The Train Length problem[1]

Martin Gantenbein[2]

May 7, 2001

[1]Work partially supported by the Swiss Federal Office for Education and Science under the Human Potential Programme of the European Union under contract no. HPRN-CT-1999-00104 (AMORE)
[2]Department for Computer Science, ETH Zürich

Contents

1 Introduction **3**

2 The Train Length problem **4**
- 2.1 Notation and Assumptions 4
  - 2.1.1 Train Cars 4
  - 2.1.2 Stations 4
  - 2.1.3 Events 5
  - 2.1.4 Scheduled Rides 5
  - 2.1.5 Empty Rides and Waiting Trains 6
  - 2.1.6 Train Length 7
  - 2.1.7 Cost Functions 8
  - 2.1.8 Costs and Constraints 8
- 2.2 Problem Definition 9
- 2.3 A graphical Representation 10
- 2.4 Applications 10
  - 2.4.1 Local Cost Functions 11
  - 2.4.2 Global Cost Functions 11

3 Theoretical Results **16**
- 3.1 Periodicity 16
  - 3.1.1 The One-Period Model 22
- 3.2 Fixed Charges and Complexity 23

4 Network Flow Models **28**
- 4.1 The Model 29
  - 4.1.1 Scheduled Rides 29
  - 4.1.2 Waiting Arcs 29
  - 4.1.3 Empty Ride Arcs 30
  - 4.1.4 Time complexity 32
- 4.2 Applications 33
- 4.3 Fixed Costs 33
  - 4.3.1 Empty Rides 34
  - 4.3.2 Fixed Costs on Waiting Arcs 35
- 4.4 Network Size 36
- 4.5 Conclusion 38

5 Results **39**
- 5.1 Data Preparation 39
- 5.2 Test Runs 40

6 Future Work **42**
6.1 The Train Car Routing Problem 42
6.2 Improvement Algorithms 42
6.3 Train Types 44

7 Conclusion **45**
7.1 Credits 45

1 Introduction

To implement a timetable, railway companies have to define the number of train cars on each train in order to satisfy passenger demand and also to have the required cars available when a train departs.

An informal definition of the problem is given as follows. A set of train rides (the timetable) is given. For each of these rides, the following data are given:

- departure and destination station,
- departure and arrival time, and
- the number of required train cars, according to the expected number of passengers.

In addition to this, we expect a cost function which allows to compute the cost for each trip given the number of train cars.

In a legal realization of the schedule, the cars for each trip must be available at the departure station in time. Because the timetable does not guarantee this feature, we may need to route more train cars than required from one station to another. There are two ways to do this:

- we can connect a certain number of additional cars to scheduled train rides, or
- we can add additional, non-scheduled trains.

We are interested in the case where trains are built of identical train units that are able to ride on their own, without a locomotive. Examples for trains built of identical units are tramways, the French TGV, the Italo-Swiss CISALPINO or the S-BAHN ZÜRICH trains. The goal of the Train Length problem is to realize the given schedule with minimal total cost. With "realizing the given schedule" we refer to the process of defining the length of each scheduled train and the lengths of additional, non-scheduled trains. A solution to the Train Length problem does not define the exact routes of each train car, it only defines the number of cars on each ride.

2 The Train Length problem

We begin with a definition of our notation, give a mathematical formulation of the Train Length problem and define a graphical representation for the problem.

2.1 Notation and Assumptions

The expressions we will use below will be defined here and we will also make assumptions to simplify the problem.

2.1.1 Train Cars

As mentioned above, we are interested in trains built of train units that can ride on their own. We will refer to such train units as *train cars* or simply *cars* below.

2.1.2 Stations

Each ride begins in the *departure station*, passes any number of *intermediate stations* and finally arrives at the *destination station* or *arrival station.*

Assumption 1 *The number of train cars on a train is constant during the trip from departure station to destination station.*

We assume that train cars are neither attached nor detached at intermediate stations, although such operations occur in practice. By splitting a ride at intermediate stations where adding or removing cars is possible, we can enable such behavior. If for example a ride has departure station A and destination station B, and the ride is passing through C, where cars could be coupled to or decoupled from the train, we can split the ride into two rides, one from A to C and a second one from C to B.

With assumption 1, only departure and destination stations have to be considered for the Train Length problem. To simplify the following we will just refer to *stations* when we mean departure and destination stations. We will talk about intermediate stations very rarely, and if we do so, we will explicitly call them intermediate stations. If required, we will also use the expression *terminal station* for a departure or destination station.

We will refer to $s_d(r)$ as the departure station and to $s_a(r)$ as the destination (or arrival) station of ride r. Round-trips, where $s_d(r)$ is equal to $s_a(r)$ are allowed. We will refer to m as the total number of terminal stations.

Assumption 2 *The track network allows train rides between any pair of stations (the network graph of the tracks is connected).*

In case of a partitioned network we can also partition the timetable into timetables for each connected region and solve the Train Length problem for each part separately.

2.1.3 Events

An event is defined by a station and a time. We refer to $\langle s, t\rangle$ as the event in station s at time t. Note that we will refer to s_a as the station of an event a and to t_a as the time of a. E will be referred to as the set of events. We will refer to the set of rides that arrive at an event a as $\mathrm{arr}(a)$ and to the set of rides that depart from a as $\mathrm{dep}(a)$. For each train scheduling problem we will define E according to the schedule. Arrivals and departures of scheduled trains are always contained in E, but there may be additional events. We assume that $|E|$ is finite.

2.1.4 Scheduled Rides

We consider a set S of n train rides r_{ab}, r_{ab} is the train ride that departs at event a and arrives at event b. S is referred to as the set of *scheduled rides*.

Assumption 3 *S defines a periodical timetable.*

A periodical timetable is a timetable that will be repeated over and over again. We will refer to ρ as the duration of one period. We are used to think about one-day periods ($\rho = 24$ hours), although railway companies have special timetables for Saturdays and Sundays and it would be more reasonable to talk about one-week periods ($\rho = 7$ days). Still, we will consider daily schedules (without loss of generality) to fit our intuition of railway schedules.

Each scheduled train ride is defined by its departure time t_a, its departure station s_a, its arrival time t_b and its arrival station s_b. For each scheduled train ride $r_{ab} \in S$, the minimum number l_{ab} of train cars (according to passenger demand) and the maximum number u_{ab} of cars (according to maximum train length on the respective route) are given.

Assumption 4 *For any scheduled train ride r_{ab}, l_{ab} and u_{ab} are positive integers and $l_{ab} \leqslant u_{ab}$.*

The assumption $l_{ab} \leqslant u_{ab}$ must be checked when l_{ab} is evaluated according to the expected number of passengers. At peak times, passenger demand could require longer trains than u_{ab} allows; in such cases we have to set $l_{ab} = u_{ab}$, a pretty unpopular but inevitable decision....

2.1.5 Empty Rides and Waiting Trains

With *empty rides* we refer to unscheduled train rides that are used to transfer empty train cars from one station to another, with *waiting trains* we refer to trains that are waiting in a station. A waiting train is conceptually the same as an empty ride, the only difference is that the departure and destination stations are equal for waiting trains. We will therefore also use the expression "waiting ride", despite the contradiction in terms. Empty rides are possible between any pair of events a and b where $s_a \neq s_b$, waiting rides are possible between any pair of events a and b where $s_a = s_b$. In our models we will only allow certain empty rides, for example an empty ride r_{ab} will only be allowed if b is the first event in s_b that can be reached by a train departing in a. We assume that an empty ride r_{ab} always chooses the shortest route from s_a to s_b.

We have now defined scheduled, empty and waiting rides. We have already seen that empty and waiting rides are basically the same. But also scheduled rides are very similar: the only difference is that scheduled rides have a minimum length greater than 0, while the minimum length of empty and waiting rides is 0. We will use the common notation r_{ab} for all kinds of rides, and we will refer to them as *rides* or *train rides*. The set of all rides will be referred to as R.

Timing A ride r_{ab} is defined to depart at event a and to arrive at b. Because we have a periodic schedule, r_{ab} is performed in every period. For most rides r_{ab}, the riding time is $t_b - t_a$. But consider an overnight ride or a train that is waiting during the night. For such rides a and b do not occur within the same period and the riding time is $t_b + \rho - t_a$.

We define δ_{ab} to be the number of periods between a and b for ride r_{ab}. The riding time of r_{ab} is

$$t_{ab} = t_b + \delta_{ab} \cdot rho - t_a.$$

For most scheduled trains δ_{ab} will be 0, for overnight rides it is 1. For very long-distance trains, δ_{ab} could even be greater than 1. δ_{ab} is defined by the schedule for empty rides.

