

Evolutionary Computation

Local Search Report

Authors and Source Code

- **Authors:**
 - Maksymilian Żmuda-Trzebiatowski 156 051
 - Krzysztof Bryszak 156 052
 - **Source Code Repository:**
https://github.com/MZmuda-Trzebiatowski/Evo_Comp_LS_candidate
-

Problem Description

The problem involves a set of nodes, each defined by three columns of integers:

1. **X-coordinate**
2. **Y-coordinate**
3. **Node Cost**

The goal is to select exactly 50% of the nodes (rounding up if the total number of nodes is odd) and form a Hamiltonian cycle (a closed path) through the selected set. The objective is to minimize the total sum of the path length plus the total cost of the selected nodes.

- **Distance Calculation:** Distances are calculated as Euclidean distances, mathematically rounded to integer values.
 - **Optimization Constraint:** A distance matrix must be calculated immediately after reading an instance. The optimization methods should only access this distance matrix, not the original node coordinates.
-

Implemented Algorithms (pseudocode)

- **Local Search Steepest**

- Input:
 - tour: list of node indices representing the current solution
 - d: distance matrix between nodes
 - nodes: list of nodes, each with an associated cost
 - use_swap_intra: boolean flag indicating whether to use swap or 2-opt for intra-route optimization
- Output:
 - tour: improved route after applying local search

```
Let  $n \leftarrow \text{size}(d)$  and  $k \leftarrow \text{size}(\text{tour})$ .
Initialize a boolean array is_selected of size  $n \leftarrow \text{false}$ .
For each node  $v$  in tour, set is_selected[v]  $\leftarrow$  true.
Repeat
  a. Initialize best_move.delta  $\leftarrow$  0.
  b. Inter-route improvement (V-E exchange):
    For each position  $i$  in tour do
      For each unselected node  $j$  do
        Compute delta  $\leftarrow$  delta_V_E_exchange(tour, i, j, d, nodes).
        If delta < best_move.delta then
          Update best_move  $\leftarrow$  {type = 0, i, j, delta}.
  c. Intra-route improvement (Swap or 2-opt):
    For each pair of indices  $(i, j)$  with  $i < j$  do
      If use_swap_intra = true, set delta  $\leftarrow$  delta_swap(tour, i, j, d) and type  $\leftarrow$  1.
      Else, set delta  $\leftarrow$  delta_2opt(tour, i, j, d) and type  $\leftarrow$  2.
      If delta < best_move.delta then
        Update best_move  $\leftarrow$  {type, i, j, delta}.
  d. If best_move.delta  $\geq$  0, then break.
  e. Apply the best move found:
    - If best_move.type = 0 (V-E exchange):
      Remove node at position  $i$ , insert node  $j$  using apply_V_E_exchange.
      Update selection flags in is_selected.
    - If best_move.type = 1 (Swap): apply apply_swap(tour, i, j).
    - If best_move.type = 2 (2-opt): apply apply_2opt(tour, i, j).
Until no improving move exists (best_move.delta  $\geq$  0).
Return the improved tour.
```

- **Local Search Steepest, 2-opt, Candidate**

- Input:
 - tour: current sequence of selected nodes
 - d: matrix of pairwise distances between nodes
 - nodes: list of nodes with associated costs
 - candidate_list: adjacency-based candidate neighbor list for each node

- Output:
 - tour: improved solution after steepest-descent local search using candidate moves

Let $n \leftarrow \text{size}(d)$ and $k \leftarrow \text{size}(\text{tour})$.

Initialize a boolean array `is_selected` of length $n \leftarrow \text{false}$.

For each node v in `tour`, set `is_selected[v] \leftarrow true`.

Repeat

a. Initialize `best_move.delta \leftarrow 0`.

b. **For each** position i in `tour` **do**

Let `current_node \leftarrow tour[i]`.

For each `candidate` in `candidate_list[current_node]` **do**

- **If** `candidate` is adjacent to `current_node` in `tour`, **continue**.

- **If** `candidate` is *selected* (already in `tour`):

Locate its position j in `tour`.

Compute `delta1 \leftarrow delta_2opt(tour, i, j, d)` and

`delta2 \leftarrow delta_2opt(tour, (i-1) mod k, (j-1) mod k, d)`.

If either `delta1` or `delta2` is smaller than `best_move.delta`,

update `best_move` accordingly (*type = 2-opt*).

