Tabu search

1 WORK DURING THE LAB

1. Implement the algorithm **Tabu Search** for the *knapsack problem*. Test the algorithm for two problem instances, considering different parameter settings.
2. Read data for the Traveling Salesman Problem (TSP).
3. Generate a **greedy solution** for *TSP* and verify its quality.

Points for the work during the lab: **25p**

2 ASSIGNMENT A3

1. Implement the algorithm **Tabu Search** for solving *TSP*.
2. Perform experiments for 2 TSP instances selected from the list below: one instance from the list A1-A16 and one instance from the list B1-B29.
3. Compare results, considering different parameter settings for the algorithm.

Deadline to submit A3: **Lab 4**

Points for A3: **25p**

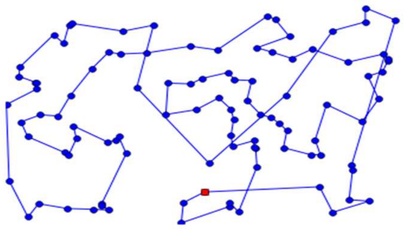
3 REQUIREMENTS

1. Source code (notebook) needs to be documented.
2. Algorithms have to be tested for several parameter values (sufficient to clearly determine performance).
3. Experiments must be performed for all available problem instances and results compared for diferent parameter settings.
4. Results of the experiments need to be saved in output files, indicating solution quality, parameter values used, number of runs.
5. A report should capture the following: problem definition, algorithm used (name, steps/pseudocode), parameter setting, comparative results of experiments, discussion of results.

4 TRAVELING SALESMAN PROBLEM (TSP)

It is a well-known combinatorial optimization problem with many applications (see http://www.math.uwaterloo.ca/tsp/apps/index.html).

Given a collection of N cities and the cost of travel between each pair of them, the **traveling salesman problem**, or **TSP** for short, is to find the cheapest way of visiting all of the cities and returning to your starting point. In the standard version we study, the travel costs are symmetric in the sense that traveling from city X to city Y costs just as much as traveling from Y to X.



The solution can be represented as permutation of size N. The cost of a solution is the sum of distances between cities in the order given by the permutation.

„The simplicity of the statement of the problem is deceptive – the TSP is one of the most intensely studied problems in computational mathematics and yet no effective solution method is known for the general case. Indeed, the resolution of the TSP would settle the P versus NP problem and fetch a $1,000,000 prize from the Clay Mathematics Institute.” [http://www.math.uwaterloo.ca/tsp/problem/index.html]

TSP Tutorial in Python: https://nbviewer.org/url/norvig.com/ipython/TSP.ipynb

5 TSP INSTANCES

TSP test data is available from:

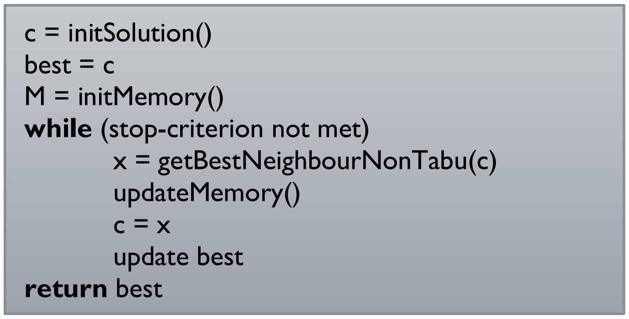
- [A] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
- [B] https://www.math.uwaterloo.ca/tsp/data/index.html

List of symmetric TSP instances:

No	Instance	Link
A1	eil101.tsp	http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/
A2	eil176.tsp	
A3	berlin52.tsp	
A4	bier127.tsp	
A5	kroA100.tsp	
A6	kroB100.tsp	
A7	kroC100.tsp	
A8	kroD100.tsp	
A9	kroE100.tsp	
A10	lin105.tsp	
A11	pr76.tsp	
A12	p107.tsp	
A13	pr124.tsp	
A14	rat99.tsp	
A15	st70.tsp	
National TSP		
B1	ar9152	https://www.math.uwaterloo.ca/tsp/world/countries.html
B2	bm33708	
B3	ca4663	
B4	ch71009	
B5	dj38	
B6	eg7146	
B7	fi10639	
B8	gr9882	
B9	ho14473	
B10	ei8246	
B11	it16862	
B12	ja9847	
B13	kz9976	
B14	lu980	
B15	mo14185	
B16	nu3496	
B18	mu1979	
B19	pm8079	
B20	qa194	
B21	rw1621	

B22	sw24978	
B23	tz6117	
B24	uy734	
B25	vm22775	
B26	wi29	
B27	ym7663	
B28	zi929	
B29	ro2950_geo.tsp	https://cadredidactice.ub.ro/ceraselacrisan/cercetare/

6 TABU SEARCH



```
c = initSolution()
best = c
M = initMemory()
while (stop-criterion not met)
        x = getBestNeighbourNonTabu(c)
        updateMemory()
        c = x
        update best
return best
```

The memory retains the number of iterations when certain moves to obtain a neighbour are tabu.

The stop criterion is given by a maximum number of iterations.