For an empty ride r_{ab}, δ_{ab} is not defined by the schedule. We assume that r_{ab} always chooses the shortest route from s_a to s_b, so we can find out how long the ride takes. This lets us find the first period in which b can be reached from a. We define δ_{ab} for an empty ride r_{ab} to be the number of periods that is at least required between a and b.

Table 1 lists some examples for $\rho = 24$ hours, with periods starting at midnight. The table shows 5 examples for a ride r_{ab}. In the first two columns

	t_a	t_b	s_a	s_b	δ_{ab}
1	08:00	08:30	A	A	0
2	08:00	09:00	A	B	0
3	13:00	08:00	A	A	1
4	08:00	08:28	A	B	1
5	23:55	00:15	A	B	2

Table 1: Examples for δ_{ab}

t_a and t_b show the departure and arrival times of r_{ab}. Columns 3 and 4 (s_a and s_b) contain the departure and arrival stations. Examples 1 and 3 are waiting rides, they start in A and end in A, examples 2, 5 and 6 are empty rides from A to B. We assume that the transfer time needed for a ride from A to B is at least 30 minutes. With this data it is possible to evaluate the last column: the minimum number of periods that have to lie between a and b.

Examples 1 and 2 show the case where both events can be in the same period (therefore $\delta_{ab} = 0$). In case 3 it is clear that b must be the event b in the period after a, because b happens before a. Therefore, δ_{ab} must be 1 in case 3. In example 4 , δ_{ab} is 1 because the time between a and b is not sufficient (30 minutes of transfer time would be required). Example 5 demonstrates the case where a is very late and b very early, such that δ_{ab} even has to be 2.

The maximum δ_{ab} depends on the properties of the schedule. If the gap between the last arrival and the first departure in the next period is large enough, the case $\delta_{ab} = 2$ does not occur, but in other schedules even $\delta_{ab} > 2$ is possible — for example, if the distance between s_a and s_b is very large. In our example 5, δ_{ab} would be 3 if the transfer time from A to B were 25 hours.

The maximum δ_{ab} also depends on the time that is chosen for the beginning of periods. A good choice would be a very inactive time, for example 4 a.m.

2.1.6 Train Length

For scheduled and empty rides, we have defined the maximum number of train cars as "the maximum length of a train on the respective route". For scheduled rides the maximum length of a train depends mainly on the platform lengths on the respective route. The shortest platform on the route of a scheduled ride determines the maximum length of that train. For empty trains, platform lengths are not important because there are no passengers that have to get on or off the train. Other factors, like shunting constraints, that may influence the maximum train length are common for scheduled and empty rides.

SBB even claims that the lengths of empty rides are not bounded at all. We will not go that far, but we will keep in mind that empty trains may be longer than scheduled trains.

In a solution of the Train Length problem, the number of train cars on each ride is defined. We refer to the number of train cars coupled to r_{ab} as x_{ab}. The solution of the Train Length problem, the vector of all x_{ab}, will be referred to as x.

2.1.7 Cost Functions

For every ride r_{ab} a cost function $f_{ab}(x_{ab}) : \mathbb{N} \to \mathbb{N}$ is defined.

Assumption 5 *The cost functions $f_{ab}(x_{ab})$ are functions of the form*

$$f_{ab}(x_{ab}) = \begin{cases} 0, & \text{if } x_{ab} = 0 \\ c_{ab} \cdot x_{ab} + d_{ab}, & \text{if } x_{ab} > 0 \end{cases}$$

where c_{ab}, and d_{ab} are nonnegative integers.

We will refer to the c_{ab} as the *linear cost* and to the d_{ab} as the *fixed cost* or *fixed charge.*

2.1.8 Costs and Constraints

To be able to argument about costs an constraints, we have to classify different kinds of costs and constraints. We will separate costs and constraints into *local* and *global* costs and constraints.

Local Costs and Constraints Costs and constraints that are only depending on one train ride are called local costs and constraints. Examples for local costs are

- waiting cost in a station,
- car and track usage costs, and
- personnel cost.

Examples for local constraints are

- the minimum and maximum number of cars on a train (l_{ab} and u_{ab}),
- the constraint that train cars can neither be destroyed nor created in stations, and
- time constraints (defining δ_{ab}).

Global Costs and Constraints In contrast to local costs and constraints, the class of global costs and constraints depends on properties of several train rides. Again, some examples for global costs:

- cost for coupling and decoupling,
- costs depending on the maximum number of trains (or train cars) riding at the same time, and
- route costs for r_{ab}: another empty ride may use a track of the shortest route of r_{ab} at the same time, in that case, r_{ab} might have to take a different route (with a different cost).

Two examples for global constraints:

- An upper bound for the total number of railway cars,
- track constraints: an empty ride might even be impossible, depending on other empty rides, because two empty rides might have to use the same track at the same time.

Traceable-car Costs and Constraints Costs and constraints that require tracking cars on their rides are more complex than global costs and constraints, because global costs and constraints only depend on the *number* of cars on each train but not on the *identity* of a car. Examples for traceable-car constraints are maintenance constraints like "No car can drive more than 10000 km without being maintained". Such costs and constraints cannot be part of the Train Length problem, because the Train Length problem is aimed to optimize the number of cars on each trip, exact routes for cars are not found by solutions to this problem. We will therefore not consider traceable-car costs and constraints here.

2.2 Problem Definition

To define the Train Length problem we have to be slightly more general than above. We have to consider that the train length assignment x may vary from period to period. We will refer to x^α as the train length assignment in period α. x^α consists of entries x^α_{ab}, x^α_{ab} is the number of train cars on ride r^α_{ab}, the ride from event $a^{\alpha-\delta_{ab}}$ to b^α. $a^{\alpha-\delta_{ab}}$ is the event a in period $\alpha - \delta_{ab}$, b^α is the event b in period α. Note that we define the rides *arriving* in period α to belong to period α.

There is one special case to be considered: for some rides, $\alpha - \delta_{ab}$ may be negative. The rides r^α_{ab} with $\alpha - \delta_{ab} < 0$ are used to "feed" the system with train cars, we could think of them connecting the train car factory to the railway system.

We will refer to $f(x^\alpha)$ as the total cost for period α.

Assumption 6

$$f(x^\alpha) = \sum_{r_{ab} \in R} f(x^\alpha_{ab})$$

We assume that the total cost in period α is the sum of the costs for all the rides in period α.

The problem definition is given below:

$$\min \lim_{p \to \infty} \frac{1}{p} \sum_{\alpha=0}^{p-1} f(x^\alpha) \tag{1}$$

s.t.

$$\sum_{a: r_{ab} \in R} x^\alpha_{ab} = \sum_{c: r_{bc} \in R} x^{\alpha+\delta_{bc}}_{bc} \quad \forall\, b \in E,\ \alpha = 0 \dots p-1 \tag{2}$$

$$l_{ab} \leqslant x^\alpha_{ab} \leqslant u_{ab} \qquad \forall\, r_{ab} \in R,\ \alpha = 0 \dots p-1 \tag{3}$$

$$x^\alpha_{ab} \text{ integer} \qquad \forall\, r_{ab} \in R,\ \alpha = 0 \dots p-1. \tag{4}$$

In this formulation, we minimize the average cost per period assuming the schedule repeats infinitely (1). Equation 2 makes sure that the same number of cars that arrives in an event b also departs from b. We will refer to this equation as the *car-conservation equation*, because it ensures that cars are neither destroyed nor constructed in stations. Inequations 3 define the minimum and maximum number of train cars per ride, we will refer to those inequations as the *bound inequations*. Equation 4's task is to take care that train cars are not cut into pieces.

2.3 A graphical Representation

We represent events by nodes and rides by arcs connecting the respective nodes. For better understanding, we will draw the nodes ordered by time from left to right (early rides left, late rides right). Figure 1 shows an example with $n = 2$, solid arcs represent scheduled rides, dashed arcs represent potential empty rides. In this example we have decided that the set of events, E, is the set of scheduled arrivals and departures and we decided that empty rides are allowed from arrival nodes to departure nodes.

2.4 Applications

The Train Length problem does not define the cost function to be minimized. The goal is to find an optimal train length assignment according to a given cost function. We will give examples for cost functions that can be modelled according to our assumptions.