- **Else if** `candidate` is *not selected*:

Compute `delta1 \leftarrow delta_V_E_exchange(tour, (i + k - 1) % k,`

`candidate, d, nodes)`.

Compute `delta2 \leftarrow delta_V_E_exchange(tour, (i + 1) % k, candidate,`

`d, nodes)`.

If either `delta1` or `delta2` is smaller than `best_move.delta`,

update `best_move` accordingly (*type = V-E exchange*).

c. **If** `best_move.delta \geq 0`, **terminate** (no further improvement found).

d. **Apply the best move:**

- If `best_move.type = 0`, execute `apply_V_E_exchange(tour, best_move.i,`
`best_move.j)` and

update selection flags accordingly.

- If `best_move.type = 2`, execute `apply_2opt(tour, best_move.i, best_move.j)`.

Until no improving move exists.

Return the final `tour`.

Results and Analysis

Table of results

| Algorithm | Instance A | Instance B |
|---|--------------------------|--------------------------|
| Random | 264152 (239114 - 291474) | 212540 (185581 - 238526) |
| NN end-only | 85108.5 (83182 - 89433) | 54390.4 (52319 - 59030) |
| NN all-pos | 73302.4 (71695 - 75953) | 48498.9 (44242 - 57283) |
| Greedy Cycle | 72617.6 (71488 - 74410) | 51339.5 (48765 - 57324) |
| NN all-pos 2-regret | 117138 (108151 - 124921) | 74444.5 (69933 - 80278) |
| Greedy Cycle 2-regret | 115579 (105692 - 126951) | 72740 (67809 - 78406) |
| NN all-pos 2-regret weighted (0.5, 0.5) | 72401.2 (70010 - 75452) | 47664.5 (44891 - 55247) |
| Greedy Cycle 2-regret weighted (0.5, 0.5) | 72129.7 (71108 - 73395) | 50897.1 (47144 - 55700) |
| Steepest LS swap, rand init | 88179.1 (80805 - 97462) | 62949.8 (54696 - 71421) |
| Steepest LS swap, greedy init | 72010 (69801 - 75440) | 47137 (44488 - 54391) |
| Steepest LS 2-opt, rand init | 73975.5 (71248 - 78900) | 48421.5 (45882 - 51676) |
| Steepest LS 2-opt, greedy init | 70722.3 (69540 - 72546) | 46342 (44320 - 51431) |
| Greedy LS swap, rand init | 86548 (79976 - 94362) | 61330.1 (54462 - 70020) |
| Greedy LS swap, greedy init | 72010.4 (69801 - 75440) | 47108.3 (44456 - 54372) |
| Greedy LS 2-opt, rand init | 73324.9 (71455 - 76688) | 48189.2 (44632 - 51038) |
| Greedy LS 2-opt, greedy init | 70943.8 (69497 - 73149) | 46372.4 (44320 - 51462) |
| Steepest LS 2-opt, rand init, candidate | 77709.6 (73310 - 82396) | 48362.6 (45822 - 52155) |

