Assignment 7: Large neighborhood search

Uladzislau Lukashevich 155671, Kiril Andrukh 162069
Source: link

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

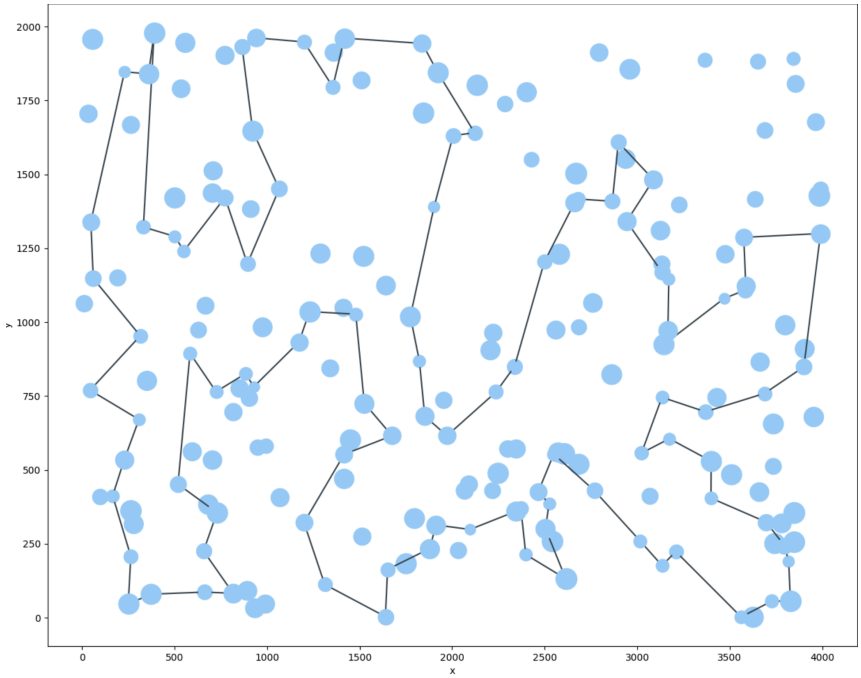
MSLS, ILS, and the best greedy heuristic (including regret) and basic local search. The outline of the report as previously.

Pseudo code

```
Generate initial solution using LS and mark it as the best

Calculate number of nodes to be destroyed in each destroy() operator (e.g.
25%)

Calculate costs of connecting each node to each other node in the problem
Calculate neighborhood costs of all nodes, used as weighted for
probability to choose nodes in destroy() operator

Function Destroy():
      Removes 25% of nodes probabilistically
      Probability based on neighborhood costs
      Neighborhood cost = sum of k-nearest neighbor costs (k=10)
      Higher cost nodes more likely to be removed

Repair:
      Nearest Neighbor Heuristic

While total running time is less than timeout:
      Create destroyed solution by applying destroy() to the best solution
      Create repaired solutions by using repair() on the destroyed
solution
      Run Local Search on repaired solution (optional)
      If objective function of the repaired solution is less:
            update the best solution

Return the best solution
```