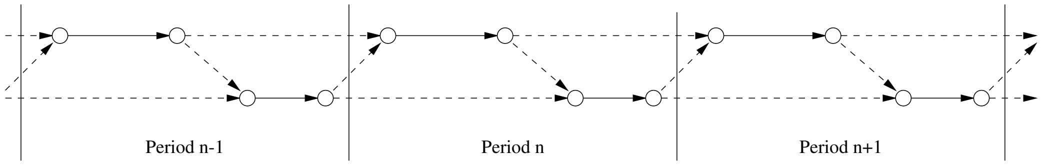


Figure 1: Schedule with $n = 2$, showing 3 periods

2.4.1 Local Cost Functions

Assumption 6 restricts the cost functions to local costs, because the total cost is defined to be the sum of the costs for all the rides. Costs that are depending on ride r_{ab} are contained in the local cost function $f(x_{ab})$. Assumption 5 restricts the local cost functions to functions with a linear and a fixed charge term. For many of the costs occurring in reality, such cost functions are sufficient. If we choose time or distance as cost, not even the fixed charge term is required.

Empty and Waiting Time To minimize the time spent on empty or waiting rides, we would use the following cost functions:

$$f(x_{ab}) = \begin{cases} 0, & \text{if } l_{ab} > 0 \\ (t_b + \delta_{ab} - t_a) \cdot x_{ab}, & \text{if } l_{ab} = 0. \end{cases}$$

Rides with $l_{ab} > 0$ are scheduled rides, therefore we don't have to consider them. For the other rides, we set the linear cost to the time duration of the ride.

Empty Rides To minimize the number of empty rides, the following cost function would apply:

$$f(x_{ab}) = \begin{cases} 1, & \text{if } l_{ab} = 0 \wedge s_a \neq s_b \wedge x_{ab} > 0 \\ 0, & \text{otherwise.} \end{cases}$$

We only have to count empty rides r_{ab} with $x_{ab} > 0$, the other potential empty rides are not used. Empty rides are rides with $l_{ab} = 0$ and $s_a \neq s_b$. This is an example for a cost function with fixed charges.

2.4.2 Global Cost Functions

General global costs cannot be modelled with a cost function as assumed in assumption 6. In the following examples we will show global costs that can be mapped to local cost functions. We assume that global cost functions are of the form

$$f(y) = \begin{cases} c \cdot y + d, & \text{if } y > 0 \\ 0, & \text{if } y = 0. \end{cases} \tag{5}$$

y depends on several rides, otherwise $f(y)$ would not be a global cost function.

Total Riding Distance A cost function depending on the total riding distance is a global cost function, because it clearly depends on more than one ride. But it is obvious that all cost functions depending on the sum of local functions like distance, time, ... can easily be mapped to local functions. The global cost function $f(y)$, where y is the total distance, can be expressed as

$$f(y) = d + c \cdot \sum_{r_{ab} \in R} y_{ab} \cdot x_{ab},$$

where y_{ab} refers to the distance of r_{ab}. So we can define local cost functions as follows:

$$f(x_{ab}) = c \cdot y_{ab} x_{ab} \tag{6}$$

Now the global cost function evaluates to

$$f(y) = d + \sum_{r_{ab} \in R} f(x_{ab}).$$

Except the d term, this has the form of the objective function (1) of the Train Length problem. If we solve the Train Length problem with cost functions (6), we get an optimal solution that does not consider d. But obviously $f(y)$ is minimized if $f(y) - d$ is minimized, and we can find the value of $f(y)$ by adding d to the cost of the solution, if this cost is greater than 0.

Cost functions depending on the total riding distance are an example for global cost functions that can be expressed as a *linear combination* of local cost functions. Obviously, all global cost functions that can be expressed as a linear combination of local cost functions can be modelled with our problem definition.

Number of Train Cars If we have a global cost function that depends on the total number of train cars (y is the total number of cars), y can be expressed as a linear combination of local cost functions. Because this is not obvious, we will show how this works.

Because of the car-conservation equation (2) and because the length of a train is not modified on a trip (assumption 1), the number of train cars is constant and we can just count train cars at any point in time, for example at the beginning of a period. We assume that there are no events at the beginning of the period and that all the cars are on some ride at this time (probably most of them are on a waiting train). To count the cars at the beginning of a period, we can just count the cars that are riding when periods change (at midnight, for example). So cars on rides r_{ab} with $\delta_{ab} = 0$ are not

Figure 2: Decoupling and coupling

counted, they begin and end within the same period. But rides r_{ab} with $\delta_{ab} = 1$ are riding when the periods change, so we have to count the cars on r_{ab}. Rides r_{ab} with $\delta_{ab} = 2$ change periods twice, so the cars on such rides have to be counted twice. We can assign local cost functions to rides as follows:

$$f(x_{ab}) = c \cdot \delta_{ab} x_{ab}.$$

We can again model the Train Length problem with those cost functions and finally add d (in case the cost of the solution is greater than 0) to get the value of $f(y)$.

This example shows how we can model global functions that can be expressed as linear combinations of ride costs at certain time intervals.

Coupling and Decoupling Costs for coupling and decoupling occur at events and they depend on the trains that arrive at and depart from those events and on how the cars of those trains are routed. Because the Train Length problem does not define exact routes for the railway cars, it is not possible to consider coupling and decoupling in general. But there is a special case, where this is possible. Figure 2 shows two examples for events: in the example on the left it is possible to count the number of decoupling operations, on the right we can count couplings. The routing is clear in those cases. On the left we have one train that arrives at the event a and several rides departing. If there is only one departing ride r_{ab} with $x_{ab} > 0$, no decoupling is required. If there are two departing trains, one decoupling is required, ... Except for the first one, each ride with $x_{ab} > 0$ adds a decoupling. Therefore we assign the coupling cost as a fixed cost to each ride departing from a. We can assign the following cost functions to all r_{ab} departing from events a where only one ride arrives:

$$f(x_{ab}) = g_a, \text{ if } x_{ab} > 0 \wedge |\operatorname{arr}(a)| = 1, \; r_{ab} \in R.$$

g_a is the cost for a decoupling at event a. With those cost functions we always count one more decoupling than there is in reality. We can correct this by assigning the negative cost for one decoupling to the ride arriving in a: $f(x_{ca}) = -g_a$.

On the right side of figure 2 we have the inverse situation: there are several arriving rides and only one departing. So we can assign the following cost functions to all r_{ab} arriving at events b where only one ride departs:

$$f(x_{ab}) = h_b, \text{ if } x_{ab} > 0 \wedge |\operatorname{dep}(b)| = 1, \ r_{ab} \in R.$$

h_b is the cost for a coupling at event b. Again, we have counted one more coupling than there is. We correct this by assigning $-h_b$ to the ride departing from b.

If we choose E and R such that for each event a either $\operatorname{arr}(a) = 1$ or $\operatorname{dep}(a) = 1$, it is possible to count couplings and decoupling operations. In the example of figure 1 this is the case: scheduled rides r_{ab} are the only rides departing from a and they are the only rides arriving in b. Because there are only scheduled events, all the events have either only one arriving or only one departing train. In this case we can use the following cost functions:

$$f(x_{ab}) = \begin{cases} -h_a - g_b, & \text{if } l_{ab} > 0 \\ g_a + h_b, & \text{if } x_{ab} > 0 \wedge l_{ab} = 0 \\ 0, & \text{if } x_{ab} = 0. \end{cases}$$

In the first case ($l_{ab} > 0$) we assign the negative cost for a coupling in a and a decoupling in b to scheduled rides r_{ab}, to correct for the coupling and decoupling that is counted in addition to those that have to be counted. In the second case we count the decoupling in a and the coupling in b for empty and waiting rides r_{ab} with $x_{ab} > 0$. The third case is clear: unused empty and waiting trains r_{ab} do not require any coupling and decoupling operations.

Linear Combinations We have seen that cost functions that can be expressed as linear combinations of local costs are fine with our problem definition. Therefore also linear combinations of such cost functions can be modelled. It is clear that the final goal is to minimize monetary cost.

