Evolutionary Computation - Assignment 3

Sebastian Chwilczyński 148248, Karol Cyganik 148250

November 6, 2023

1 Description of the problem

We are given a graph with n n nodes, each described by its x and y coordinates and a node cost. The goal is to select precisely $ceil(n/2)$ nodes that form a Hamiltonian cycle (closed path) and minimize

$$\sum_{e \in E} cost(e) + \sum_{v \in V} cost(v)$$

Where E is a set of selected edges, $cost(e)$ is Euclidean distance between two nodes rounded mathematically to an integer value, V is a set of selected nodes and $cost(v)$ is node cost.

2 Pseudocode

2.1 Neighborhood moves generation methods

For simplicity we assume that procedures by default has access to:

- *solution* - List of nodes/edges that are in current solution
- *not_solution* List of nodes that aren't in current solution

Note on $N(x)$: $N(x)$ denotes the neighborhood of the current solution x. It is composed of all possible moves from 1 Intra and 1 Inter operation. $N_{\text{rand}}(x)$ represents a random permutation of all neighbors. This means that each iteration of the while loop will explore potential solutions in a different, random order.

---

procedure INTRA2NODESEXCHANGE
    $node1$, $node2 \leftarrow$ next 2 nodes to consider from *solution*
    $left\_neighbor1$, $right\_neighbor1 \leftarrow$ nodes adjacent to $node1$
    $left\_neighbor2$, $right\_neighbor2 \leftarrow$ nodes adjacent to $node2$
    $current\_cost \leftarrow$ sum of distances between nodes and their neighbors
    $cost\_after\_exchange \leftarrow$ sum of distances as we swap $node1$ with $node2$
    return $current\_cost - cost\_after\_exchange$
end procedure

---

procedure INTRA2EDGESEXCHANGE
    $edge1$, $edge2 \leftarrow$ next 2 edges to consider from *solution*
    $edge1\_new$, $edge2\_new \leftarrow$ Edges obtained by commuting second node of $edge1$ and $edge2$
    $current\_cost \leftarrow length(edge1) + length(edge2)$
    $cost\_after\_exchange \leftarrow length(edge1\_new) + length(edge2\_new)$
    return $current\_cost - cost\_after\_exchange$
end procedure

---

procedure INTER2NODESEXCHANGE
    $node1 \leftarrow$ next node from *solution*
    $node2 \leftarrow$ next nodes from *no_solution*
    $left\_neighbor1$, $right\_neighbor1 \leftarrow$ nodes adjacent to $node1$
    $current\_cost \leftarrow$ sum of distances between $node1$ and its neighbors + cost of $node1$
    $cost\_after\_exchange \leftarrow$ sum of distances between $node2$ and $node1$ neighbors + cost of $node2$
    return $current\_cost - cost\_after\_exchange$
end procedure

2.2 Local search - steepest

Algorithm 1 Algorithm that evaluates every neighbour and select the one with the biggest improvement of the objective function.

Input: An initial solution x
Output: The best found solution

```
x ← initial solution
improved ← true
while improved do
    best_move ← null
    best_score ← 0
    improved ← false
    for all move ∈ All possible moves from N(x) do
        score ← Evaluate move on x
        if score > best_score then
            best_score ← score
            best_move ← move
        end if
    end for
    if best_score > 0 then
        x ← Apply(x, best_move)
        improved ← true
    end if
end while
return x
```

2.3 Local search - greedy

Algorithm 2 Algorithm that moves to the first encountered neighbour that yields improvement in the objective function

Input: An initial solution x
Output: The best found solution

$x \leftarrow$ Generate initial solution
improved $\leftarrow$ true
while *improved* **do**
    best_move $\leftarrow$ null
    best_score $\leftarrow 0$
    improved $\leftarrow$ false
    for all $y \in N_{\text{rand}}(x)$ **do** ▷ Randomized neighborhood search
        score $\leftarrow$ Evaluate move from x to y
        if *score* $>$ *best_score* **then**
            best_score $\leftarrow$ *score*
            best_move $\leftarrow y$
        end if
    end for
    if *best_score* > 0 **then**
        $x \leftarrow \text{Apply}(x, \textit{best\_move})$
        improved $\leftarrow$ true
    end if
end while
return x

3 Results

Table 1: Combined Results for TSP A & B Variants

Method	TSPA (avg(min-max))
Random	265672.09(2338179.0-289219.0)
NN	87679.135(84471.0-95013.0)
Greedy cycle	77064.41(75666.0-80321.0)
2 regret	116240.25(104829.0-124764.0)
weighted 2 regret	76341.56(74563.0-78976.0)
greedy LS, nodes, random start	95900.41(88078.0-104026.0)
greedy LS, nodes, w-2-regret start	76094.96(**74300.0**-78547.0)
greedy LS, edges, random start	81822.44(78114.0-90175.0)
greedy LS, edges, w-2-regret start	75946.44(**74300.0**-78359.0)
steepest LS, nodes, random start	97883.57(89263.0-106551.0)
steepest LS, nodes, w-2-regret start	76092.19(**74300.0**-78616.0)
steepest LS, edges, random start	82109.39(78007.0-87773.0)
steepest LS, edges, w-2-regret start	75905.52(**74300.0**-78492.0)
Method	**TSPB (avg(min-max))**
Random	267823.05(2338697.0-299450.0)
NN	79282.58(77448.0-82631.0)
Greedy cycle	70735.35(68743.0-76324.0)
2 regret	118806.91(109774.0-128550.0)
weighted 2 regret	71801.35(70153.0-77676.0)
greedy LS, nodes, random start	90621.88(84123.0-100707.0)
greedy LS, nodes, w-2-regret start	71424.31(68853.0-77581.0)
greedy LS, edges, random start	75537.28(70901.0-82619.0)
greedy LS, edges, w-2-regret start	71071.61(**68181.0**-77082.0)
steepest LS, nodes, random start	93472.44(84423.0-103405.0)
steepest LS, nodes, w-2-regret start	71423.22(69500.0-77597.0)
steepest LS, edges, random start	75503.39(69957.0-82912.0)
steepest LS, edges, w-2-regret start	71005.67(68810.0-76959.0)