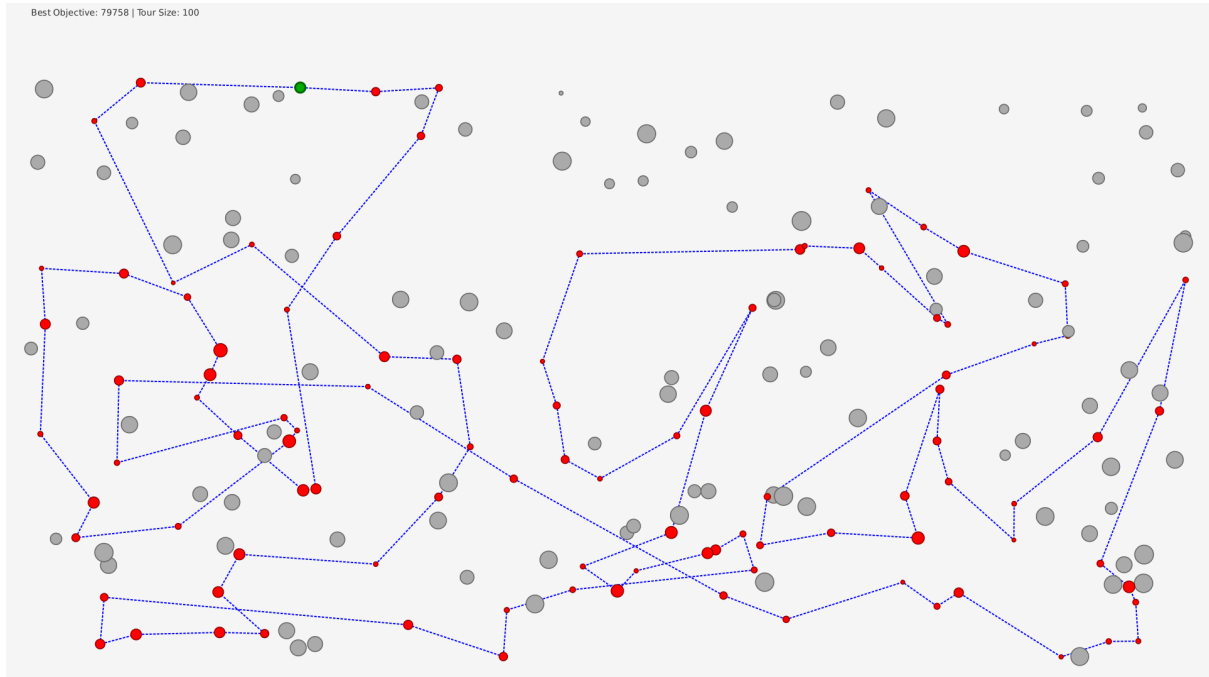
Table of Runtimes

| Algorithm | Instance A (runtime in sec) | Instance B (runtime in sec) |
|---|-----------------------------|-----------------------------|
| Steepest LS swap, rand init | 1.964 | 1.932 |
| Steepest LS swap, greedy init | 1.245 | 0.297 |
| Steepest LS 2-opt, rand init | 1.348 | 1.344 |
| Steepest LS 2-opt, rand init cand | 0.496 | 0.444 |
| Steepest LS 2-opt, rand init cand + list | 0.711 | 0.663 |
| Steepest LS 2-opt, greedy init | 0.28 | 0.318 |
| Greedy LS swap, rand init | 0.44 | 0.33 |
| Greedy LS swap, greedy init | 0.236 | 0.245 |
| Greedy LS 2-opt, rand init | 0.362 | 0.27 |
| Greedy LS 2-opt, greedy init | 0.259 | 0.254 |

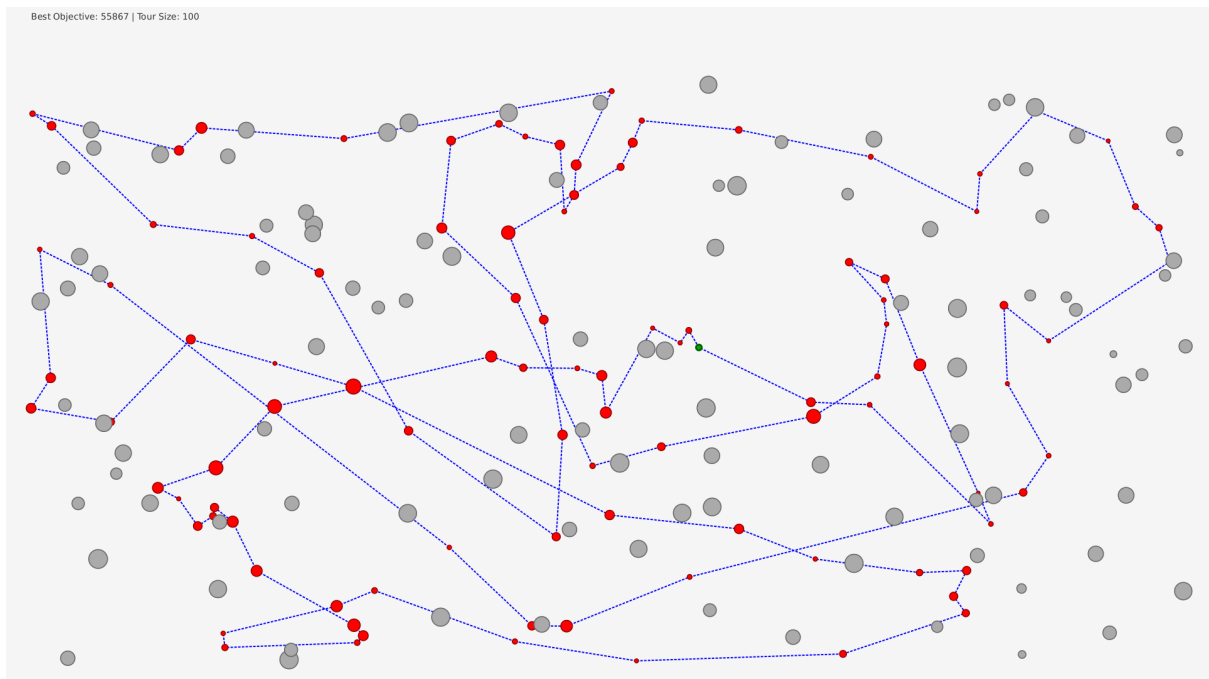
Visual Comparisons (Visual Comparison) - Instance A

