Evolutionary Computation - lab assignment 7

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

Problem description

Large-Scale Neighborhood Search (LSNS) - Reminder of the method:

```
Generate an initial solution x
x := Local search (x) (optional)
Repeat
     y := Destroy (x)
     y := Repair (y)
     y := Local search (y) (optional)
     If f(y) > f(x) then
          x := y
Until stopping conditions are met
```

Destroy operator should remove a relatively large fraction of nodes/edges from the current solution, e.g. 20-30%. The removed edges could be selected at random, as a single subpath, or several subpaths. You can try to propose some heuristic rules to remove “bad” nodes/edges, e.g. long edges or costly nodes. Such heuristics should be, however, randomized not completely deterministic. For example the probability of removal should depend on the length/cost.

As repair operator use the best greedy heuristic (including greedy-regret) from previous assignments.

The destroy-repair operators should be clearly described.

As the starting solution use random solution.

Implement two versions of LNS – using or not local search after destroy-repair operators. Use the best version of steepest local search. Always apply local search to the initial solution.

Computational experiment: Run each of the methods (with and without local search) 20 times for each instance. Use the average running time of MSLS from the previous assignment. Report also the number of iterations of the main loop.

Reporting results: Use tables as in the previous assignments. Add a table with the number of iterations of the main loop. Include results of MSLS, ILS, and the best greedy heuristic (including regret) an basic local search.

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
- Pseudocode is simplified to the required configuration of Local Search - that is: Steepest LS, Random starting solution, two edges exchange intra moves
- LS with candidate moves was used

Large-Scale Neighborhood Search (LSNS)

1. Generate initial solution (Local Search) - remember it as the best solution known so far
2. Calculate number of nodes to destroy in each destroy() operator, given specified "**percentage destroyed**" (used 25%) - "**nodes to destroy**"
2.Ad.1. Ceiling of (required number of nodes * **percentage destroyed** / 100)
3. Calculate costs of connecting each node to each other node in the problem
3.Ad.1. Additional cost of node + additional cost of destination node + edge length between node and destination node
4. Calculate "**neighborhood costs**" of all nodes, given "**neighborhood size**" (used 10)
4.Ad.1. Used as weight for probability of which nodes should be chosen in destroy() operator
4.Ad.2. Sum of (**neighborhood size**) first costs calculated in **3.**
5. While total running time of the algorithm is less than that of MSLS
5.1. Create destroyed solution by applying destroy() operator on the best solution known so far
5.1.1. Randomly choose (**nodes to destroy**) nodes from the solution, using their (**neighborhood costs**) as weights
5.1.2. Remove chosen nodes from the solution
5.2. Create repaired solution by applying repair() operator on the destroyed solution
5.2.Ad.1. Run Nearest Neighbor at any construction heuristic, starting from destroyed solution state
5.3. (depending on configuration) run Local Search on repaired solution
5.4. If objective function of repaired solution is lower, update best solution known so far
6. Return the best solution known so far

Results

Note: All best solutions were checked with the solution checker

Statistics

Method	**TSPA** avg (min - max)	**TSPB** avg (min - max)
NN at any	73173 (71179 - 75450)	45870 (44417 - 53438)
Local Search (steepest, greedy, two_edges)	71998 (70110 - 74270)	45171 (44140 - 51146)
Local Search (steepest, random, two_edges)	74222 (71725 - 78624)	48627 (45619 - 52245)
Multiple Start Local Search (MSLS)	72324 (71458 - 72825)	46801 (45825 - 47480)
Iterated Local Search (ILS)	69498 (69259 - 70321)	43858 (43568 - 44484)
Large-Scale Neighborhood Search (LSNS) - no LS	69935 (69230 - 71274)	44437 (44984 - 46112)
Large-Scale Neighborhood Search (LSNS) - with LS	69774 (69230 - 70258)	44373 (43550 - 45506)

Running times (s)

Method	**TSPA** avg (min - max)	**TSPB** avg (min - max)
NN at any	0.15 (0.14 - 0.19)	0.16 (0.15 - 0.53)
Local Search (steepest, greedy, two_edges)	0.08 (0.07 - 0.08)	0.08 (0.07 - 0.08
Local Search (steepest, random, two_edges)	0.22 (0.18 - 0.42)	0.22 (0.17 - 0.32)
Multiple Start Local Search (MSLS)	47.48 (42.77 - 53.01)	45.31 (44.61 - 45.89)
Iterated Local Search (ILS)	47.37 (47.26 - 47.51)	47.39 (47.26 - 47.54)
Large-Scale Neighborhood Search (LSNS) - no LS	47.29 (47.22 - 47.36)	47.28 (47.20 - 47.37)
Large-Scale Neighborhood Search (LSNS) - with LS	47.26 (47.20 - 47.32)	47.39 (47.18 - 47.54)

