Traveling salesperson problem

Homework 1.

EOA

CTU, FEL

David Koleckar, winter 21/22

Running the application

Program can be either run as script by running '*model.py*' for console output or by running '*main.py*' which starts the full graphical user interface of the program.

Modules

model.py: Represents algorithm and it's parameters.

local_search.py: Implements exhaustive 2-opt, 3-opt and first-improving local search with variable perturbation (either swap2 or reverse_subsequence)

evolutionary_algorithm.py: Basic evolution pipeline:
parent selection(truncate/tournament) → *breed*:*(crossover*→*mutate)* → *population replacement*
with multiple changeable parameters and operators.

operators.py: Implements various crossover and mutation/perturbation operators.

crossovers:

- *OX2* (Order Xover 2),
- *PMX* (Partially Mapped Xover),
- *ERX* (Edge recombination Xover)

mutations:

- *reverse_sub* = reverse sub-sequence of length *k*
- *swap2* =swap two genes in permutation (yields random k-opts)

function_tools.py Contains all other and helper functions and tools.

Gathering statistics by running algorithm multiple times.

Population initialization:
- '*random*' = initialize population of given size with random solutions.
- '*nearest_neighbour* = constructive heuristics, choosing the best (distance) available neighbour, deterministic.
-

controller.py: Represents controller in MVC (model-view-controller), one main GUI thread and two helper threads. Worker – running model, ViewUpdater – live plotting the best solution and fitness in time(fitness calls).

view.py: Implements the application UI. Using *PyQt5*. Live plots use *pyqtgraph* package.

plotting_matplotlib.py, *plotting_pyqtgraph.py*: Plotting the graphs running the application from console or GUI respectively.

Dependencies:

- *PyQt5*
- *pyqtgraph*
- *matplotlib*
- *numpy*

Graphical user interface

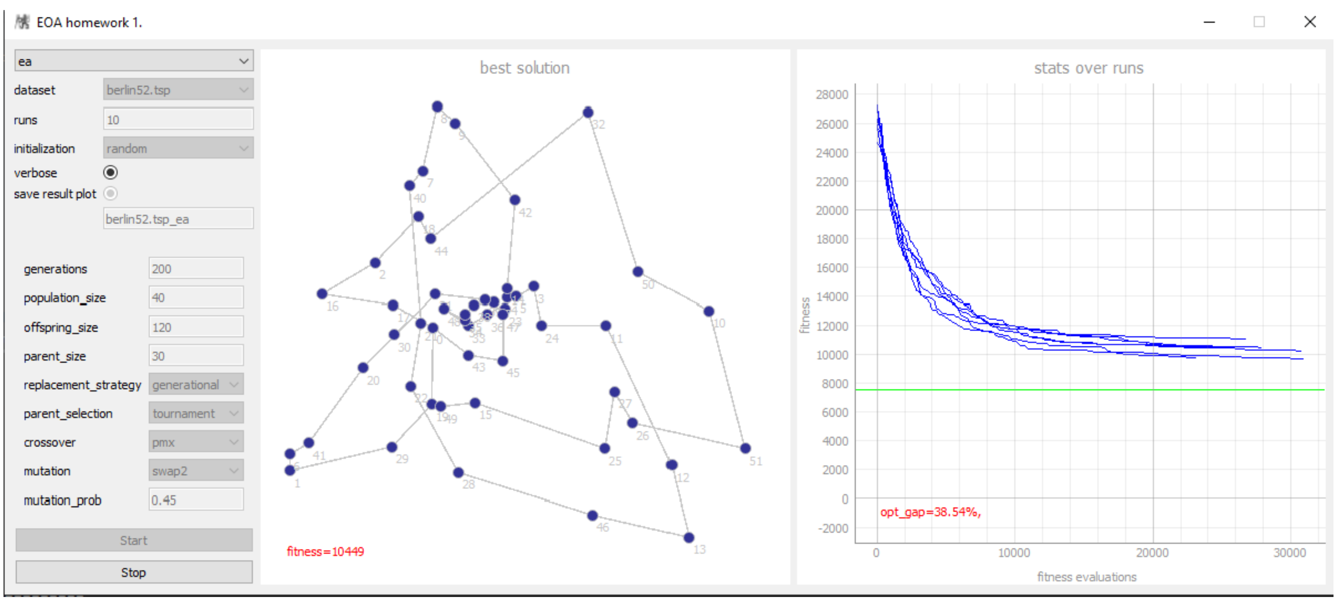


Image 1. Graphical user interface of the application.