The size of the dot corresponds to its cost (the bigger it is the bigger the cost), and the green dot is the starting node.

- **Steepest LS swap, rand init**
 - Instance A

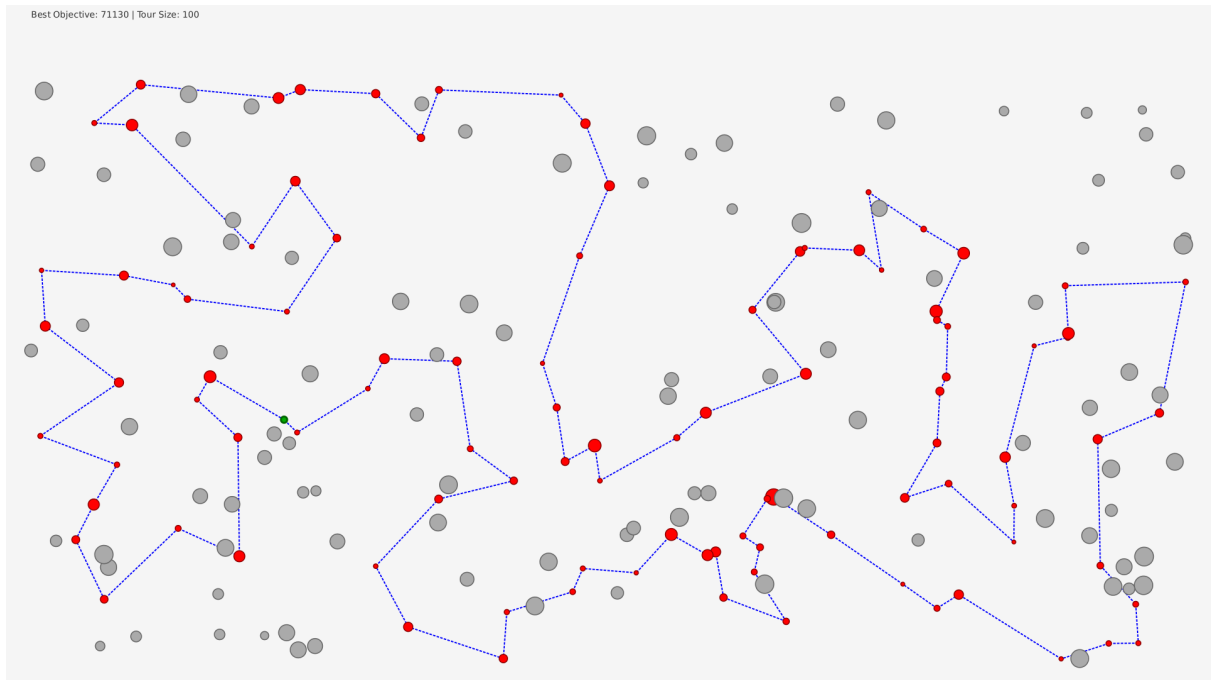


- Instance B

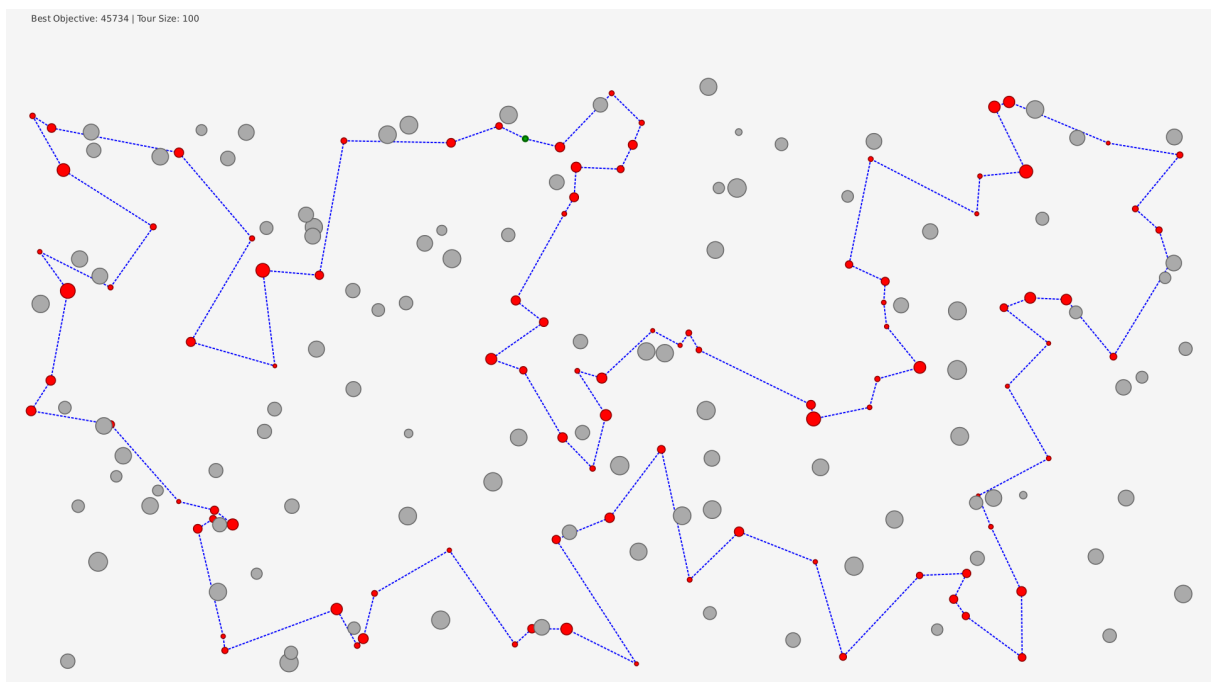


- **Steepest LS 2-opt, rand init**

- Instance A

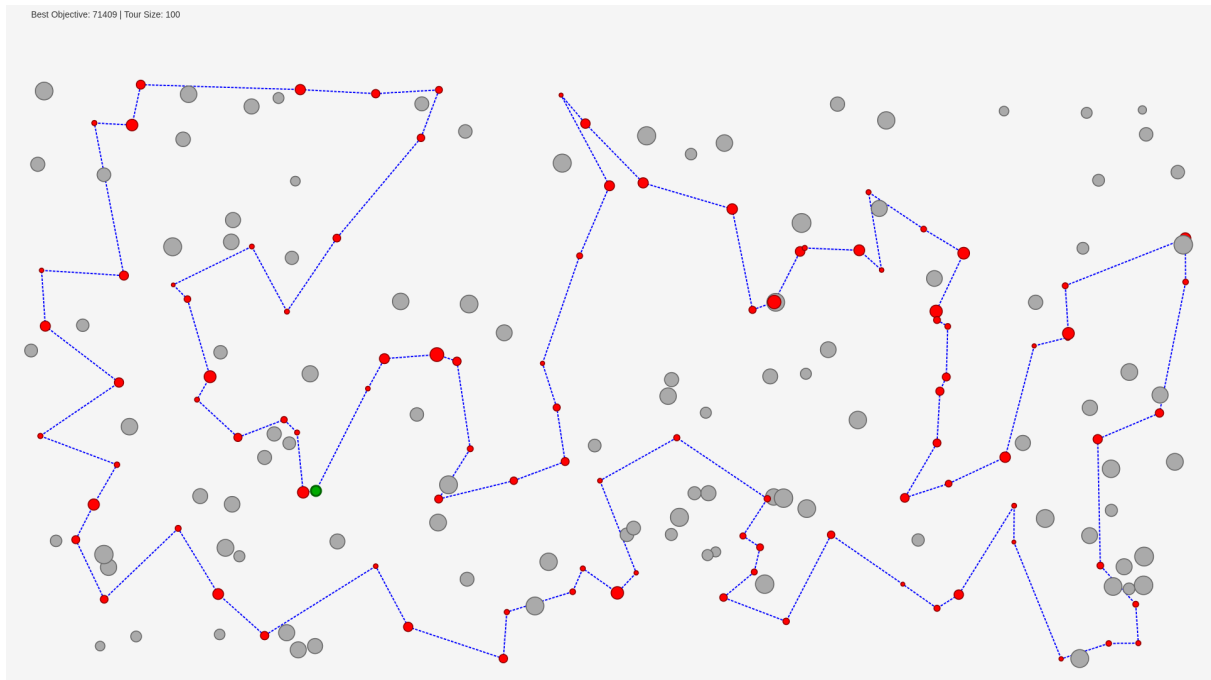


- Instance B

