Evolutionary Computation - Assignment 7

Sebastian Chwilczyński 148248, Karol Cyganik 148250

December 10, 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

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

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 Large-scale neighborhood search

Algorithm 1 Destroy - removes 20% of solution nodes. The probability of being removed is inversely proportional to the node cost

Input: current_solution
Output: destroyed_solution
1: $n\_to\_remove \leftarrow length(solution) * 0.2$
2: $removal\_probability \leftarrow solution\_costs/sum(solution\_costs)$
3: $nodes\_to\_remove \leftarrow choice(current\_solution, n\_to\_remove, removal\_probability)$
4: **return** $solution - nodes\_to\_remove$

Algorithm 2 Repair - uses weighted 2 regret greedy algorithm to find optimum in the destroyed solution

Input: destroyed_solution
Output: repaired_solution

return $greedy\_2\_regret(destroyed\_solution, weight = 1/2)$

Algorithm 3 Algorithm that explores solution space by constantly applying Destroy and Repair operators

Input: apply_local_search, max_time
Output: The best-found solution

$best\_solution \leftarrow generate\_random\_solution()$
$best\_solution \leftarrow steepest\_ls(best\_solution)$
$best\_cost \leftarrow cost(best\_solution)$
$time \leftarrow start\_timer()$
while $time < max\_time$ **do**
  $current\_solution \leftarrow Destroy(best\_solution)$
  $current\_solution \leftarrow Repair(current\_solution)$
  if $apply\_local\_search$ is True **then**
    $current\_solution \leftarrow steepest\_ls(current\_solution)$
  end if
  $new\_cost \leftarrow cost(current\_solution)$
  if $new\_cost < best\_cost$ **then**
    $best\_solution \leftarrow current\_solution$
    $best\_cost \leftarrow current\_cost$
  end if
  $time.update()$
end while
return $best\_solution$

3 Results

Table 1: Combined Results for TSP A & B Variants

Method	TSPA (avg(min-max))
steepest LS, edges, random start	77956(74782-81590)
MSLS	75183.3(74705-75631)
ILS	74133.4(73213-74934)
LNS	73650(**72886**-74242)
LNS_LS	73698.6(73238-74340)

Method	TSPB (avg(min-max))
steepest, edges, random start	71373(68219-75872)
MSLS	68174.8(66596-68707)
ILS	67076.1(66596-67787)
LNS	67142.5(66628-68275)
LNS_LS	67036.6(**66499**-68062)

Table 2: Combined Results for TSP C & D Variants

Method	TSPC (avg(min-max))
steepest LS, edges, random start	51362(48804-54796)
MSLS	49102.3(48544-49405)
ILS	48404.5(47841-49526)
LNS	47949.8(47370-48899)
LNS_LS	47881.9(**47296**-49145)

Method	TSPD (avg(min-max))
steepest LS, edges, random start	48335(45155-53428)
MSLS	45492.9(44783-46017)
ILS	44416(43667-45289)
LNS	44401(43586-45931)
LNS_LS	44172(**43351**-44929)

Table 3: Consolidated Time Results

Method	**TSPA (s)**	**TSPB (s)**
steepest LS, edges, random start	7.01 (6.95 - 7.14)	7.19 (7.13 - 7.28)
MSLS	1403 (1391 - 1430)	1450 (1422 - 1496)
ILS, LNS, LNS_LS	1403	1450
Method	**TSPC (s)**	**TSPD (s)**
steepest LS, edges, random start	6.96 (6.89 - 7.02)	7.06 (7.03 - 7.09)
MSLS	1398 (1389 - 1401)	1420 (1413 - 1440)
IILS, LNS, LNS_LS	1398	1420

Table 4: Combined Number of iterations for TSP A & B Variants

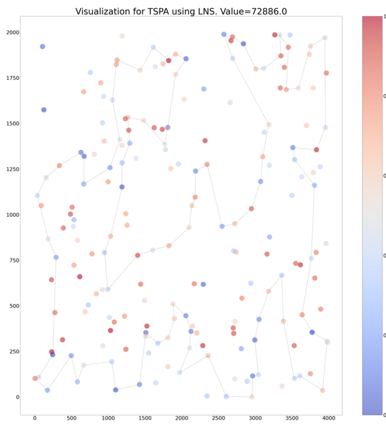
Method	**TSPA (avg(min-max))**
ILS	278(196-276)
LNS	5979(5545-6412)
LNS_LS	3924(3334-4645)
Method	**TSPB (avg(min-max))**
ILS	250(211-290)
LNS	5923(5157-6747)
LNS_LS	3798(2902-4277)