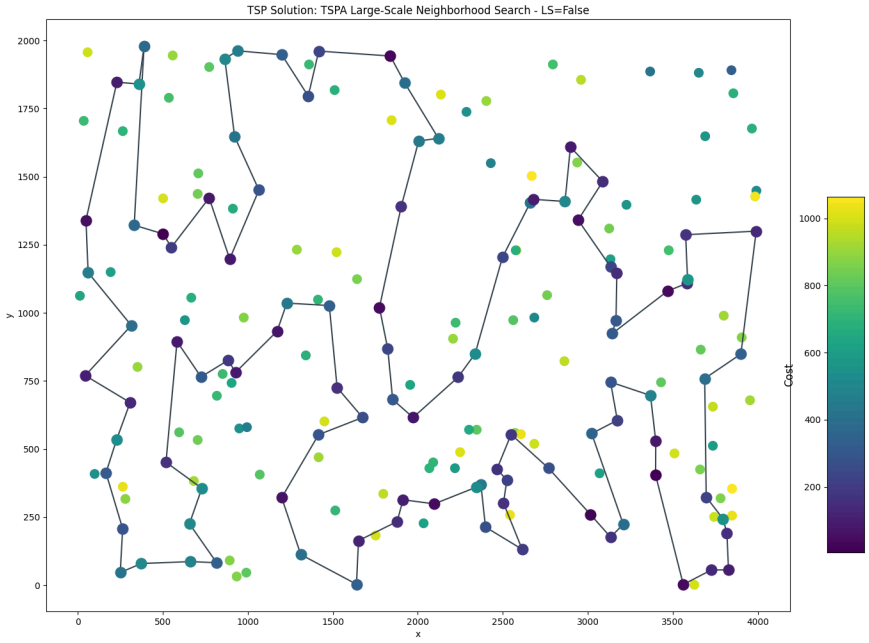
Number of Local Search / main loop runs

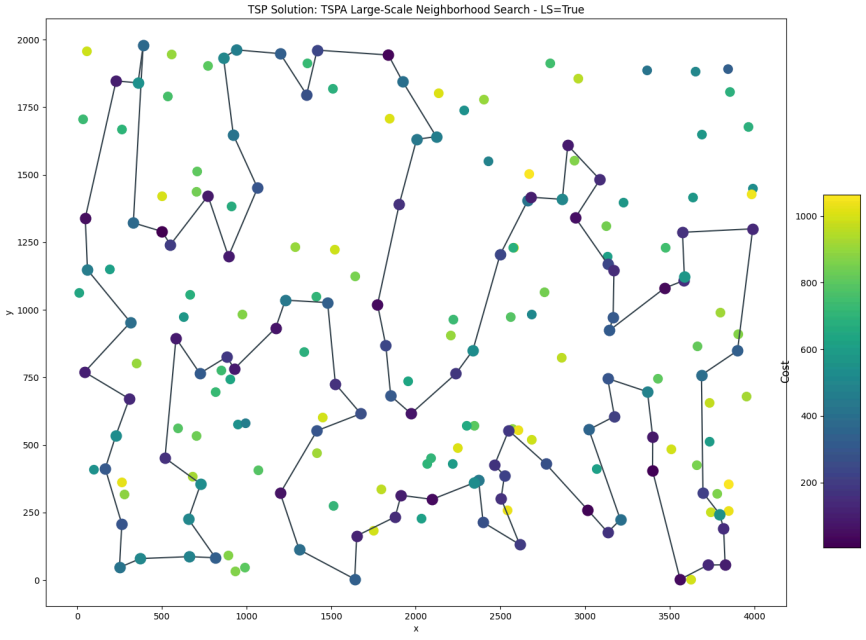
Method	**TSPA** avg (min - max)	**TSPB** avg (min - max)
Multiple Start Local Search (MSLS) *Number of LS runs*	200 (200 - 200)	200 (200 - 200)
Iterated Local Search (ILS) *Number of LS runs*	901 (815 - 951)	901 (875 - 940)
Large-Scale Neighborhood Search (LSNS) - no LS *Number of main loop runs*	678 (494 - 785)	720 (682 - 756)
Large-Scale Neighborhood Search (LSNS) - with LS *Number of main loop runs*	613 (592 - 622)	616 (593 - 630)