The left panel allows to set parameters of the model. User can choose algorithm, dataset, etc. Setting verbose, prints run-time info/changes to the console.

Middle pannel shows at real-time the best found solution and it’s fitness.

Right pannel plots the real-time progress of actual run of the chosen model and the actual optimum gap.

Start the search by clicking Start button.

Model parameters

```
self.available_algorithms = {
                          "ea": evolutionary_algorithm,
                          "ls": local_search,
                          "opt-2": opt_2_best_improving,
                          "opt-3": opt_3_best_improving
                          }
```

Code snippet 1: Available algorithms.

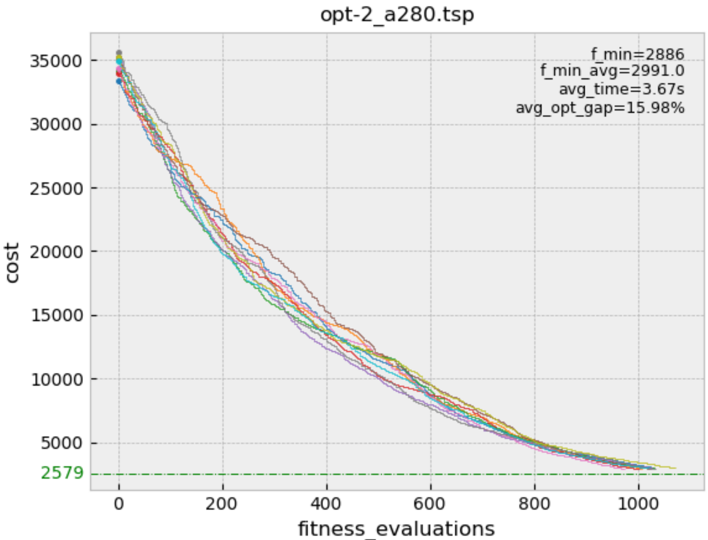
```
self.optimums = {"gr17.tsp": 2085, "gr21.tsp": 2707, "gr24.tsp": 1272, "gr48.tsp": 5046,
             "hk48.tsp": 11461,
             "si175.tsp": 21407, "si535.tsp": 48450, "si1032.tsp": 92650,
             "swiss42.tsp": 1273,
             "berlin52.tsp": 7542,
             "a280.tsp": 2579}
```

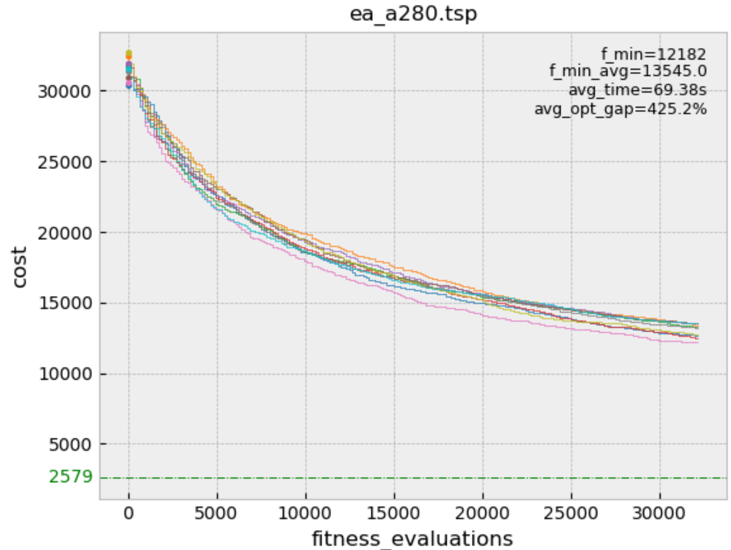
Code snippet 2: Prepared datasets (with their optimums).