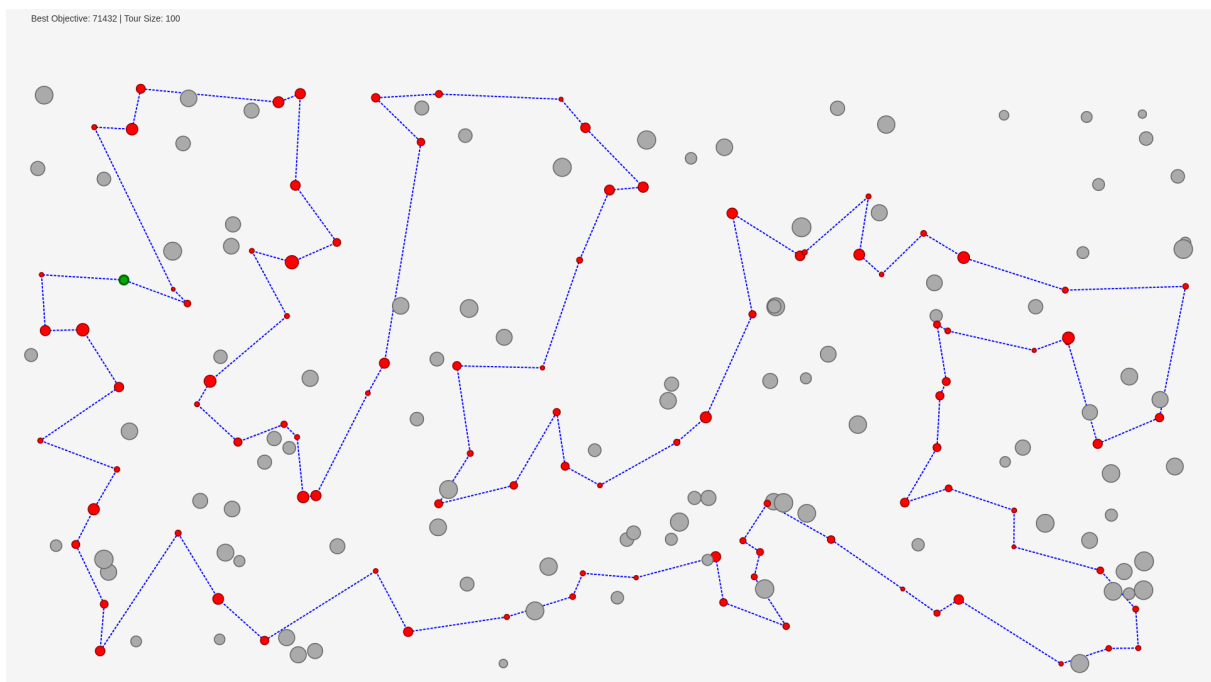


- **Steepest LS 2-opt, rand init, candidate**

- Instance A



- Instance B



Best Solutions

The best solutions were checked with the solution checker.