Monetary cost is minimized if the total distance is not too large, if driver costs are low, if there are only few cars, ... With the cost functions described above we can find the minimum distance, the minimum driver cost and the minimum number of cars that's required and we can also find minima for other criteria. But when considering monetary cost, we don't want to minimize a single cost factor but a combination of all of them. To model monetary cost we have to split costs into cost factors that can be modelled as local cost and build the sum of those factors:

$$f(x_{ab}) = (c_1 + c_2 + c_3 + \ldots)x_{ab} + d_1 + d_2 + d_3 + \ldots.$$

Mapping impossible Many global costs can be mapped to local cost functions. But there are still global cost functions we cannot model. For general sets of events and rides, coupling and decoupling costs cannot be modelled because those costs depend on the exact routing of the cars. Also global cost functions of the form (5) cannot be modelled if y is not a linear combination of local costs. But there are few realistic costs of that kind. An example would be an empty ride r_{ab} that can take two different routes with different costs, but a track of the cheaper route could be used by another empty ride, so that if the second ride is used, r_{ab} would have to take the more expensive route:

$$f(x_{ab}) = \begin{cases} c_1 \cdot x_{ab} + d_1 & \text{if } x_{cd} = 0 \\ c_2 \cdot x_{ab} + d_2 & \text{if } x_{cd} > 0. \end{cases}$$

Global costs where y is not a linear combination of local costs have to be ignored in our model.

3 Theoretical Results

As mentioned before, we consider periodical schedules. Therefore our problem definition (equations 1 – 4) built a sum over infinitely many periods. We will show that it is sufficient to model a finite number of periods to get optimal solutions. We will even assume that one period is sufficient. Further we will give an $\mathcal{NP}$-hardness proof for the general problem as defined above.

3.1 Periodicity

Our problem definition (equations 1 – 4) makes it difficult to model the Train Length problem, as it refers to infinitely many periods. We will assume that it is sufficient to model one period to get the optimal solution. As a first step, we show that there must exist an optimal solution which can be repeated after a finite number of periods. Remember that x^α denotes the train length assignment in period α.

Proposition 1 *There exists an optimal solution to the Train Length problem where $x^{\alpha+\varphi} = x^\alpha$ for some nonnegative integer φ and for any α, $0 \leqslant \alpha \leqslant \varphi - 1$.*

The proposition claims that for each instance of the Train Length problem there exists a φ, such that the train length assignment is periodic with a frequency of φ periods.

Proof: The optimal solution x to the infinite problem defines an infinite sequence of train length assignments $[x^0, x^1, x^2, \ldots]$. Because each entry in x^α is bounded above and below by the bound inequations (3) and because all the values are integer (equation 4), there exists only a finite number of possibilities to set each entry in x^α. Because x^α contains only a finite number of entries (one for each ride), there exists only a finite number of possible settings for x^α. It follows that there must be pairs x^α, x^β in x where $x^\alpha = x^\beta$ and $\alpha < \beta$. Consider such a pair x^α, x^β and the subsequence $s = [x^\alpha, x^{\alpha+1}, \ldots, x^{\beta-2}, x^{\beta-1}]$.

Because $x^{\beta-1}$ can legally be followed by x^β it can also be followed by x^α, therefore it is a legal realization of the schedule to repeat s infinitely. It is also a legal realization of the schedule to remove s from x (because $x^{\alpha-1}$ can legally be followed by x^β). The average cost

$$a_s = \frac{1}{\beta - \alpha} \sum_{i=\alpha}^{\beta-1} f(x^i)$$

of s must be equal to the average cost a_x of x, otherwise x would not be optimal:

- If $a_s < a_x$, the infinite repetition of s would lead to a better solution than x, and
- if $a_s > a_x$, removing s from x would lead to a better solution than x.

This proves the proposition and we can define a periodic optimal solution $[x^\alpha, x^{\alpha+1}, x^{\alpha+2}, \ldots, x^{\beta-2}, x^{\beta-1}]$ which repeats after $\varphi = \beta - \alpha$ periods. $\square$

This leads to a simplification of our problem definition:

$$\min \frac{1}{\varphi} \sum_{\alpha=0}^{\varphi-1} f(x^\alpha)$$

$$\text{s.t. } (2), (3), (4), \text{ and}$$

$$x^{\alpha+\varphi} \;=\; x^\alpha,\ \alpha = 0 \ldots \varphi - 1. \tag{7}$$

We will refer to solutions of this problem as $\bar{x}$. Equation 7 ensures that $\bar{x}$ can be repeated after φ periods. Inserting (7) in (2) leads to the following reformulation of the car-conservation equation (2):

$$\sum_{a:r_{ab}\in R} x_{ab}^\alpha \;=\; \sum_{c:r_{bc}\in R} x_{bc}^{\alpha+\delta_{bc} \mod \varphi} \quad \forall b \in E, \quad \alpha = 0 \ldots \varphi - 1 \tag{8}$$

So we can reformulate our problem definition again: we replace (2) by (8) and we remove (7).

After all those reformulations let us consider what the new problem definition means for our rides. In subsection 2.2 we have described the meaning of x_{ab}^α as the number of train cars on the ride r_{ab}^α, the ride from event a in period $\alpha - \delta_{ab}$ to event b in period α. Things are not as simple in the new formulation. Now x_{ab}^α represents the number of train cars on the rides

$$r_{ab}^{\alpha+p\cdot\varphi}, \quad p = 0 \ldots \infty, \quad 0 \leqslant \alpha \leqslant \varphi - 1$$

riding from a in periods $\alpha - \delta_{ab} + p \cdot \varphi$ to b in periods $\alpha + p \cdot \varphi$.

Figure 3 shows the same timetable as figure 1. Due to the new version of the car-conservation equation (8), the new model is now circular.

We have shown that we can find an optimal solution by considering a finite number of periods φ, but φ is not known and we could expect that φ even depends on the problem instance. But we propose $\varphi = 1$ for each instance:

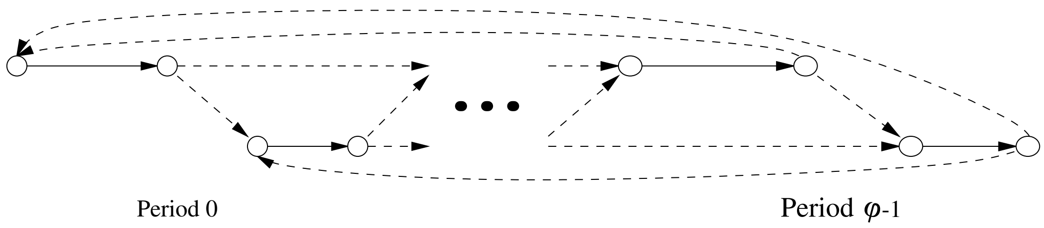


Figure 3: Circular model, repeating all φ periods

Proposition 2 *There exists an optimal solution to the Train Length problem where $x^{\alpha+1} = x^{\alpha}$ if the cost functions satisfy assumptions 5 and 6.*

Let us reformulate our problem definition again, assuming the proposition is true. We only have to model one period now, so we can replace the x_{ab}^{α} by x_{ab} as there will only be one instance of each ride r_{ab}. This simplifies our problem definition to:

$$\min f(x) \tag{9}$$

s.t.

$$\sum_{a:r_{ab}\in R} x_{ab} = \sum_{c:r_{bc}\in R} x_{bc} \quad \forall\ b \in E \tag{10}$$

$$l_{ab} \leqslant x_{ab} \leqslant u_{ab} \qquad \forall\ r_{ab} \in R \tag{11}$$

$$x_{ab} \text{ integer} \qquad \forall\ r_{ab} \in R \tag{12}$$

Let us refer to the optimal solution of this problem as $\hat{x}$. If we want to prove proposition 2, we have to compare $\bar{x}$ (the optimal solution to the φ-period case) to $\hat{x}$. To be able to do this, $\hat{x}$ has to be repeated φ times to yield a legal realization for φ periods. We will refer to this repetition as $\varphi\hat{x}$. Before we can compare $\bar{x}$ to $\varphi\hat{x}$, we need some notation: A *directed path* is a sequence of rides that can be performed by a train car in the given order. A directed path is a sequence of distinct rides

$$\left[r_{ab}^{\alpha}, r_{bc}^{\alpha+\delta_{bc}}, r_{cd}^{\alpha+\delta_{bc}+\delta_{cd}}, \ldots\right].$$

A *directed cycle* is a directed path starting and ending at the same event. An *undirected path* is a sequence of distinct rides that could be performed by a train car in the given order if time were allowed to flip between running forwards and backwards, but not to jump. We refer to rides r_{ab} where time is running backwards as rides $-r_{ab}$. We will also refer to "normal" rides as *forward rides* and to rides where time is running backwards as *backward rides*.

