Large Scale Optimization for Transport & Mobility

Assignment 2

Deadline: Wednesday October 15, 17:00

Rolf van Lieshout

1. This is a team assignment with two students per team. It is **not** allowed to collaborate with other teams (helping each other with small coding questions is allowed). Copying code or text from other teams is considered plagiarism. If you use online sources (such as ChatGPT, Stack Overflow, but also Python libraries), indicate this clearly in the comments of your code.
2. Test your implementations on small instances you constructed yourself, and verify that the results are correct. It is often good practice to write a little bit of verification code to verify the correctness of your solutions.

Solving a Rich Vehicle Routing Problem using Adaptive Large Neighborhood Search

In this assignment, you will solve the Pickup and Delivery Problem with Time Windows (PDPTW) by developing an Adaptive Large Neighborhood Search heuristic (ALNS). The PDPTW is the problem to transport goods from given pickup locations to delivery locations, such that time windows and vehicle capacities are respected, and the total transportation distance is minimized. Formally, we are given a number of requests that should be satisfied, where each request is characterized by (i) a pickup location, (ii) a delivery location, (iii) a time window for the pickup, (iv) a time window for the delivery, (v) a service time for the pickup, (vi) a service time for the delivery and (vii) the size of the request. In addition, we are given a depot from which all vehicles must start and end, the capacity of vehicles, and the distances between all locations. In this assignment, it is assumed that the number of available vehicles is nonrestrictive.

The Template

You do not have to programme the ALNS algorithm from scratch. We have prepared code for the structure and basic functionalities of the program, which you can download via Canvas.

For clarity, the code has been distributed over multiple files. The *zip-file* contains the following files:

- **`Problem.py`:** the Python file that contains classes to model an instance of the PDPTW. It contains a class for a request, a class for a location, and a class for representing a problem instance. Make sure that you understand the attributes of each class and what they represent.
- **`Route.py`:** the Python file that contains the class that represents a route taken by a vehicle. The class contains methods to check the feasibility of the roure and to compute the distance of the route. In addition, there is a method that inserts a request into the route, which will be used by the operators of the ALNS.
- **`Solution.py`:** the Python file that represents a solution to the PDPTW. In ALNS, we always move from one solution to another, so this is an important class. This class also contains the destroy and repair operators of the ALNS algorithm.
- **`ALNS.py`:** the Python file that contains the actual ALNS class. The `execute` method runs ALNS. In addition, this file contains a class with all parameters.
- **`Main.py`:** the Python file that reads in the data from a specific instance and calls the ALNS algorithm. If you run this file, you see the output of the algorithm.
- The folder "Instances": a folder with a set of benchmark instances of different sizes.

Before you continue with the actual assignment, take your time to familiarize yourself with the code. You do not need to understand each line of code, but make sure you understand the global structure of each class and its methods.

Your Task

In the current implementation, there is only one destroy and one repair operator. In addition, new solutions are only accepted if they are the best found solution so far, causing the algorithm to quickly get stuck in local minima. Therefore, your task is to improve the ALNS algorithm. Below, you can find a number of improvement suggestions. However, it is up to you to decide which improvements you want to implement.

- Implement additional destroy and operators, either from the literature (see references) or ones you come up with yourself (up to 1 point per operator, with a maximum of 5)
- Make the ALNS *adaptive* by implementing and updating weights according to which the operators are selected (1 point)
- Embed the search in a simulated annealing metaheuristic by adjusting the method `checkIfAcceptNewSol` (1 point)
- Tune the parameters (2 points)
- Implement any other improvements to the ALNS, e.g. by making the code more efficient or implements an extension, e.g. electric vehicles (up to 3 points)

Submission Requirements

Summarize your findings in a report of at most 10 pages (font size = 11pt, margin = 1 inch). Your report should include

1. a short introduction that states the problem, summarizes the solution approach and the main results, (1 point)
2. a methodology section that discusses the developed algorithm and motivates the design choices, (4 points)
3. a results section that presents and discusses the results of the computational experiments that you conducted, (3 points)
4. a short conclusion that summarizes your research and discusses its limitations and how your algorithms/implementation could be improved, (2 points)
5. and an appendix that contains your Python code (the appendix does not count towards the page limit).

Your submission should contain both the report in pdf format, and a zip-file that contains your code and the ten instances that you used to test your algorithms. Make sure to upload the pdf and the zip-file separately!

Assessment

This assignment will be assessed based on your solution approach (0-12 points) and on your report (0-10 points). Your grade is equal min(points/2,10).

In case anything about the assignment is unclear, please use the designated discussion board on Canvas.

References

R. Masson, F. Lehuédé, and O. Péton. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47(3):344–355, 2013.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

M. Schneider, A. Stenger, and D. Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520, 2014.