Report 3

Local Search

Evolutionary Computation

Zuzanna Buchnajzer

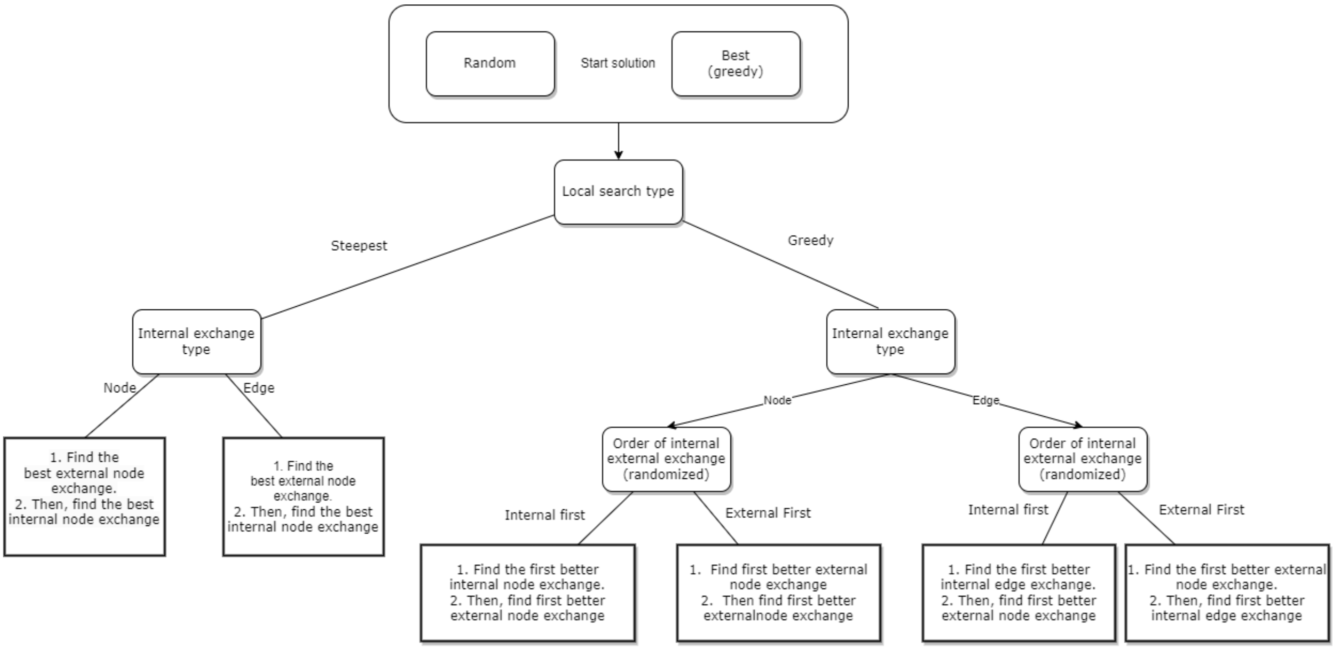
Overview of local search function:

Parameters:

- **data**: from data given in this task we derived distance_matrix, and cost_list as static variables of DatasetD.
- **start_solution**: The function initializes the starting solution from the **start_index**. If start_solution is "random", it shuffles the indices and selects half, then computes the total cost. If it's "best" utilizes a greedy cycle construction heuristic.
- **alg_type**: Specifies the type of local search algorithm, either "steepest" or "greedy".
- **neighbors**: Specifies the type of neighbor search strategy only in terms of internal search, either "nodes" or "edges". External search is obtain only using nodes.

The function concludes either upon reaching 500 iterations or when no better solution is found, whichever comes first.

Actions performed for each of the parameters.