undirected path rules

ride	next		
r^{α}_{ab}	$r^{\alpha+\delta_{bc}}_{bc}$	or	$-r^{\alpha}_{cb}$
$-r^{\alpha}_{ab}$	$-r^{\alpha-\delta_{ab}}_{ca}$	or	$r^{\alpha-\delta_{ab}+\delta_{ac}}_{ac}$

(a)

undirected cycle rules

first	last		
r^{α}_{ab}	$r^{\alpha-\delta_{ab}}_{ca}$	or	$-r^{\alpha-\delta_{ab}+\delta_{ac}}_{ac}$
$-r^{\alpha}_{ab}$	$-r^{\alpha+\delta_{bc}}_{bc}$	or	r^{α}_{cb}

(b)

Table 2: Undirected path and cycle rules

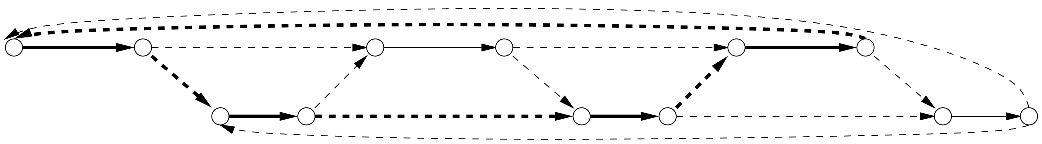


Figure 4: Directed cycle (bold arcs)

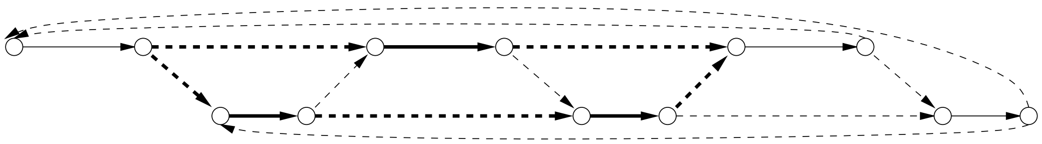


Figure 5: Undirected cycle (bold arcs)

In an undirected path, ride sequences as shown in table 2(a) are possible. An *undirected cycle* is an undirected component path that starts and ends as shown in table 2(b). Figure 4 shows an example of a directed cycle, figure 5 shows an example of an undirected cycle.

Lemma 1 *A solution x to the Train Length problem can be expressed as a sum of directed cycles.*

The lemma follows from the car-conservation equation (8) and the directed version of Euler's theorem (see [1]). Each of the cycles could be used as a route for a train car. Note that each ride r_{ab} is contained in exactly x_{ab} directed cycles because each cycle defines the route of exactly one train car.

Lemma 2 *The difference $d = x - y$ between two legal realizations of a schedule can be expressed as a sum of undirected cycles.*

In x and y, the car-conservation equation (8) is satisfied, therefore it is also satisfied in the difference of x and y. So d must consist of cycles (by Euler's theorem), but there may be $d_{ab} < 0$; the $d_{ab} < 0$ correspond to backward rides $-r_{ab}$ in the cycles.

Let us now compare the legal realizations $\varphi\hat{x}$ and $\bar{x}$. We are interested in the cost $f(\varphi\hat{x})$ and $f(\bar{x})$.

Proposition 3 *$f(\bar{x}) = f(\varphi\hat{x})$ for linear cost functions.*

The proof of this proposition will prove proposition 2 for the simplified cost functions

$$f(x_{ab}) = c_{ab} \cdot x_{ab}$$

We will *modify a train length assignment x along an undirected cycle C*, with this statement we refer to the following modification of x:

$$x'_{ab} = \begin{cases} x_{ab} + 1, & \text{if } r_{ab} \in C \\ x_{ab} - 1, & \text{if } -r_{ab} \in C \\ x_{ab}, & \text{otherwise.} \end{cases}$$

Changing a train length assignment along a cycle is done by increasing the number of cars on forward rides and decreasing it on backward rides. This results in a train length assignment that does not violate the car-conservation equation (8) but that may violate the bound inequations (3). Because the rides have only linear costs, it is easy to find out the cost of a solution $x' = x + C$, the train length assignment x that was modified along a cycle C:

$$f(x') = f(x + C) = f(x) + f(C) = f(x) + \sum_{r_{ab} \in C} c_{ab} - \sum_{-r_{cd} \in C} c_{cd}.$$

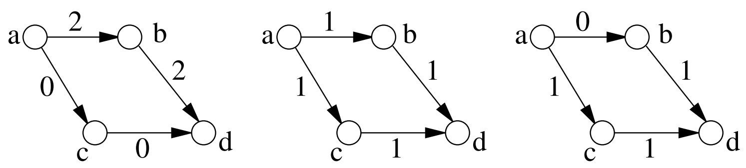
It is therefore possible to evaluate the *cost of a cycle C*, which corresponds to the difference between $f(x)$ and $f(x + C)$. The cost of a cycle C is

$$f(C) = \sum_{r_{ab} \in C} c_{ab} - \sum_{-r_{cd} \in C} c_{cd}.$$

After this definition of $f(C)$ we can prove the proposition.

Proof: We want to prove $f(\bar{x}) = f(\varphi\hat{x})$, where $\bar{x}$ is the optimal solution to the φ-period case and $\hat{x}$ is the optimal solution to the one-period case. $f(\bar{x}) \leqslant f(\varphi\hat{x})$ is clear, because $\bar{x}$ is the optimal solution to the φ-period case. We will show that $\hat{x}$ could not be the optimal solution to the one-period case if $f(\bar{x}) < f(\varphi\hat{x})$, this proves the proposition by contradiction.

Assume $f(\bar{x}) < f(\varphi\hat{x})$. The difference between $\bar{x}$ and $\varphi\hat{x}$ can be expressed as a sum of undirected cycles (Lemma 2). A forward ride r^{α}_{ab} in such a cycle denotes that $\bar{x}^{\alpha}_{ab} > \hat{x}_{ab}$, a backward ride $-r^{\beta}_{cd}$ means $\bar{x}^{\beta}_{cd} < \hat{x}_{cd}$. The sum of the costs of all cycles is equal to the difference between $f(\bar{x})$ and $f(\varphi\hat{x})$. Because of the assumption $f(\bar{x}) < f(\varphi\hat{x})$ we have $f(\bar{x}) - f(\varphi\hat{x}) < 0$ and



ab	2
ac	1
bd	2
cd	1

Figure 6: Flows with fixed charges

therefore the sum of the costs of all the cycles must be negative. This means that at least one cycle must have negative cost.

Each cycle C can be mapped to a cycle in the one-period case, by mapping rides r_{ab}^{α} to rides r_{ab}. Because C may contain a ride r_{ab} in several periods, it might be possible to split the cycle in the one-period case into several undirected cycles. If we map a cycle with negative cost, at least one of the cycles in the mapping must also have negative cost. We can select such a cycle C' with negative cost and modify $\hat{x}$ along C' to get a new train length assignment $x' = \hat{x} + C'$. The cost of x' is

$$f(x') = f(\hat{x}) + f(C'),$$

which is less than $\hat{x}$ because $f(C')$ is negative; the car-conservation equation is not violated because we have modified a valid solution along a cycle. It remains to show that the bound inequations are not violated: this is true because for each $\hat{x}_{ab}$ that has been increased, there must have been at least one $\bar{x}_{ab}^{\alpha}$ that was greater than $\hat{x}_{ab}$, otherwise r_{ab} could not be a forward ride in C. The respective is true for backward rides and decreased $\hat{x}_{ab}$. So it is possible to increase and decrease the number of cars on all the rides where we did, and because we have only changed the number of cars by 1, it is not possible that the bound inequations are violated in x'. So x' is in contradiction to $\hat{x}$ being the optimal solution for the one-period case. Therefore $f(\bar{x}) < f(\varphi\hat{x})$ is impossible, this proves proposition 3. □

With this proof, proposition 2 is only partly proved. If there are fixed cost functions, the proof is not valid. If we compare $\varphi\hat{x}$ to $\bar{x}$ in this case, the difference can still be expressed as a sum of cycles. Mapping those cycles to the one-period case leads again to a set of cycles. Modifying the solution along one of those cycles leads to another valid solution. But as opposed to the linear case, with fixed costs it is possible that there is no cycle that would reduce the cost of the solution. Figure 6 gives an example that shows the difficulty. The example has rides r_{ab}, r_{ac}, r_{bd} and r_{cd} with fixed charges as defined in the figure (to the right). The labels on the arcs represent the number of cars on the respective ride. In the leftmost situation, there are 2 cars on r_{ab} and r_{bd}, the rides r_{ac} and r_{cd} are not used. In this situation we

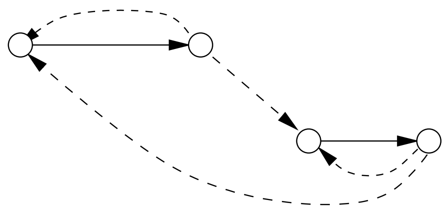


Figure 7: Circular model, repeating every period

have a total cost of $2+2=4$. The cost in the rightmost situation is $1+1=2$. The difference of the two train length assignments can be expressed as two equal cycles ($[-r_{ab}, -r_{bd}, r_{cd}, r_{ac}]$). If we change the leftmost train length assignment along such a cycle, we get train lengths as in the middle situation, with total costs of $2+2+1+1=6$! In this example we see the situation where there is no cycle that reduces the cost, but the combination of all the cycles does. Because we have assumed that the number of cars on each rides is changed at most by one to prove that the bound inequations are not violated, the prove for the linear case cannot be used for the fixed cost case. We assume that a cost-reducing combination of cycles that does not violate the bound inequations can always be found, but we have not yet found a proof for this assumption.