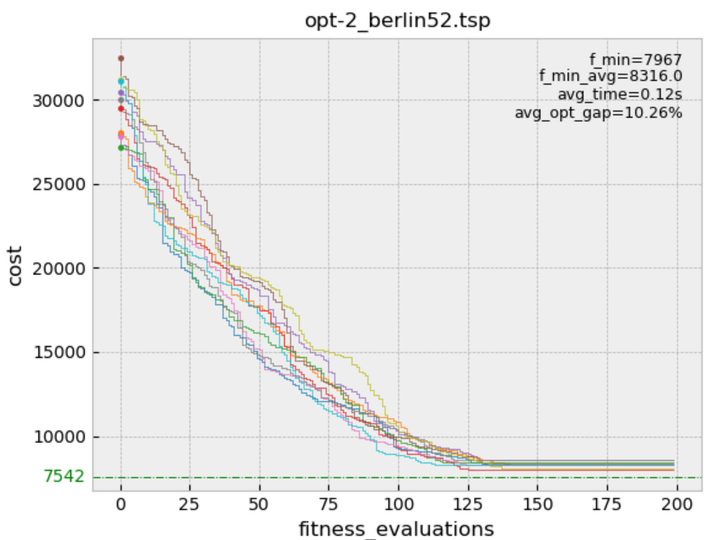
```
self.params_general = {'algorithm': self.available_algorithms[self.algorithm_name],
                  'algorithm_name': self.algorithm_name,
                  'generations': 200,
                  'solution_size': self.graph.dimension,
                  'runs': 10,
                  'dataset_name': self.dataset_name,
                  'optimum': self.optimums[self.dataset_name],
                  'plot_solutions': False,
                  'verbose': True,
                  'save_result_plot': True,
                  'save_results_file_name': self.dataset_name + "_" + self.algorithm_name}

self.params_ea = {'population_size': 40,
              'offspring_size': 120,
              'parents_size': 30,
              'initialize_solution_func': initialize_solution_permutation,
              'replacement_strategy': "generational",
              'parent_selection': "tournament",
              'crossover': "pmx",
              'mutation': "swap2",
              'mutation_prob': 0.45,
              }

self.params_ls = {'initialize_solution_func': initialize_solution_permutation,
              'perturbation_operator': mutate,
              'distances': get_distances(self.graph),
              'ls_stype': opt_2_best_improving,
              }
```

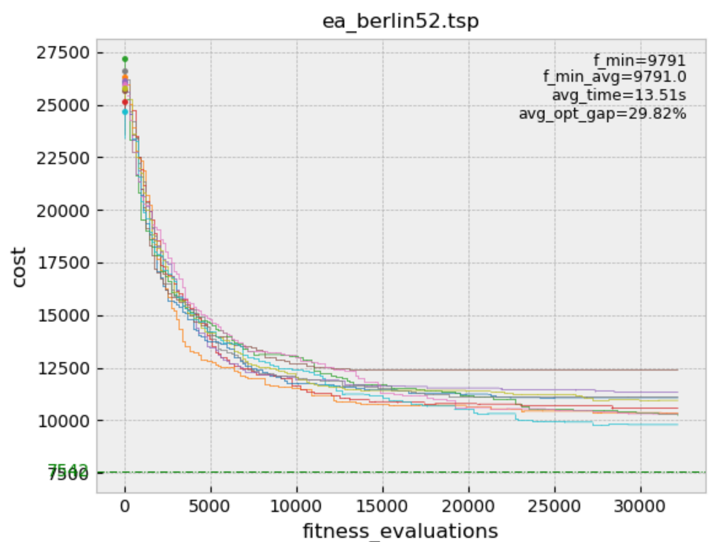
Code snippet 3: Model parameters – default setting.