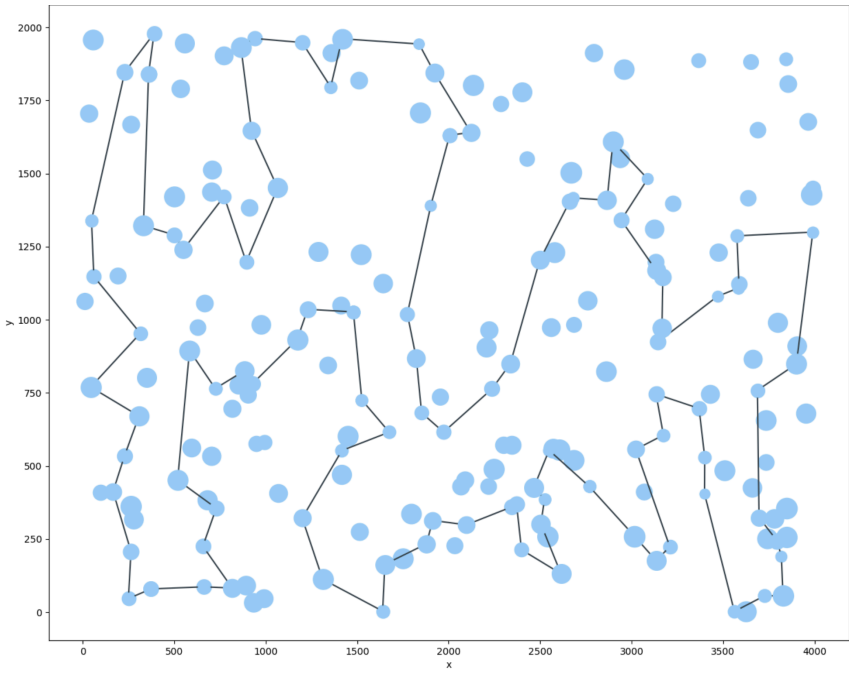
TSPA

Without LS



Best solution:
[159,193,41,139,115,46,68,140,93,117,0,143,183,89,186,23,137,176,80,79,63,94,124,148,9,62,102,144,14,49,178,106,52,55,185,40,119,165,90,81,196,179,57,129,92,145,78,31,56,113,175,171,16,25,44,120,2,152,97,1,101,75,86,26,100,53,180,154,135,70,127,123,162,133,151,51,118,59,65,116,43,42,184,35,84,112,4,190,10,177,54,48,160,34,181,146,22,18,69,108]

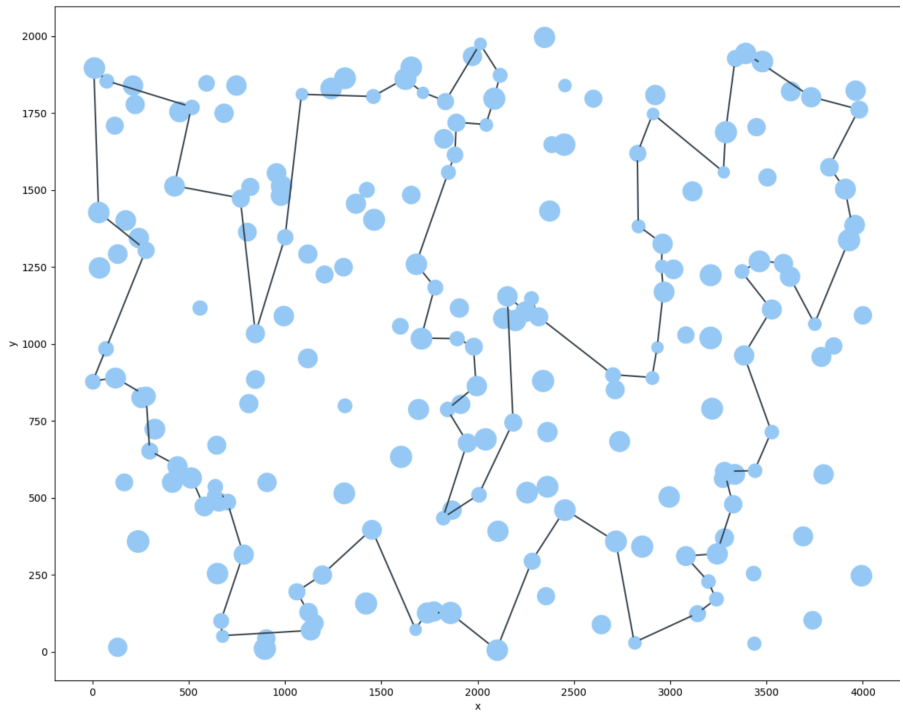
With LS



Best solution:
[181,146,22,18,108,69,159,193,41,139,115,46,68,140,93,117,0,143,183,89,186,23,137,176,80,79,63,94,124,148,9,62,102,144,14,49,178,106,52,55,185,40,119,165,90,81,196,31,56,113,175,171,16,78,145,179,57,92,129,25,44,120,2,152,97,1,101,75,86,26,100,53,180,154,135,70,127,123,162,133,151,51,118,59,65,116,43,42,184,35,84,112,4,190,10,177,54,48,160,34]

