EC - Local Search
Assignment 3

Adam Korba - 151962 Łukasz Sztukiewicz - 151959

PROBLEM DESCRIPTION

We are given three columns of integers with a row for each node. The first two columns contain x and y coordinates of the node positions in a plane. The third column contains node costs. The goal is to select exactly 50% of the nodes (if the number of nodes is odd we round the number of nodes to
be selected up) and form a Hamiltonian cycle (closed path) through this set of nodes such that the sum of the total length of the path plus the total cost of the selected nodes is minimized.

The distances between nodes are calculated as Euclidean distances rounded mathematically to integer values. The distance matrix should be calculated just after reading an instance and then only the distance matrix (no nodes coordinates) should be accessed by optimization methods to allow
instances defined only by distance matrices.

PSEUDOCODE

Greedy localsearch
Generate initial solution x
repeat
for each y N(x) in a random order
    if f(y) > f(x) then
        x := **y**
until no better solution was found after checking the whole N(x)

Steepest version
Generate initial solution x
repeat
find the best solution y N(x)
    if f(y) > f(x) then
        x := **y**
until no better solution was found after checking the whole N(x)

SCORES

	TSPA			TSPB		
method	**AVERAGE of score**	MIN of score	MAX of score	**AVERAGE of score**	MIN of score	MAX of score
intra_edge_steepest_heuristic	**71590**	**70098**	**73287**	50673	47923	56800
intra_edge_greedy_heuristic	71608	70098	73508	50692	48082	56845
solve_weighted_regret_greedy_cycle	72100	70657	73345	49230	46323	52735
intra_node_steepest_heuristic	72289	70944	73656	50771	47979	56856
intra_node_greedy_heuristic	72290	70944	73656	50799	48033	56856
solve_greedy_cycle	72609	71488	74410	51302	48765	57324
intra_edge_greedy_random	73859	71388	78311	48414	45992	**50973**
intra_edge_steepest_random	73998	71046	79798	**48299**	**45728**	51300
solve_nn_any	75693	72941	77500	56191	51174	61245
solve_nn_first	85109	83182	89433	54390	52319	59030
intra_node_greedy_random	85943	78547	92472	60773	53737	68573
intra_node_steepest_random	88365	80484	95289	62953	53870	71808
solve_regret_greedy_cycle	115510	106194	124688	73307	68661	78823

EXECUTION TIMES

	TSPA			TSPB		
method	**AVERAGE of time ms**	MIN of time ms	MAX of time ms	**AVERAGE of time ms**	MIN of time ms	MAX of time ms
solve_nn_first	**0.03**	**0.03**	**0.09**	0.03	0.03	0.05
intra_node_steepest_heuristic	0.43	0.10	1.11	0.86	0.45	1.50
intra_edge_steepest_heuristic	0.45	0.15	0.97	0.45	0.23	0.72
solve_nn_any	0.59	0.57	0.68	0.63	0.59	0.82
solve_greedy_cycle	2.13	2.00	3.10	2.08	1.96	2.49
intra_node_greedy_heuristic	3.57	1.05	10.05	6.14	3.57	12.38
intra_edge_greedy_heuristic	6.18	1.45	12.85	6.30	3.28	13.25
intra_edge_steepest_random	6.56	5.75	8.05	5.95	5.13	6.72
intra_node_steepest_random	14.10	11.15	17.55	13.78	11.11	22.96
solve_weighted_regret_greedy_cycle	37.66	36.36	42.57	38.13	36.67	40.33
solve_regret_greedy_cycle	38.00	36.94	52.53	38.56	37.32	42.27
intra_edge_greedy_random	170.11	145.46	223.47	158.16	140.42	188.50
intra_node_greedy_random	171.58	130.82	228.00	161.65	130.12	190.37

NUMBER OF ITERATIONS

We were surprised by execution times so we decided to also inspect number of iterations for each local search method

method	TSPA			TSPB		
	AVERAGE of iter	MIN of iter	MAX of iter	AVERAGE of iter	MIN of iter	MAX of iter
intra_node_steepest_heuristic	**5**	**1**	**12**	**10**	**5**	17
intra_node_greedy_heuristic	**5**	**1**	18	11	6	21
intra_edge_steepest_heuristic	9	3	17	**10**	**5**	**16**
intra_edge_greedy_heuristic	11	2	26	11	**5**	27
intra_edge_steepest_random	134	121	155	134	117	151
intra_node_steepest_random	155	123	193	154	125	186
intra_node_greedy_random	415	322	517	418	342	489
intra_edge_greedy_random	418	366	475	417	370	467

CONCLUSIONS

In the end local search methods allowed us to get slightly better solutions than previously used greedy methods, however not all methods were equal.

- Intra move methods: edge exchange has proven to be consistently better than node exchange as it allows for introducing more sophisticated changes
- Steepest vs greedy: steepest performed slightly better than greedy in most cases, surprising result was that greedy was much slower, it might be due to some compiler magic that is done on the steepest side, as the algorithm is much simpler *(less branches, no randomization, no additional memory allocation)*
- Starting conditions: Is is self-evident that starting with some good heuristic solution is very beneficial for local search, results were much better when starting with **greedy cycle** solution and execution times were reduced dramatrically.

CODE

https://github.com/BbqGamer/TSP/tree/master/tsp/localsearch