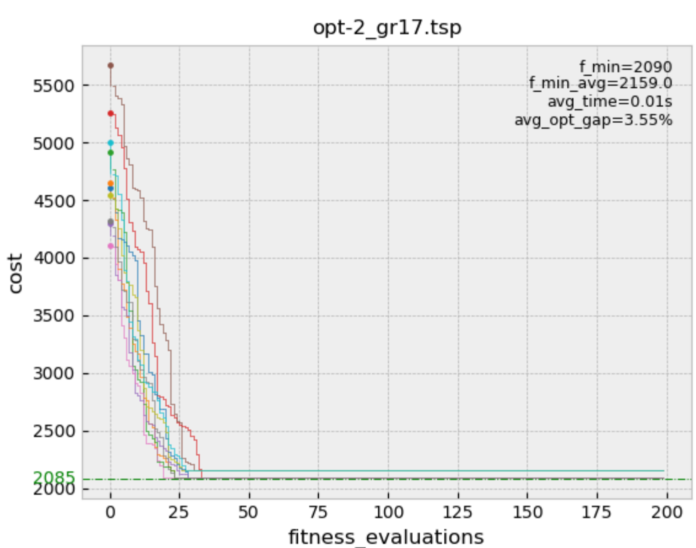
Graphs (10 runs)