Table 2: Combined Results for TSP C & D Variants

Method	**TSPC (avg(min-max))**
Random	215498.38(195689.0-236233.0)
NN	58872.68(56304.0-63697.0)
Greedy cycle	55842.07(53226.0-58876.0)
2 regret	69013.72(65095.0-73090.0)
weighted 2 regret	55946.20(54126.0-58288.0)
greedy LS, nodes, random start	68421.01(61103.0-75316.0)
greedy LS, nodes, w-2-regret start	55573.32(53661.0-57935.0)
greedy LS, edges, random start	54774.54(51266.0-59924.0)
greedy LS, edges, w-2-regret start	55386.09(53350.0-57935.0)
steepest LS, nodes, random start	69805.12(63647.0-77118.0)
steepest LS, nodes, w-2-regret start	55564.42(53600.0-57935.0)
steepest LS, edges, random start	54636.31(**50438.0**-60017.0)
steepest LS, edges, w-2-regret start	55424.30(53274.0-57935.0)
Method	**TSPD (avg(min-max))**
Random	220321.22(196249.0-241897.0)
NN	54290.68(50335.0-59846.0)
Greedy cycle	54838.01(50409.0-60964.0)
2 regret	70442.125(64682.0-74903.0)
weighted 2 regret	53691.48(49165.0-59416.0)
greedy LS, nodes, random start	67349.22(60159.0-77027.0)
greedy LS, nodes, w-2-regret start	53140.96(48138.0-59068.0)
greedy LS, edges, random start	52072.30(48308.0-57568.0)
greedy LS, edges, w-2-regret start	52913.23(48138.0-58888.0)
steepest LS, nodes, random start	69258.85(58721.0-77917.0)
steepest LS, nodes, w-2-regret start	53137.48(48138.0-59068.0)
steepest LS, edges, random start	51836.81(**47179.0**-57441.0)
steepest LS, edges, w-2-regret start	52843.935(48138.0-58594.0)

Table 3: Consolidated Time Results

Method	TSPA (s)	TSPB (s)
greedy LS, nodes, random start	13.025 (9.502 - 19.002)	12.770 (9.330 - 18.567)
greedy LS, nodes, w-2-regret start	0.885 (**0.681** - 1.298)	0.994 (0.761 - 1.357)
greedy LS, edges, random start	12.900 (9.724 - 18.722)	12.385 (8.978 - 17.687)
greedy LS, edges, w-2-regret start	0.996 (0.729 - 1.452)	1.129 (**0.747** - 1.478)
steepest LS, nodes, random start	12.231 (8.282 - 17.082)	11.822 (7.592 - 16.753)
steepest LS, nodes, w-2-regret start	0.917 (0.684 - 1.281)	1.057 (0.796 - 1.429)
steepest LS, edges, random start	8.177 (6.272 - 11.222)	11.491 (5.661 - 15.100)
steepest LS, edges, w-2-regret start	1.026 (0.752 - 1.298)	1.927 (0.759 - 2.109)
Method	**TSPC (s)**	**TSPD (s)**
greedy LS, nodes, random start	16.274 (8.333 - 22.001)	17.787 (8.540 - 24.055)
greedy LS, nodes, w-2-regret start	1.529 (0.623 - 2.319)	1.526 (0.799 - 2.394)
greedy LS, edges, random start	17.695 (8.575 - 23.570)	17.300 (7.132 - 24.099)
greedy LS, edges, w-2-regret start	1.459 (0.666 - 2.600)	1.773 (0.782 - 2.771)
steepest LS, nodes, random start	17.761 (7.482 - 24.661)	17.417 (6.838 - 25.102)
steepest LS, nodes, w-2-regret start	1.554 (**0.601** - 2.395)	1.950 (0.827 - 3.001)
steepest LS, edges, random start	10.768 (5.438 - 15.001)	11.325 (4.963 - 15.678)
steepest LS, edges, w-2-regret start	1.517 (0.689 - 1.957)	1.804 (**0.756** - 2.222)

4 Code

Implementation of algorithms and visualizations is available here

5 Conclusions

Local search always improves the starting heuristic solution, as well as the random one. In most cases, the greedy heuristic at the beginning gives the best results, but for TSPD and C, we can observe, that random start gave better results while having a much worse time of completion. This may be because of deeper space exploration. The times are always much worse for LS starting from a random solution. There's no strict way of saying, that any kind of local search is the best, as it differs throughout the problems.

TSPA_collage.png

.

Figure 1: TSPA

TSPB_collage.png

.

Figure 2: TSPB

TSPC_collage.png

.

Figure 3: TSPC

TSPD_collage.png

11

.

Figure 4: TSPD