- **Steepest LS swap, rand init**

- Instance A

93, 108, 18, 193, 139, 118, 51, 151, 162, 123, 35, 84, 112, 4, 190, 10, 177, 127, 70, 135, 154, 101, 97, 26, 100, 53, 158, 180, 121, 124, 148, 94, 63, 79, 80, 176, 137, 9, 62, 102, 49, 178, 106, 144, 14, 138, 165, 40, 185, 52, 152, 1, 2, 82, 129, 55, 57, 92, 78, 145, 196, 90, 81, 31, 56, 113, 175, 171, 16, 25, 44, 120, 75, 86, 133, 59, 181, 160, 116, 65, 47, 184, 54, 48, 34, 146, 22, 159, 41, 96, 5, 42, 43, 131, 149, 115, 46, 0, 143, 117

- Instance B

35, 109, 0, 29, 144, 160, 33, 138, 182, 112, 151, 198, 117, 193, 31, 54, 73, 136, 142, 78, 175, 80, 190, 45, 5, 36, 141, 187, 127, 89, 103, 163, 153, 81, 82, 158, 121, 131, 1, 27, 38, 63, 135, 177, 61, 91, 77, 194, 166, 86, 95, 185, 94, 47, 148, 20, 28, 140, 183, 152, 155, 3, 15, 145, 43, 11, 104, 21, 25, 51, 90, 122, 107, 40, 133, 10, 147, 70, 13, 195, 168, 132, 169, 188, 6, 134, 139, 8, 111, 159, 124, 62, 18, 34, 55, 128, 176, 113, 106, 143

- **Steepest LS 2-opt, rand init**

- Instance A

116, 5, 42, 43, 35, 184, 177, 54, 48, 160, 34, 181, 146, 22, 159, 193, 41, 115, 46, 68, 139, 69, 18, 108, 140, 93, 117, 0, 143, 183, 89, 23, 137, 176, 80, 79, 122, 63, 94, 124, 167, 148, 9, 62, 102, 49, 144, 14, 138, 3, 178, 106, 52, 55, 57, 129, 92, 78, 145, 179, 185, 40, 119, 165, 90, 81, 196, 31, 113, 175, 171, 16, 25, 44, 120, 2, 74, 152, 97, 1, 101, 75, 86, 26, 100, 121, 53, 180, 154, 135, 70, 127, 123, 162, 133, 151, 51, 118, 59, 65

- Instance B

169, 132, 70, 3, 15, 145, 13, 195, 168, 139, 11, 182, 138, 104, 8, 144, 33, 160, 29, 0, 109, 35, 143, 159, 106, 124, 128, 62, 18, 55, 34, 152, 183, 140, 199, 4, 149, 28, 20, 60, 148, 47, 94, 179, 99, 130, 95, 185, 86, 166, 176, 113, 114, 137, 127, 89, 103, 163, 187, 153, 81, 77, 111, 82, 21, 141, 91, 61, 36, 177, 5, 78, 175, 45, 80, 190, 193, 31, 73, 54, 117, 1, 27, 38, 102, 63, 135, 122, 100, 40, 107, 133, 90, 131, 121, 125, 51, 147, 6, 188

- **Steepest LS 2-opt, rand init, candidate**

- Instance A

159, 22, 146, 195, 181, 34, 160, 48, 54, 177, 10, 184, 84, 112, 123, 127, 135, 154, 180, 53, 26, 86, 75, 101, 1, 97, 152, 2, 120, 44, 25, 16, 171, 175, 113, 31, 78, 145, 92, 129, 57, 55, 52, 178, 106, 185, 119, 40, 196, 81, 90, 165, 138, 14, 49, 102, 144, 62, 9, 15, 148, 124, 94, 63, 79, 80, 133, 162, 151, 51, 176, 137, 23, 186, 89, 183, 143, 117, 0, 118, 59, 149, 131, 65, 116, 43, 42, 5, 115, 139, 198, 46, 68, 93, 140, 108, 69, 18, 193, 41

- Instance B

159, 22, 146, 195, 181, 34, 160, 48, 54, 177, 10, 184, 84, 112, 123, 127, 135, 154, 180, 53, 26, 86, 75, 101, 1, 97, 152, 2, 120, 44, 25, 16, 171, 175, 113, 31, 78, 145, 92, 129, 57, 55,

52, 178, 106, 185, 119, 40, 196, 81, 90, 165, 138, 14, 49, 102, 144, 62, 9, 15, 148, 124, 94, 63, 79, 80, 133, 162, 151, 51, 176, 137, 23, 186, 89, 183, 143, 117, 0, 118, 59, 149, 131, 65, 116, 43, 42, 5, 115, 139, 198, 46, 68, 93, 140, 108, 69, 18, 193, 41

Conclusions

- The candidate system did reduce the runtime significantly.
- The average value did increase (bad) quite a bit however the best found objective value is not that far apart from the full steepest LS.