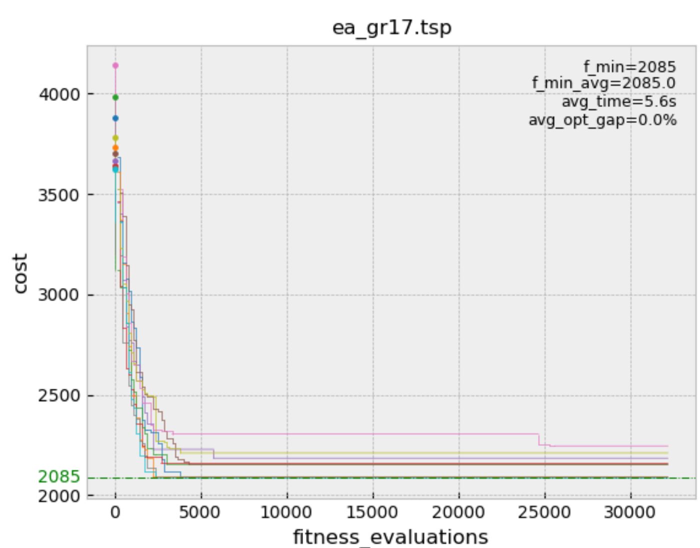


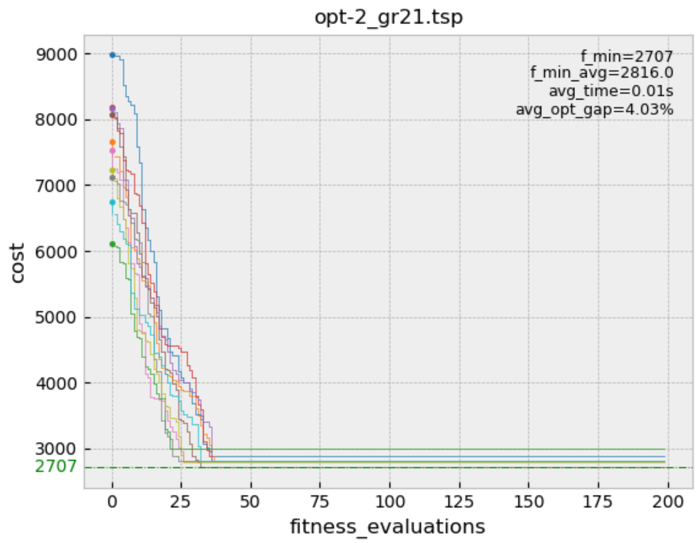


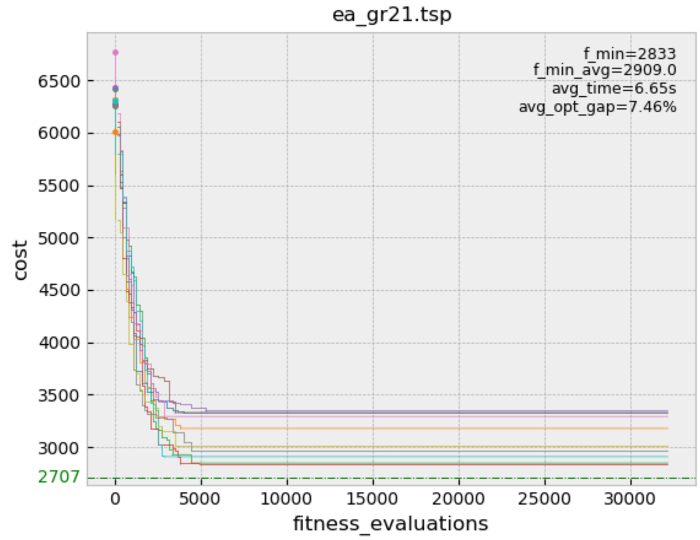


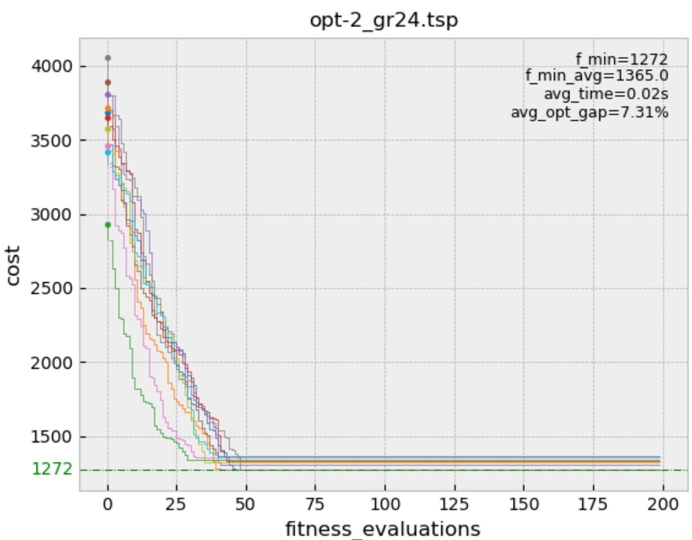


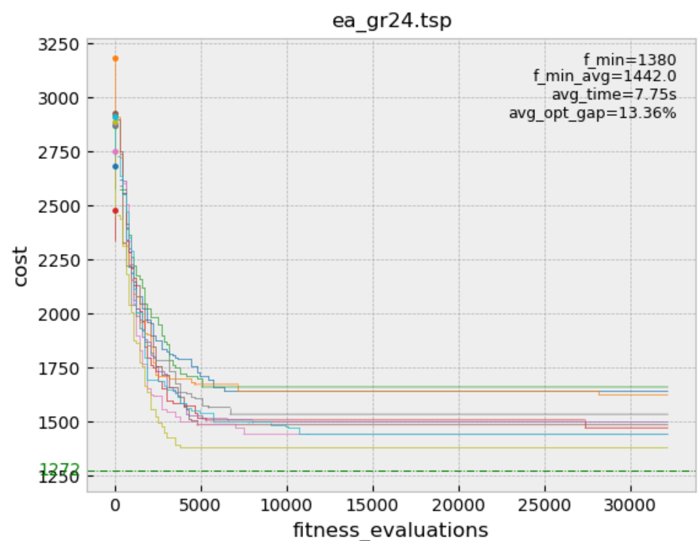


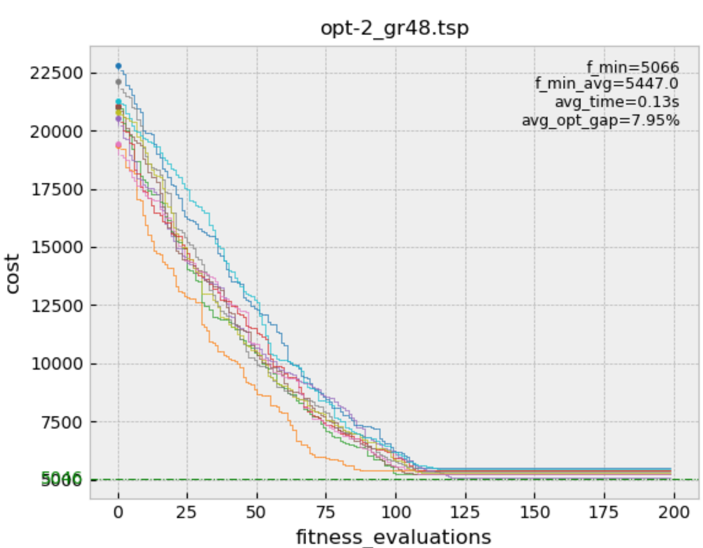


