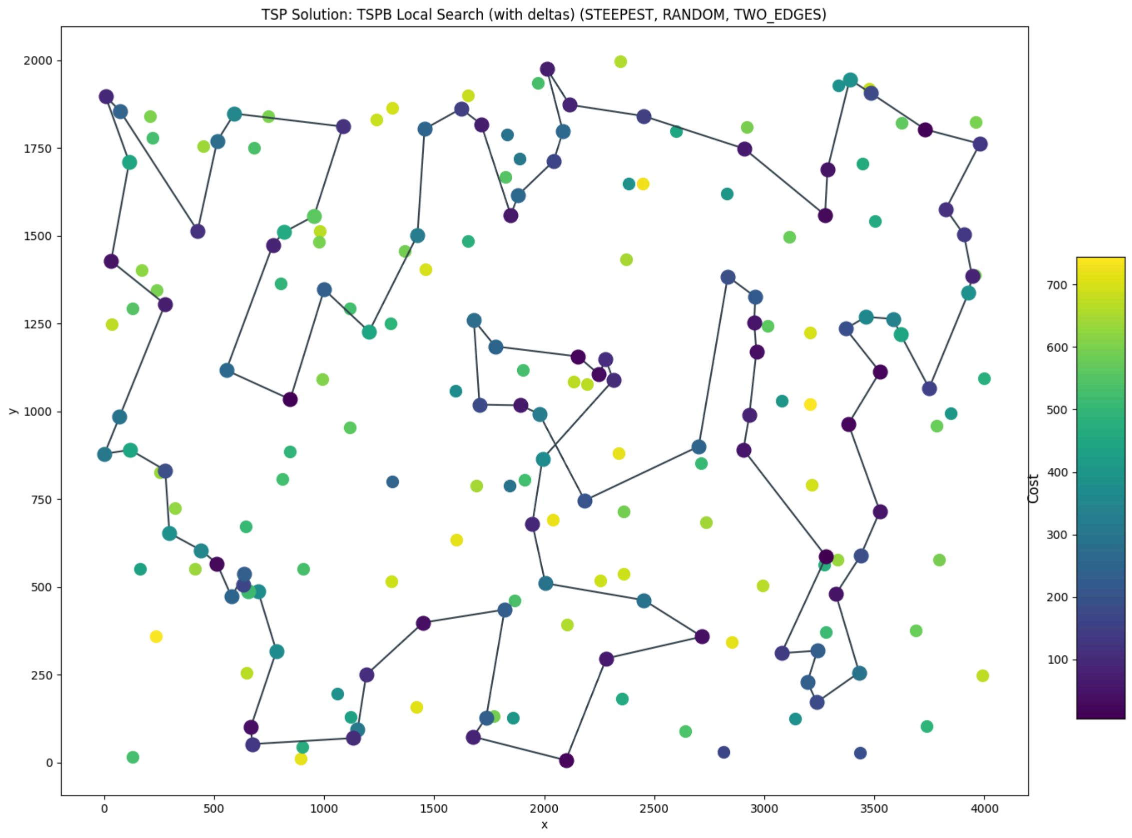
## Best solutions

### Note: additional cost is depicted using a color scale

### TSPA



## TSPB



TSP Solution: TSPB Local Search (with deltas) (STEEPEST, RANDOM, TWO_EDGES)
x
y
Cost
0
500
1000
1500
2000
2500
3000
3500
4000
250
750
1250
1750
100
200
300
400
500
600
700

## Conclusions

- Use of deltas leads to significant time complexity improvement;
  On the considered problem instances
    - Around 10x smaller avg running times
    - Around 17x less moves evaluated
- Use of deltas does not impair the solution’s quality
- It’s worth mentioning that memory complexity increases a lot, since moves are dynamically computed and stored for potential later use