TU Wien
Institute of Logic and Computation
Algorithms and Complexity Group

Heuristic Optimization Techniques, WS 2025

Programming Exercise: Assignment 2

v1, 29-08-2025

Work on the problem specified in the separate "Problem Description and General Information" document within your programming exercise group. You are encouraged to split tasks evenly, but everybody has to understand and to be able to explain all the concepts involved, your overall implementation, and the submitted report. Hand in your complete report, submit the solutions for at least three of the instances in the competition, and upload your source code via TUWEL by **Sunday, 18. 01. 2026, 23:55 Uhr**. For questions please contact us via `heuopt@ac.tuwien.ac.at`

Competition: The three student teams that will have uploaded the best solutions to a selection of *competition instances* in TUWEL by the aforementioned deadline will be awarded a small prize! Submissions are evaluated according to the sum of the fractions of obtained solution values over the overall best solution values over the competition instances. Upload date/time serves as tie breaker. Your rank in the competition has no influence on the final mark you receive for the course.

The second programming assignment is to **develop more advanced metaheuristics** for the *Selective Capacitated Fair Pickup and Delivery Problem* (SCF-PDP) in your preferred programming language, and to perform a **more advanced parameter tuning and statistical comparison**. You can build upon everything you developed for the first programming exercise.

The tasks for this exercise are:

1. Design and implement **two** of the following approaches to solve the SCF-PDP.
   - Evolutionary/Genetic Algorithm.
   - Ant Colony Optimization.
   - (Adaptive) Large Neighborhood Search
   - Some other hybrid algorithm: You can (and are encouraged to) build upon the algorithms you implemented so far, but the algorithm has to involve something substantially new.
2. Further investigate how the problem changes when the fairness measure changes. Report how the objective value and the number of stops over the routes change. Replace $J(R)$ in the objective function with two other fairness measures. You may develop your own fairness measure but can also use the two measures described below.
   - max-min fairness:
   $$M = \frac{\min_{k \in K} d(R_k)}{\max_{k \in K} d(R_k)}$$
   - Gini coefficient (inverted):
   $$G = 1 - \frac{\sum_{k' \in K} \sum_{k'' \in K} |d(R_{k'}) - d(R(k'')|}{2n_K \sum_{k' \in K} d(R_{k'})}$$

   For further tasks use Jain fairness again.
3. Perform a more advanced automated parameter tuning of the parameters of each of your two algorithms of Task 1 with the aim to find as good as possible solutions within limited time. Use the **tuning** instances to do so, leaving the test set untouched. Do so separately for each size of instances you can solve in reasonable time. Are you able to find solutions for all instance sizes? Get inspired by the possibilities stated in the slides regarding the tuning of metaheuristics. Recall that you have access to the AC group's cluster in case you want to perform computationally more demanding tuning runs, see the general information PDF. Report the found parameters and the procedure to obtain them.

4. Run experiments on the test instances provided in TUWEL using the better of the two tuned solution approaches. If the test results were inconclusive, choose one which you expect to be more promising and argue why. If not already done perform the same experiments on the most promising algorithm from assignment 1. Up to which instance size can you solve the instances in a reasonable time?

5. Test whether there is a significant difference between the two best performing algorithms from assignment 1 and 2, concerning the quality of the best found solutions on the **test** instances. You may use the statistical testing Jupyter notebook from the supplementary material. Include plots depicting the variance of your solutions in your report.

6. Write a concise report containing the description of your solution approaches, the applied parameter tuning method with corresponding results, and the experimental results. Include plots depicting the achieved solution quality over time, iterations, or generations on selected instances.

At the end: Reflect on your work. Which experiences do you take away from the programming assignments?

We hope you enjoy these tasks and wish you success!