Although we cannot prove it, we will rely on the assumption $f(\varphi\hat{x}) = f(\bar{x})$ below, this allows us to use a model that relies on the one-period problem definition (equations 9 – 12).

3.1.1 The One-Period Model

Figure 7 shows the graphical representation of the Train Length problem from figure 1, this time according to the one-period problem definition (equations 9 – 12). Note that with this graphical representation, it is impossible to see how long a certain ride r_{ab} takes. With the infinite model we could just count the number of periods between a and b to see δ_{ab} graphically. The only thing we can find out here is that arcs going from right to left certainly have $\delta_{ab} > 0$. Figure 8 shows a ride r_{ab} with $\delta_{ab} = 2$. On the left there is the graphical representation for the infinite model, on the right the one-period model. In the infinite model we see the rides s r_{ab}^{n-1}, r_{ab}^{n}, r_{ab}^{n+1}, r_{ab}^{n+2} and r_{ab}^{n+3}, in the one-period model there is only r_{ab}. Note that both representations show the same timetable!

We will see that we can use exactly the same cost functions in both models. Because this is not obvious, we will give some explanations here.

It is clear that the cost function for rides r_{ab} with $\delta_{ab} = 0$ is the same as

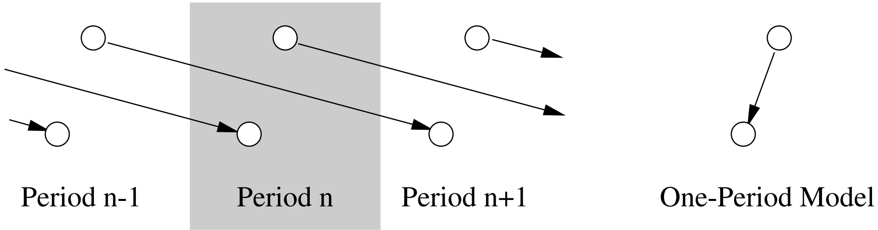


Figure 8: Ride r_{ab} with δ_{ab}=2 in infinite and one-period model

the cost function for r^{α}_{ab}, because for such rides there is no difference in the models. The question is what cost we have to assign to rides r_{ab} with $\delta_{ab} > 0$. We want to minimize the per-period cost and if δ_{ab} is greater than 0, the corresponding ride runs during more than one period. Therefore only part of the costs for r^{α}_{ab} occurs within period α. In our example (figure 8) the cost that occurs in period n is the sum of three partial costs from r^{n}_{ab}, r^{n+1}_{ab} and r^{n+2}_{ab}. But this sum is obviously equal to the cost that occurs for one full ride r^{α}_{ab}. So we can just assign the cost for this ride to r_{ab} in the one-period model.

This shows that we do not have to define special cost functions for the one-period model, we can just use cost functions as in subsection 2.4.

3.2 Fixed Charges and Complexity

Remember the cost functions are of the form

$$f_{ab}(x_{ab}) = \begin{cases} 0, & \text{if } x_{ab} = 0 \\ c_{ab} \cdot x_{ab} + d_{ab}, & \text{if } x_{ab} > 0 \end{cases}$$

where c_{ab} and d_{ab} are nonnegative integers.

Note that for the special case where all d_{ab} are 0, this is a simple linear function. We will now consider the case where the d_{ab} are arbitrary, nonnegative integers, and we will give an $\mathcal{NP}$-hardness proof for that case by reduction from the Steiner tree problem. The following proof is an adaptation of the $\mathcal{NP}$-hardness proof for the Fixed-Cost Transportation problem by Hochbaum and Segev in [5].

We consider the most basic version of the Steiner tree problem: given a weighted graph $G = (V, E)$ (all edge weights positive), in which a subset of k vertices $T \subseteq V$ are identified as *terminals*, find a minimum-weight tree that connects all the terminals. Figure 9 shows an example for an instance of the Steiner Tree problem. Terminal nodes are drawn in black. The nodes q, r, s and t are terminals, a, b, c and d are additional nodes, which do not have to

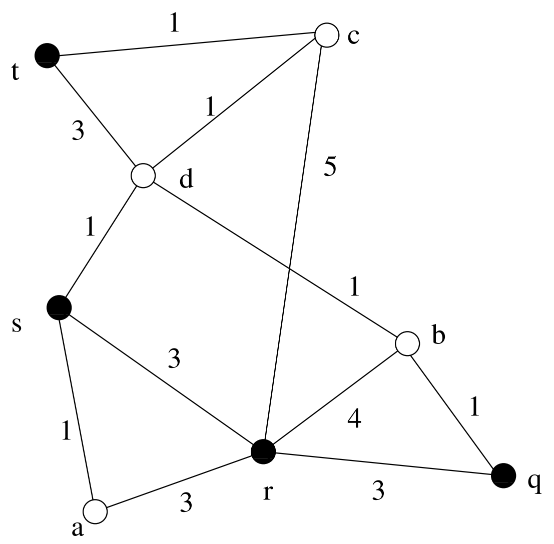


Figure 9: Steiner tree problem (terminal nodes are black)

be part of the tree. The figure shows the weight of each edge, for example the edge (r, q) has weight 3.

We use the graph G to define rides and events. According to edge weights we can define costs for the rides, we will also give rules to define lower and upper bounds for the number of train cars on each train ride. The linear cost c_{ab} will be set to 0 on all rides, we will not mention linear costs below. Figure 10 shows the graphical representation of the reduction from the Steiner Tree problem shown in figure 9. We will now describe the reduction, we will use vertex and ride names according to our example in figure 10. For any terminal vertex $t \in T$ we define a scheduled ride $r_{t_1 t_2}$. The minimum and maximum numbers of train cars are set to $l_{t_1 t_2} = u_{t_1 t_2} = k$ (remember, k is the total number of terminal nodes). For each vertex $v \in V \setminus \{t\}$ we define a scheduled ride $r_{v_1 v_2}$ with $l_{v_1 v_2} = 1$ and $u_{v_1 v_2} = k$ (in the example, this refers to nodes a, b, c, d, q, r and s). All those scheduled rides are free of charge ($d_{v_1 v_2} = 0$). We will refer to rides $r_{v_1 v_2}$ with $v \in T$ as *terminal rides*. The terminal rides in the example are $r_{q_1 q_2}$, $r_{r_1 r_2}$, $r_{s_1 s_2}$ and $r_{t_1 t_2}$. In figure 10, scheduled rides are shown as bold arcs, labeled with the lower bound, the upper bound and the fixed charge. The departure and arrival events of terminal rides are drawn black, the other events are white nodes.