Random
Start solution
Best (greedy)
Local search type
Steepest
Greedy
Internal exchange type
Node
Edge
1. Find the best external node exchange. 2. Then, find the best internal node exchange
1. Find the best external node exchange. 2. Then, find the best internal node exchange
Internal exchange type
Node
Edge
Order of internal external exchange (randomized)
Internal first
External First
1. Find the first better internal node exchange. 2. Then, find first better external node exchange
1. Find first better external node exchange 2. Then find first better externalnode exchange
Order of internal external exchange (randomized)
Internal first
External First
1. Find the first better internal edge exchange. 2. Then, find first better external node exchange
1. Find the first better external node exchange. 2. Then, find first better internal edge exchange

Type of local search:

Greedy randomization:

- random_lista_indexes is a list of indices corresponding to the visited noded. This list of indexes is later is shuffled.
- After the randomization steps, the algorithm:
  - iterates over the shuffled indices (random_lista_indexes) and shuffled unvisited nodes for “**intra**” exchange.
  - iterates over the shuffled indices (random_lista_indexes) twice for “**inter**” exchange.

Results

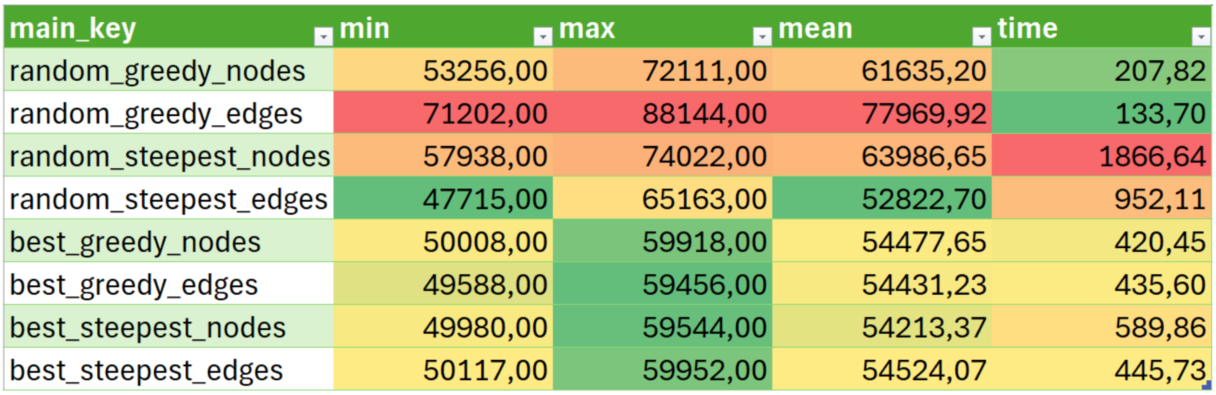
The results are obtained by aggregating the total_cost of algorithms starting from each possible vertex. The time is the sum of these 200 algorithm runs for a given configuration in seconds.

Quality of result

The best solution was achieved by the algorithm that started with a random solution, traversing its entire vicinity using only edge swaps, with a time only about twice as long as algorithms starting from the best solution. Those algorithms that utilize non-random initial solutions consistently achieve very good and stable results regardless of the configuration of the remaining parameters.

Time

Generally, algorithms operating on edges had shorter times than those iterating over nodes. Noticeably, algorithms starting with a random solution and operating in a greedy manner had the shortest termination times, yet their results ranked in the bottom 3. Interestingly, the random_steepest_nodes algorithm repeatedly exceeded the average time obtained by the other algorithms, yet its results were the second worst among all.



main_key	min	max	mean	time
random_greedy_nodes	53256,00	72111,00	61635,20	207,82
random_greedy_edges	71202,00	88144,00	77969,92	133,70
random_steepest_nodes	57938,00	74022,00	63986,65	1866,64
random_steepest_edges	47715,00	65163,00	52822,70	952,11
best_greedy_nodes	50008,00	59918,00	54477,65	420,45
best_greedy_edges	49588,00	59456,00	54431,23	435,60
best_steepest_nodes	49980,00	59544,00	54213,37	589,86
best_steepest_edges	50117,00	59952,00	54524,07	445,73