Table 5: Combined Number of iterations for TSP C & D Variants

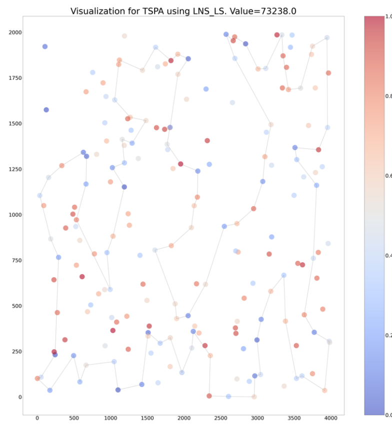
Method	**TSPC (avg(min-max))**
ILS	237(207-288)
LNS	5908(5089-6615)
LNS_LS	3153(2374-3801)
Method	**TSPD (avg(min-max))**
ILS	240(217-273)
LNS	5952(5464-6330)
LNS_LS	3564(3023-4433)

4 Code

Implementation of algorithms and visualizations is available here

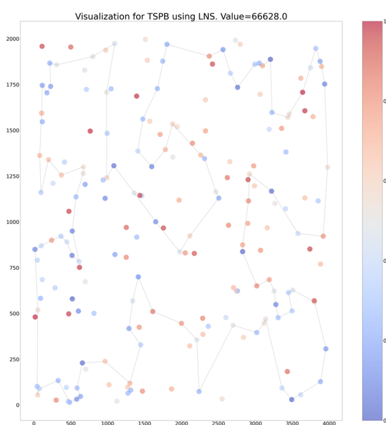


(a) LNS

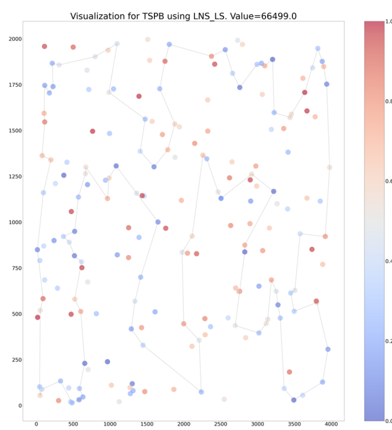


(b) LNS_LS

Figure 1: LNS vs LNS_LS on TSPA

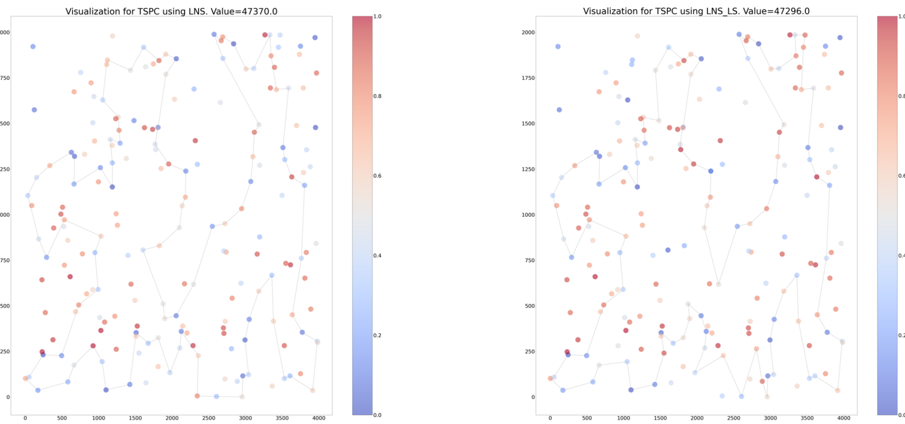


(a) LNS



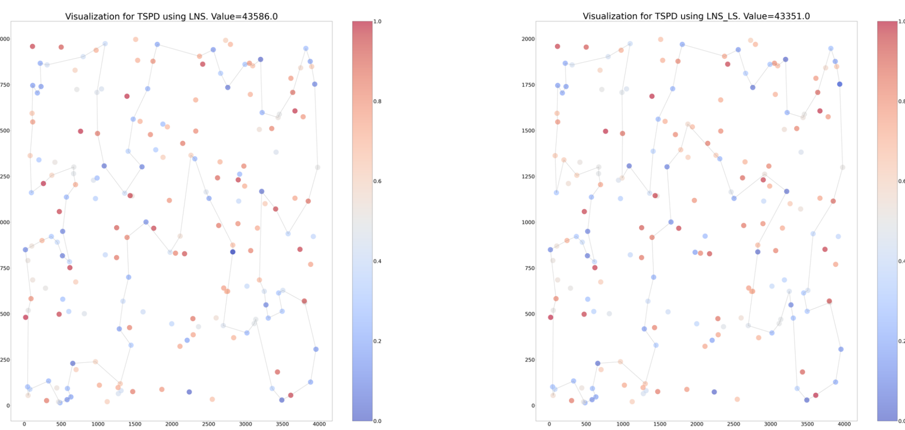
(b) LNS_LS

Figure 2: LNS vs LNS_LS on TSPB



(a) LNS (b) LNS_LS

Figure 3: LNS vs LNS_LS on TSPC



(a) LNS (b) LNS_LS

Figure 4: LNS vs LNS_LS on TSPD

5 Conclusions

Large-scale neighborhood search was always better than ILS, which was currently the best solution. Not always, but most often, the version with local search gave the best results, but the differences were not significant.