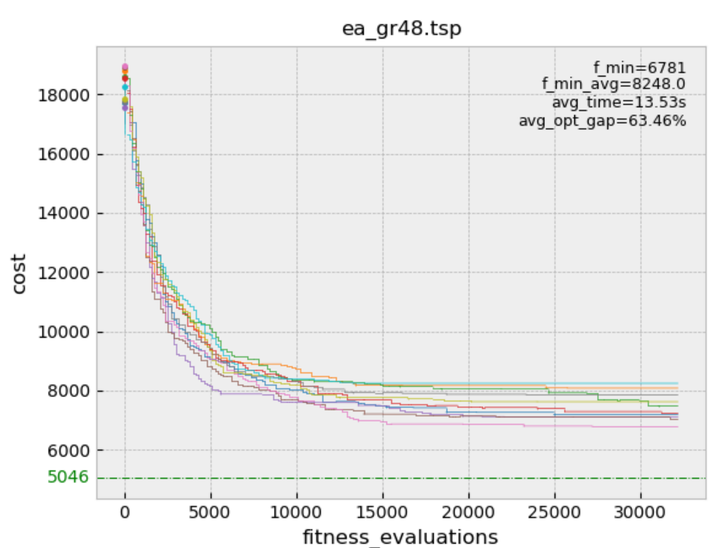

opt-2_gr21.tsp
f_min=2707
f_min_avg=2816.0
avg_time=0.01s
avg_opt_gap=4.03%
cost
fitness_evaluations
2707

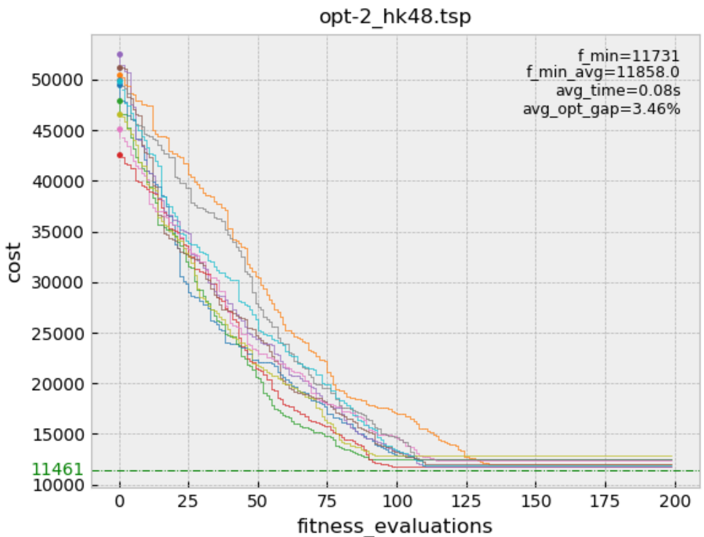

ea_gr21.tsp
f_min=2833
f_min_avg=2909.0
avg_time=6.65s
avg_opt_gap=7.46%
cost
fitness_evaluations
2707

