Assignment 3: Rolling Stock Scheduling and Column Generation

Rowan Hoogervorst – Transport Optimization

October 12, 2022

1 Problem Description

For this assignment, imagine that you are helping Netherlands Railways (NS) with the scheduling of rolling stock. In an earlier step of the planning process, NS has already determined the timetable for a set of Intercity lines that roughly connect the southwest and northeast of the country. Moreover, NS has provided details about the available train unit types, such as the cost of using a train unit of that type and the number of available seats. You can find this information in the CSV files accompanying this assignment (*trains.csv* and *types.csv*). Moreover, a Julia file is provided to you to read this data (*dataReader.jl*).

NS has also given you further information about the way in which they solve rolling stock scheduling problems. Based on that information, you start out with the following constraints/assumptions:

- The length (in carriages) of a composition can be no longer than the maximum length specified for each train in the input file.
- Satisfying the passenger demand is not a hard constraint, but not offering a seat to a passenger in either first or second class leads to a cost (penalty). This cost is 0.5 per passenger that cannot get a seat on a train.
- The number of train units that stay overnight at each station should be equal each night. However, individual trains do not have to stay at the same station every night.
- The goal is to minimize the total cost, consisting of the costs for using train units and the costs for not offering enough seats to passengers.

If something is not specified explicitly above, you can assume that the assumptions as made in the course slides still hold.

Currently, the rolling stock scheduling of NS is based on a multi-commodity flow based model. However, NS is interested in the benefits of a path-based model, as they were told that path-based models often scale better. You will therefore be using the path-based model presented in lecture 7 and adapt it further to the setting of NS. Your final goal is to provide a first prototype of a column generation model for NS.

2 Tasks

The work you will have to do is subdivided into three different steps.

2.1 Defining the Problem and Solution Approach

First, you write down the mathematical problem and the details of the column generation procedure to solve it:

1. Write down the path-based rolling stock scheduling model for the problem, taking into account the above assumptions and constraints.

2. Describe the reduced master problem and pricing problem that you will use in the column generation procedure to solve the model that you have defined under **1.**, i.e., give mathematical (programming) formulations of the problems that you will be solving.

2.2 Implementing the Column Generation Procedure

After defining the problem, you implement the column generation procedure to solve the instance that has been provided to you:

3. Create the time-space graph, through implementing the needed data structure, that you will use in the pricing problem. Assume that all dual costs are zero for now. Check the graph carefully and report basic information on the graph (e.g., number of nodes and edges).

4. Implement the reduced master problem and, if needed, initialize it with a set of artificial columns to ensure feasibility. Describe the obtained model, i.e., give the number of variables and constraints.

5. Execute the first three iterations of the column generation procedure. Give information on the columns found and their reduced cost.

6. Now solve the linear programming relaxation of the path-based model through column generation, i.e.., continue the procedure started under **5.**. For the optimal solution, report the number of used trains of each type and the number of passengers that are not offered a seat. Moreover, describe the set of optimal paths.

7. Use the set of columns you found in the column generation procedure to try and find an integer solution, i.e., solve the original integer path-based model with only those columns that you just found in the column generation procedure. Report on the results found. Does this provide you with the optimal solution, or only with a lower or upper bound?

2.3 Adjusting your Prototype Based on Feedback from NS

After discussing your results with NS, you are told that an important restriction has so far been missing from your model. In particular, NS tells you that the number of trains that can be stored in station Utrecht (Ut) during the night is limited due to long-term construction work. In total, no more than 8 train units can be parked there overnight. To show that your prototype can handle the new constraint, do the following:

8. Formulate the new model, which takes into account the new restriction on parked train units. Describe the changes to your reduced master problem and pricing problem, if any.

9. Adapt your column generation implementation, in the way you described under **8.**, and re-solve the instance. Compare the found solution to the existing one. How expensive is this restriction on the number of parking spots for NS?

10. Perform a sensitivity analysis on the number of available parking spots at station Utrecht. What would the value be for NS of trying to achieve more parking spots during the construction works?

Another concern that NS has is about the quality of the integer solution that you found. In particular, they are worried that only using the columns from the initial column generation procedure might not be sufficient. Do the following to show that your method could be adapted to potentially find better integer solutions:

11. Describe how a Branch-and-Price approach could be used to find the optimal integer solution. Which branching decisions will you make in such an approach. **Note**: you do not have to implement this Branch-and-Price approach.

3 Hints

- The JuMP tutorial on column generation gives further information on which functions can be used in JuMP to implement column generation and how a column generation procedure in Julia might look like.

- For implementing the time-space graph, you might want to represent the graph through a list (vector) of arcs as you did in assignment 2. Each arc describes the source node of the arc, the target node of the arc and the cost of using the arc. You can then use set operations, like in assignment 2, for creating your constraints. Also see this JuMP tutorial for working with set indexing for constraints.

- It could be handy to create a small test dataset to test your methods on and to use for debugging. For example, you could select about 5-10 trains from the instance.
- The paper by Lusby et al. (2017) in the reading material of week 7 might also be of help in working out your column generation procedure.

4 Requirements for Handing In

When handing in your report, make sure to:

- Hand in the report as a PDF file.
- Hand in the code together with the report either as a single Julia file or as a ZIP file containing all your Julia files. Ensure that the code is sufficiently documented.
- Describe clearly what each member in your team did
- Stick to a page limit of 6 pages for the report, excluding the title page and a (potential) table of contents. Adding supplementary results in an appendix section is allowed, but note that it should be possible to grade/read the report without consulting the appendix.