Rail Scheduling Problem

Fedir Kovalov

January 26, 2024

1 The problem

In the project description, I did not address the project goal and the problem description properly, which I am going to address in this section.

1.1 Motivation

Consider the following image :

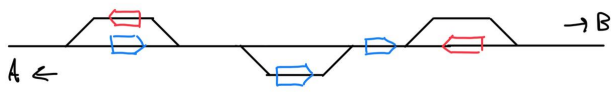

Figure 1: In the image, you can see a piece of track, with two types of trains: blue, which must arrive at the point B(go towards the right side), and red, which must arrive at the point A(go towards the left side).

Now consider the following question: *which trains should yield, and which should have priority?*. Also consider, that the trains might have different speeds, acceleration rates, lengths, and how the schedule needs to change if one of the trains breaks down. Such problem not only has no straight forward optimal solution, but in practice those kinds of problems often need to be solved in real time.

The project aims to generalize and solve those types of problems using a computer model.

1.2 Model

In order to solve the mentioned above problem we need a proper representation of the system, which is going to be represented as follows:

- Rail network is represented as a directed graph of *Tracks*, which are grouped into *Sections*, such that the train is able to move from one track to another if and only if there is a corresponding edge in a graph.

- *Section*, is a piece of the rail network such that if two trains move onto it, they will inevitably collide with each other. Those are not necessarily comprised out of one track, but rather some set of tracks that cross each other at some point. For example, an x-crossing is a single section made of two crossing tracks. This is necessary for routing.
- *Track*, is a vertex in our graph that represents an actual piece of rail.

Typically, railroad operators already have the track divided into sections, as this is the way rail traffic is traditionally managed: Semaphores are placed on the entrances to the section, and one train must wait at the semaphore until the other leaves.

For example, our system from the Figure 1, would look like this:

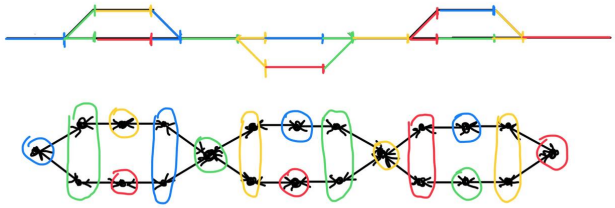

Figure 2: Colored divisions are sections, and the black vertices are tracks.

Trains, and their schedule specifically is represented as a path in this graph. The specifics of the definition of paths are addressed in the Solution section below.

2 Solution

This section describes the results of (still ongoing) research on the problem, and possible practical solution of it (see section "Were is the code?").

2.1 Conflict Resolution Problem(CRP)

We say, that a conflict occurs, when two trains want access to the same section at the same time.(`https://doi.org/10.1016/j.ejor.2006.10.034`) Solving such a conflict boils down to deciding on which one of the trains must wait. This process in an integral part of creating a schedule. (the problem we are solving is commonly known as the *Conflict resolution problem(CRP)*).

2.1.1 Job-Shop scheduling

This method of solving the CRP is sourced from the articles:
`https://doi.org/10.1016/S0377-2217(01)00338-1`

https://doi.org/10.1016/j.ejor.2006.10.034

The idea of this approach is to represent the CRP in the same way as the job-shop problem, where each train is a job, and each section is a machine. The schedule then, is represented as an alternative graph $G = (N, F, A)$, where each node is train passing through some Section. I won't go into malicious details of the alternative graph model here, my sources describe it quite well already.

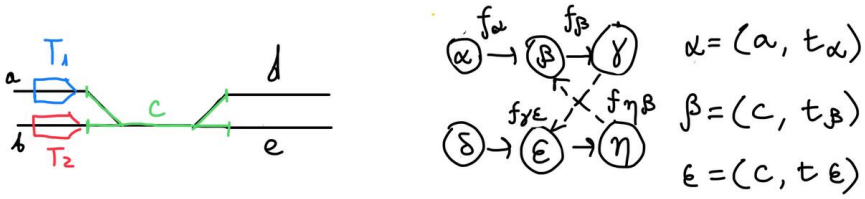

Figure 3: Example of the alternative graph model for the two trains: The top part of the graph represents schedule for the train T_1, and the bottom one is the schedule for the train T_2. Each vertex π contains a section x being occupied, and the time t_π at which the train is entering it. Normal(stable) edges, drawn as a normal line, on the other hand contain *time constraint* f_π such that $t_\theta >= t_\pi + f_\pi$, where θ is the vertex the edge is pointing towards.

What are the alternative edges (drawn as dashed lines) (η, β) (γ, ϵ) ? Those are our conflicts: edge (η, β) has weight $f_{\eta\beta}$, meaning that job(as is the job-shop model) β can only start after $f_{\eta\beta}$ time since η started, or $t_\beta >= t_\eta + f_{\eta\beta}$ (otherwise the train would occupy the position that is previous to η at the same time as the other train occupies position at β, and we can't allow that). Similarly we have an edge (γ, ϵ), which means that $t_\epsilon >= t_\gamma + f_{\gamma\epsilon}$. The CRP then, reduces to identifying such conflicts, and changing operation starting time in such a way that exactly one of the conditions imposed is satisfied.

The original paper (https://doi.org/10.1016/j.ejor.2006.10.034) talks about dummy nodes(start(also known as planning horizon) and the finish node), and cycles in this graph. In the same paper, authors actually built a simple algorithm, that just tries all the possibilities one-by-one. One of the results that they have shown indicated that schedules calculated by the computer had a lot less delay then those generated by more conventional methods of routing trains.

2.1.2 Solutions to the problem

Right now, figuring out the exacts of the solution to the CRP is in the working. There are overwhelming amount of different solutions to this problem with overwhelming amount of different complicated proofs. Some of the solutions considered:

- https://doi.org/10.1016/j.trb.2009.05.004 - tabu search
- https://doi.org/10.1016/j.cor.2022.105859

3 Where is the code?

You may have noticed that there is no code attached to the winter report. This is the result of the fact that this task has proven to be a lot more difficult than anticipated. From the start, I tried to solve this problem in a much more naive way (more detail in the section below). Unfortunately, this has proven to be a waste of time, as many of the problems of my model have arose and I decided to scrap it(I talk about why below).

3.1 My original model and solution

The model I have originally created is a lot more simple than the one described above: it represents train routes as sequences of vertices in the network graph. The i-th vertex in a path represents position of the respective train after i amount of time since the planning horizon. The algorithm that I have tried using in this model was also quite simple: it looked for the train that would spend the least amount of time on the contested section, and made every other train wait it, by extending time spent on the previous vertex.

3.2 Why was I wrong?

Let's first address the algorithm: the inefficiency of this approach was a direct result of the `https://doi.org/10.1016/j.ejor.2006.10.034` paper. The researchers compared their method to the one similar to mine, and found out theirs to be much more efficient(relative to the produced delay). In fact, this method is one of those traditionally used for train scheduling.

As for the model, there are many problems, but the largest one probably is that my model does not allow for any additional information to be added for the scheduler(like time constraints for arriving to the station, transfers, etc). The Job-shop model on the other hand allows to do this with ease, and generally looks like a superior model to mine.

4 Conclusion

The problem of train scheduling turned out to be a lot more complicated than I anticipated. After trying a more naive solution, and eventually proving my own solution wrong, I am now looking for a better solution in the literature on the topic.