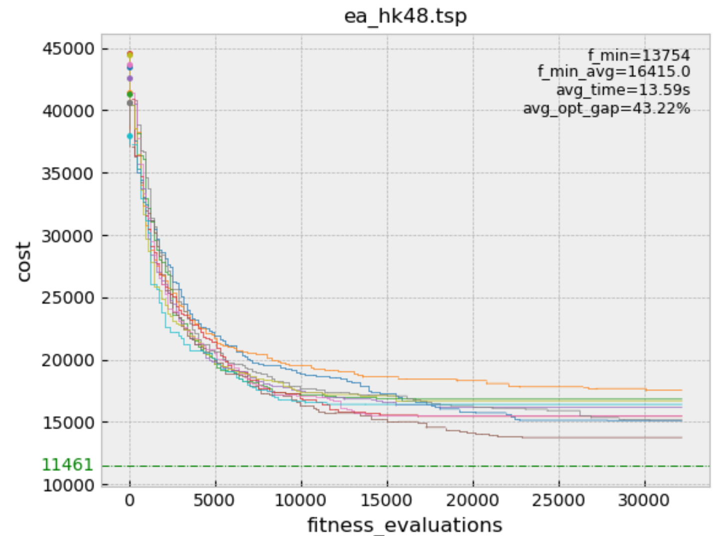

opt-2_gr24.tsp
f_min=1272
f_min_avg=1365.0
avg_time=0.02s
avg_opt_gap=7.31%
cost
fitness_evaluations
1272

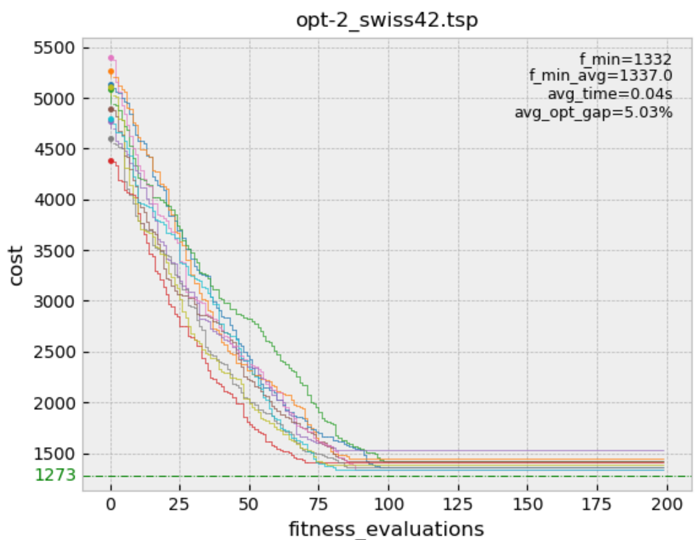

ea_gr24.tsp
f_min=1380
f_min_avg=1442.0
avg_time=7.75s
avg_opt_gap=13.36%
cost
fitness_evaluations
1272

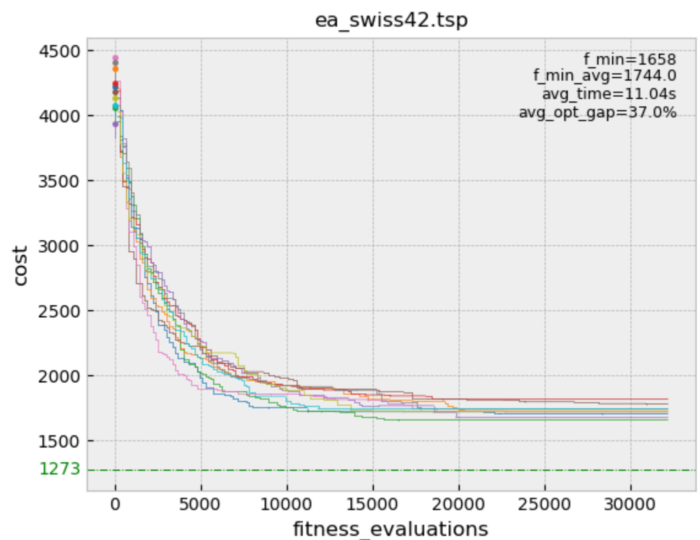

opt-2_gr48.tsp
f_min=5066
f_min_avg=5447.0
avg_time=0.13s
avg_opt_gap=7.95%
cost
fitness_evaluations
5046