Best solutions

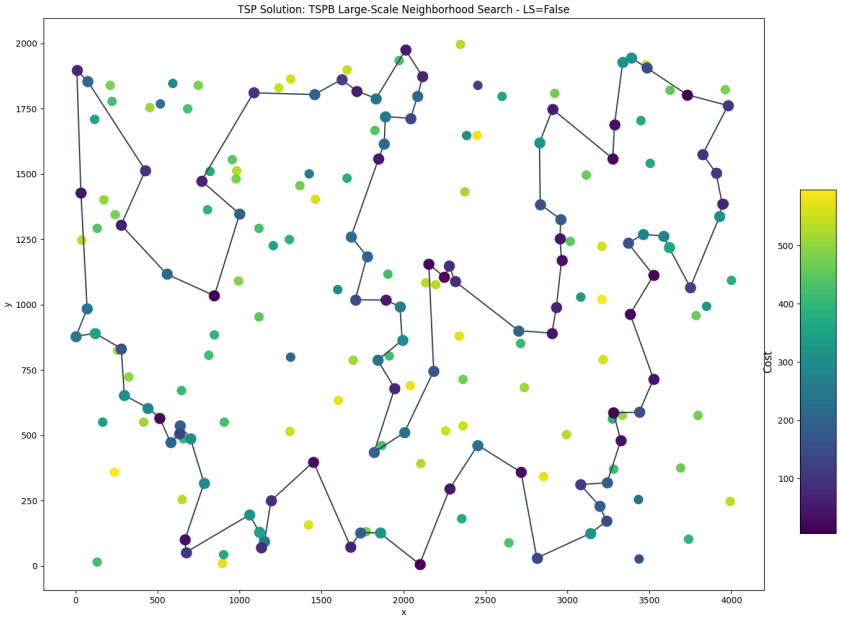
Note: additional cost is depicted using a color scale

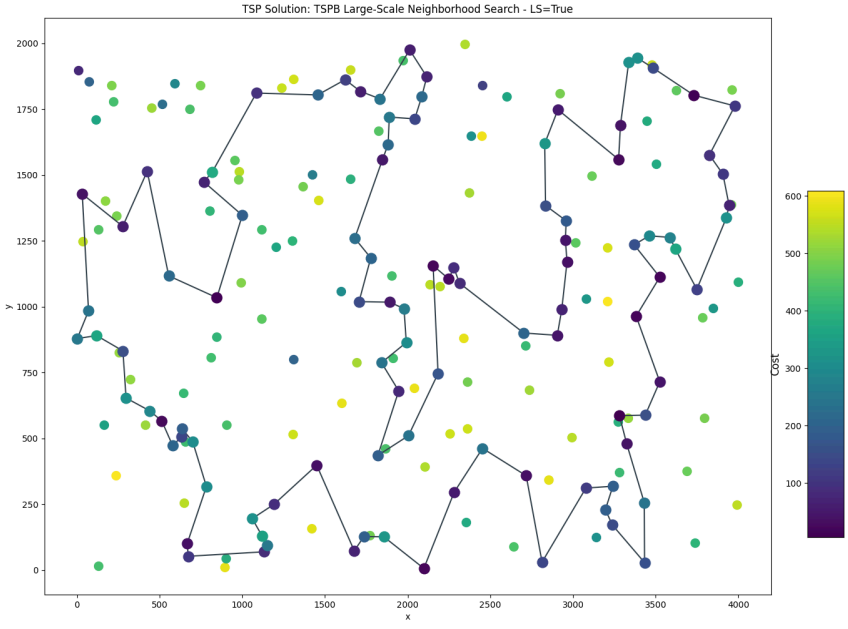
TSPA


TSP Solution: TSPA Large-Scale Neighborhood Search - LS=False
x
y
Cost


TSP Solution: TSPA Large-Scale Neighborhood Search - LS=True
x
y
Cost

TSPB


TSP Solution: TSPB Large-Scale Neighborhood Search - LS=False
x
y
Cost


TSP Solution: TSPB Large-Scale Neighborhood Search - LS=True
x
y
Cost

Conclusions

- Both LSNS configurations yield overall better results than any other method, except ILS
- Both LSNS may offer slightly better best results over ILS
- However, ILS is better on average, delivers relatively as good best results, and is more consistent (its max are significantly better than that of LSNS, almost as good as their averages)
- Applying LS after repair() operator leads to slightly better results, at the expense of slightly less main loop runs (and subsequently - less solutions checked)
- Applying LS on the final result from LSNS in config that does not use LS in the main loop could lead to at least slightly better results