We have to define costs and upper bounds for the empty rides now. To define the costs, we will refer to w_{vw} as the weight on edge $(v, w) = (w, v) \in E$. Table 3 defines the costs and bounds for all types of empty rides that are allowed. Case 1 shows free empty rides $r_{v_2 v_1}$ with $u_{v_2 v_1} = 1$ for non-terminal nodes v. Case 2 defines free empty rides $r_{u_2 t_1}$ with $u_{u_2 t_1} = 1$ from terminal

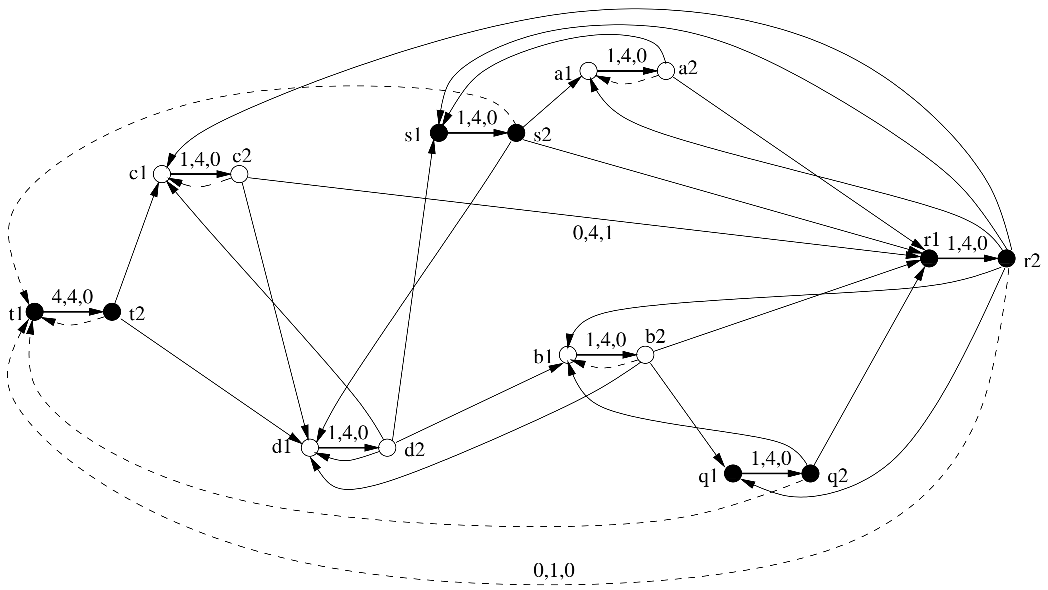


Figure 10: Scheduling instance

	ride	u	d	conditions
1	$r_{v_2v_1}$	1	0	$v \in V \setminus T$
2	$r_{u_2t_1}$	1	0	$u \in T$
3	$r_{v_2w_1}$	k	w_{vw}	$(v,w) \in E, w \neq t$

Table 3: Upper bounds and fixed costs for empty rides

rides to t_1. The free empty rides all have an upper bound of 1, they are all shown as dashed arcs in figure 10.

In case 3 the empty rides according to the edges of the Steiner Tree problem are defined: an empty ride $r_{v_2w_1}$ is allowed if $(v,w) \in E$ and $w \neq t$, the cost is set to w_{vw} and $u_{v_2w_1} = k$. This means that each edge $(v,w) \in E$ that is not adjacent to t defines two empty rides ($r_{v_2w_1}$ and $r_{w_2v_1}$), and that an edge (v,t) defines one empty ride ($r_{t_2v_1}$). To simplify the figure we have not labeled all the arcs, only as an example there is a label on the empty ride $r_{c_2r_1}$ that defines $l_{c_2r_1} = 0$, $u_{c_2r_1} = 4$ and $d_{c_2r_1} = 1$.

The empty rides that are not allowed are the following:

- empty rides between scheduled rides without a corresponding edge in E,
- empty rides from non-terminal rides to t_1
- empty rides $r_{v_2v_1}$ for terminal nodes v except t

Proposition 4 *The minimum cost solution x of the Train Length problem defines the minimum weight Steiner tree: every $x_{v_2w_1} > 0$ yields an edge (v,w) that belongs to the minimum weight Steiner tree.*

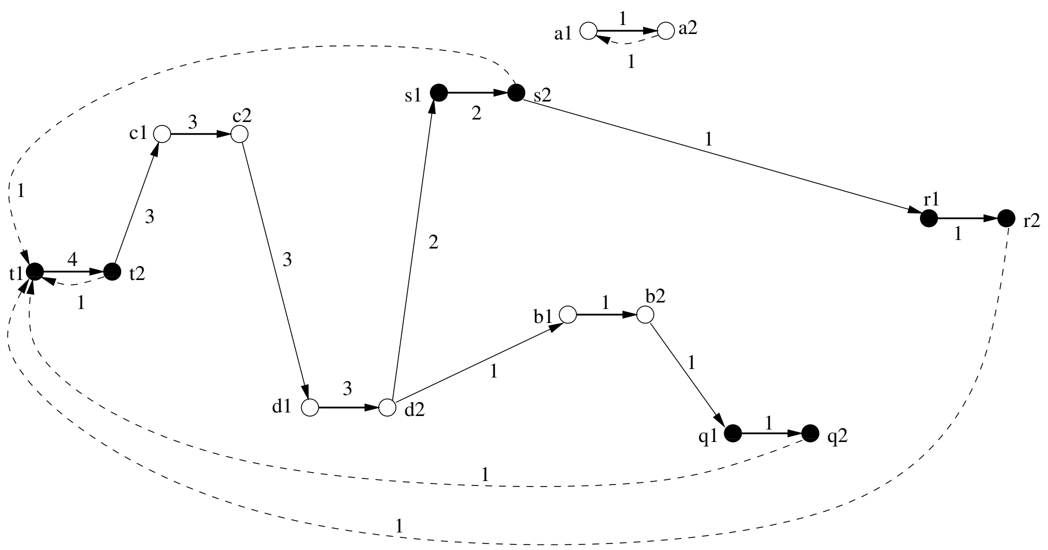


Figure 11: Scheduling solution

Before we prove this, imagine what the solution of our Train Length problem reduction will look like. Figure 11 shows the solution to the Train Length problem instance, only arcs for rides r_{vw} with $x_{vw} > 0$ are shown. The labels on the arcs denote the number of train cars on the respective ride. We have k train cars on ride $r_{t_1t_2}$ and there are k empty rides arriving at t_1 with at most one train car each. Therefore $x_{v_2t_1} = 1$ for all $v \in T$ (q, r, s and t in the example). Those rides come from the k terminal rides. Those k train cars satisfy the demand on all terminal rides and potentially on some other scheduled rides $r_{b_1b_2}$. For the non-terminal rides $r_{a_1a_2}$ where the demand ($l_{a_1a_2} = 1$) is not satisfied with one of the k train cars, there will be a separate car for $r_{a_1a_2}$ and this train car will use the free ride $r_{a_2a_1}$ to be ready for the next period. Those separate cars do not influence our tree. Let us now prove the proposition.

Proof: First we will prove that the routes of the k train cars on $r_{t_1t_2}$ define a minimum weight subgraph connecting all the terminal nodes in G. Once each period, k cars are standing in s_{t_1}, the station of event t_1. It is not possible that any train car except one of the cars on $r_{t_1t_2}$ will ride to one of the terminal rides because all the other cars only operate one non-terminal ride. Operating one non-terminal ride only[1] can be done for free while transferring a car from a non-terminal ride to any other ride would cost something[2], therefore it is not possible that any train car except one of the cars on $r_{t_1t_2}$ will ride to one

[1]like $r_{a_1a_2}$ in figure 11

[2]because all edge weights in G are positive

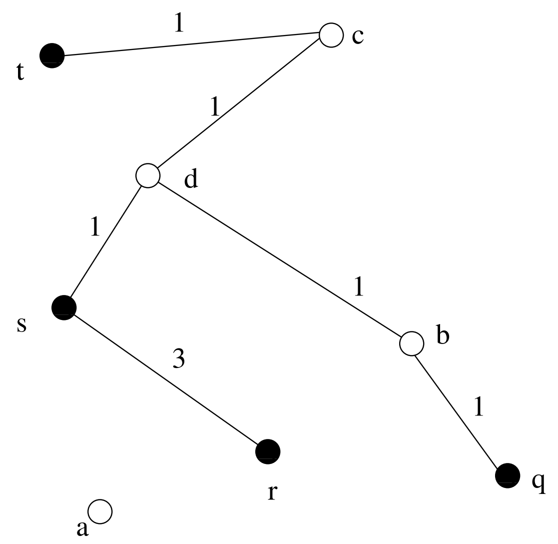


Figure 12: Steiner tree solution

of the terminal rides. Therefore each of the $k-1$ terminal nodes in $T \setminus \{t\}$ is connected to t_2 (each of the $k-1$ cars defines a route from t_2 to one of the terminal rides). Now we prove that the induced subgraph is of minimum weight. Suppose there exists a subgraph connecting all nodes in T with less weight. If our train cars would just use the rides induced by this subgraph, the Train Length problem would also be solved, but with less cost. This is in contradiction with the assumption that the first subgraph has been induced by an optimal solution to the Train Length problem.