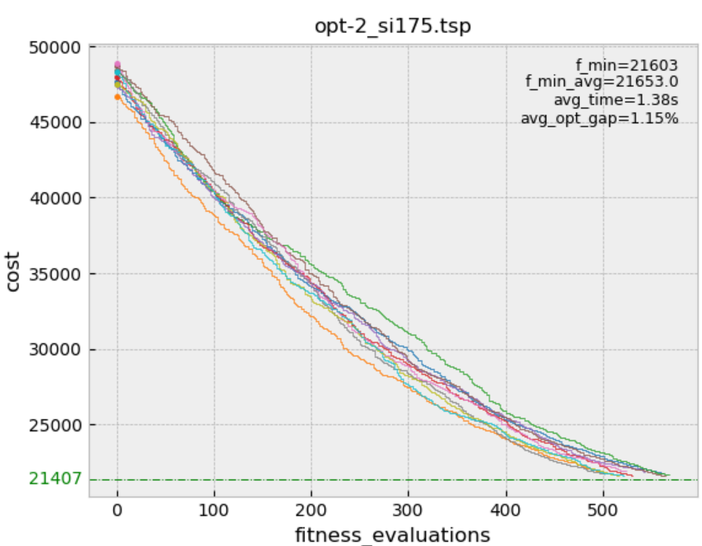

ea_gr48.tsp
f_min=6781
f_min_avg=8248.0
avg_time=13.53s
avg_opt_gap=63.46%
cost
fitness_evaluations
5046

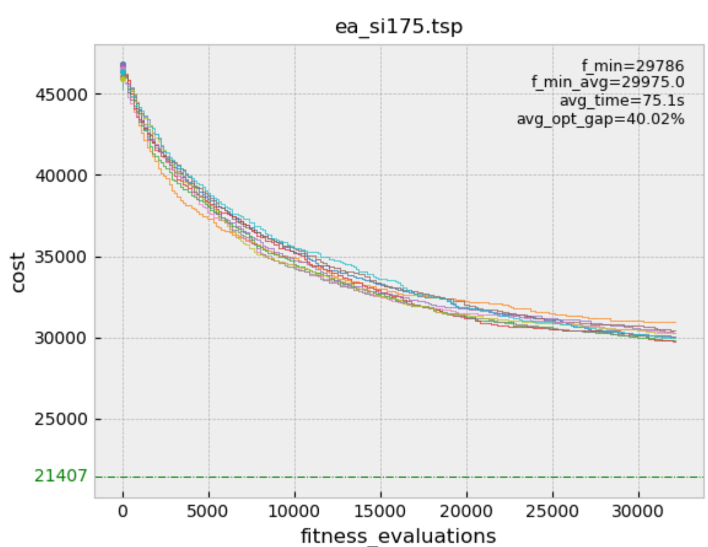

opt-2_hk48.tsp
f_min=11731
f_min_avg=11858.0
avg_time=0.08s
avg_opt_gap=3.46%
11461
cost
fitness_evaluations


ea_hk48.tsp
f_min=13754
f_min_avg=16415.0
avg_time=13.59s
avg_opt_gap=43.22%
11461
cost
fitness_evaluations


opt-2_swiss42.tsp
f_min=1332
f_min_avg=1337.0
avg_time=0.04s
avg_opt_gap=5.03%
1273
cost
fitness_evaluations


ea_swiss42.tsp
f_min=1658
f_min_avg=1744.0
avg_time=11.04s
avg_opt_gap=37.0%
1273
cost
fitness_evaluations


opt-2_si175.tsp
f_min=21603
f_min_avg=21653.0
avg_time=1.38s
avg_opt_gap=1.15%
21407
cost
fitness_evaluations


ea_si175.tsp
f_min=29786
f_min_avg=29975.0
avg_time=75.1s
avg_opt_gap=40.02%
21407
cost
fitness_evaluations

Results

The local search with 2-opt wins over evolutionary algorithm on all 10 tested datasets in terms of both speed of convergence and found optimum. The 3-opt in terms of optimality beats 2-opt but has cubic time complexity in number of nodes instead quadratic of the 2-opt, which is quite significant for the larger datasets.

Notes

Loading datasets: using *tsplib95* package facilitating datasets input.
For the bigger datasets I was not able to plot the solutions. 10 runs on datasets with ~1000 nodes was to much for my laptop.

Improvements

GUI not fully debugged, possibility of crashes in some situations.

Implement more algorithms/operators.

EA pipeline and operators could be speed/memory optimized.