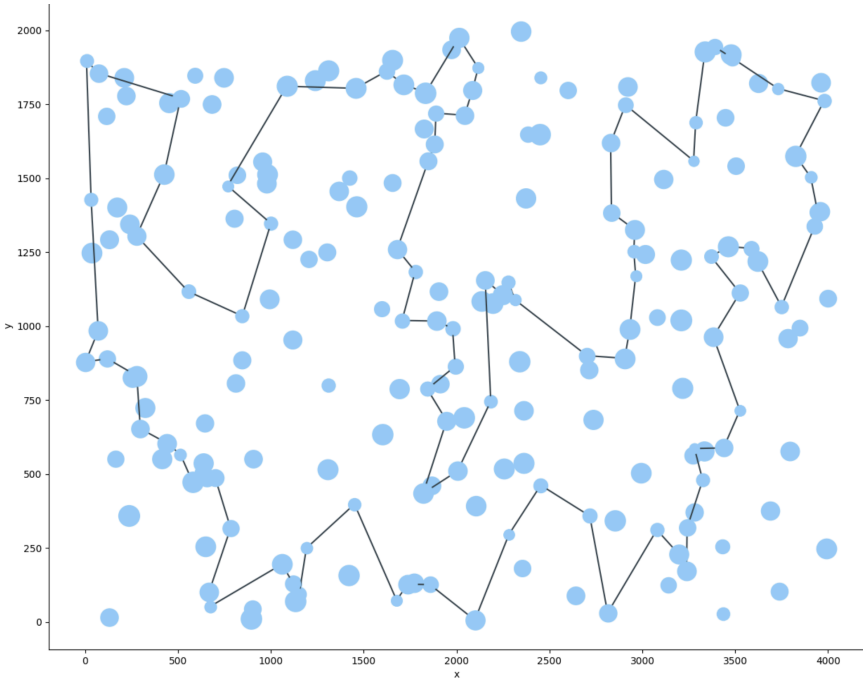
TSPB

Without LS



Best solution:
[147,6,188,169,132,70,3,15,145,13,195,168,139,11,138,33,160,144,104,8,21,82,111,29,0,109,35,143,106,124,62,18,55,34,170,152,183,140,4,149,28,20,60,148,47,94,66,179,22,99,130,95,185,86,166,194,176,113,103,163,89,127,165,187,153,81,77,141,91,61,36,177,5,45,142,78,175,80,190,136,73,54,31,193,117,198,156,1,16,27,38,135,63,40,107,133,122,90,121,51]

With LS



Best solution:
[183,140,4,149,28,20,60,148,47,94,66,179,22,99,130,95,185,86,166,194,176,113,103,127,89,163,187,153,81,77,141,91,61,36,177,5,78,175,142,45,80,190,136,73,54,31,193,117,198,156,1,16,27,38,63,40,107,133,122,135,131,121,51,90,147,6,188,169,132,70,3,15,145,13,195,168,139,11,138,33,160,144,104,8,21,82,111,29,0,109,35,143,106,124,62,18,55,34,170,152]

Comparison tables

Objective function values

Method	Dataset A	Dataset B
Random generation, edge, steepest	73855.835(70939-77610)	48296.625(45319-50992)
Multiple start local search	72010.48(70553-72972)	46477.23(45212-47381)
Iterated local search	70797.655(69875-72440)	45494.965(44070-47548)
Large Scale Neighborhood	69935(69230-71274)	44984(44437-46112)

Search without LS		
Large Scale Neighborhood Search with LS	69774(69230-70258)	44373(43550-45506)

Running times

Method	Dataset A	Dataset B
Random generation, edge, steepest	3.3 s	3.17 s
Multiple start local search	46.43 s	46.24 s
Iterated local search	46.42 s	46.23 s
Large Scale Neighborhood Search without LS	47.29 s	47.28 s
Large Scale Neighborhood Search with LS	47.26 s	47.31 s

Number of Main Loop / Local Search Runs

Method	Dataset A	Dataset B
Multiple start local search	200(200-200)	200(200-200)
Iterated local search	642.54(624-658)	645.46(593-666)
Large Scale Neighborhood Search without LS	678(494-785)	720(682-756)
Large Scale Neighborhood Search with LS	613(592-622)	616(593-630)

Conclusions

- Both LSN versions perform better than other methods in our case
- Applying LS after repair has slightly better results than not applying