One small step remains: we have to show that the minimum weight subgraph that connects all the terminal nodes is a tree. But this is obvious: A subgraph connecting all the terminal nodes that contains cycles cannot be of minimal weight, because by removing one edge of a cycle we would get a connected subgraph of less weight (if all the edge weights are positive, which we have assumed). A tree is defined to be a connected, cycle-free graph and we have just proven this feature in our subgraph. $\square$

Figure 12 shows the Steiner tree according to the train length assignment from figure 11. The labels on the edges denote the edge weights.

4 Network Flow Models

It is a well known practice to solve scheduling problems using network flow models. Brucker, Hurink and Rolfes [3] and schrijver [4] use network flow models to solve Train Length problems.

Our problem definition (equations 9 – 12) is similar to the Minimum Cost Flow problem without sources and sinks, this problem is also known as the Minimum Cost Circulation problem. We will model the Train Length problem as a Minimum Cost Circulation problem.

The network graph for the Minimum Cost Circulation problem can be defined in a straightforward way: Vertices represent events (arrivals and departures), arcs represent riding or waiting train cars. Minimum and maximum numbers of train cars on rides are modelled with lower and upper bounds for the flow on the respective arc, ride costs are modelled as cost on the arcs. The restriction that the number of arriving train cars must be equal to the number of departing train cars is easily met by defining the supply of each vertex to be 0. The solution defines the flow on each arc, which can be interpreted as the number of train cars on each ride. We will refer to the graph as $G(V, A)$, where V is the set of vertices and A is the set of arcs.

If we restrict the cost function to linear costs ($f(x) = c \cdot x$), our problem can be stated as a standard Min Cost Circulation problem:

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{13}$$

s.t.

$$\sum_{i:(i,j)\in A} x_{ij} - \sum_{k:(j,k)\in A} x_{jk} = 0 \qquad \forall j \in V \tag{14}$$

$$l_{ij} \leqslant x_{ij} \leqslant u_{ij} \qquad \forall (i,j) \in A. \tag{15}$$

The objective function (13) corresponds to the objective function (9) in the original problem definition, equation (14) corresponds to the car-conservation equation (10) and inequations (15) correspond to the bound inequations (11). In the original problem definition, there was an integrality constraint (12), in the Min Cost Circulation formulation such a constraint is not needed; optimal solutions to the Minimum Cost Flow problem are always integral (see [1]).

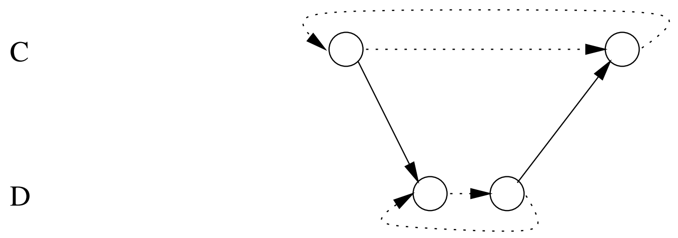


Figure 13: Network graph, 2 stations, 2 scheduled rides

4.1 The Model

We will develop the network model step by step. The model is similar to the one proposed by Brucker, Hurink and Rolfes [3].

First, we develop a model that considers linear costs only, we will include fixed costs later.

When we develop the model, we will define events and rides. The scheduled rides are given, so we already have a certain set of events and rides that cannot be changed. For further events and rides, we will try to add as many as necessary but as few as possible, because the size of the network influences the running time of Min Cost Circulation algorithms.

4.1.1 Scheduled Rides

As stated above, nodes represent events, so we introduce a node for each scheduled arrival and for each scheduled departure. In the graphical representation, we draw time from left to right and stations from top to bottom (by some arbitrary ordering). If there are several events at the same station and time, one node is sufficient for all of them. Now we add arcs for the scheduled rides r_{ab}, connecting the events a and b.

4.1.2 Waiting Arcs

Also, we have to add arcs for train cars waiting in a station. We will refer to those arcs as *waiting car arcs.* Waiting car arcs connect each event to the next event in the same station. The last event of each station is connected to the first one (because we have a periodical schedule, the first event node also represents the first event of the next period). Figure 13 shows an example, waiting arcs are drawn as dotted lines. The example schedule consists of two scheduled rides: $C \to D$ and $D \to C$.

Before we consider empty train movements, let us have a look at the current model. This model seems to be a valid network formulation for the case where empty train movements are disallowed: l_{ab} and u_{ab} can be set for

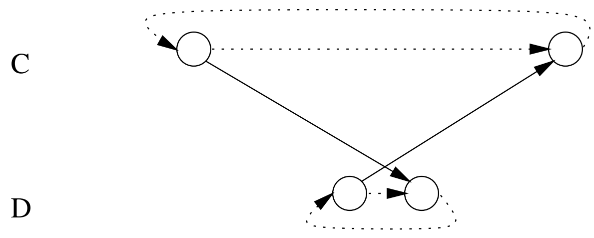


Figure 14: Adjusted arrival and departure times

each scheduled ride r_{ab}, it is even possible to define a maximum capacity for each station by defining the upper bound on waiting car arcs accordingly. The lower bound on waiting arcs is set to 0. We can define a cost function for each ride and the Minimum Cost Circulation model minimizes total cost. But unfortunately, already this model has it's problems. Between an arrival and the departure of the next ride with the same car, a certain amount of time is required. In our model, a train car could operate two trains even if the first one would arrive at the same time as the second departs. We can easily adapt our model to respect the required buffer times before and after each ride: We just consider those buffer times to belong to the respective ride and change the arrival and departure times accordingly. We refer to the buffer time needed before the departure of ride r_{ab} as $b_{\text{dep}}(r_{ab})$ and to the buffer time needed after the arrival of r_{ab} as $b_{\text{arr}}(r_{ab})$. For our model, ride r_{ab} departs at $t_a - b_{\text{dep}}(r_{ab})$ and arrives at $t_b + b_{\text{arr}}(r_{ab})$. If we recreate the network graph with the modified departure and arrival times, a train car can be used directly after its buffered arrival and the model is correct. Figure 14 shows the modified version of figure 13, assuming that the available time between rides $C \to D$ and $D \to C$ was smaller than the required buffer time. Only stations C and D are shown.

With this change we can assign individual buffer times to each ride, this has the advantage that buffer times could for example be adapted to the expected number of passengers on the train.

4.1.3 Empty Ride Arcs

Now that we have a good model for the case where no empty rides are allowed, let us add arcs for empty rides. Figure 15 shows an example for the network we have defined above. There are 3 scheduled rides connecting the stations A and B (solid arcs) and 6 waiting rides in the stations (dotted arcs). We will add arcs for empty rides to this network.

Empty ride arcs have a lower bound of 0 and an upper bound according to the route. Costs can be assigned as desired by the objective of the

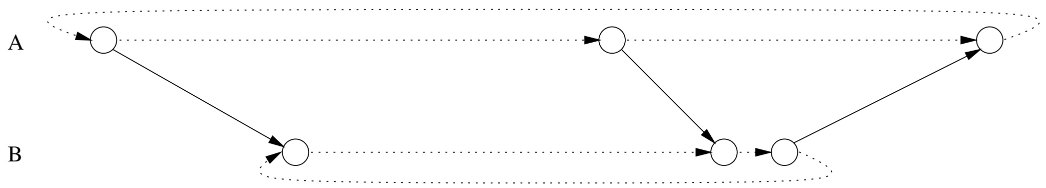


Figure 15: Network graph, for 3 scheduled rides, without empty rides

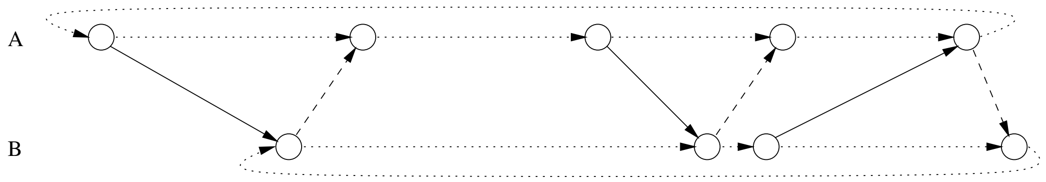


Figure 16: Empty car arcs added after arrivals

optimization.

In theory, empty train movements are possible at any time. But we do not want to add so many arcs to our network, therefore we have to find a reasonable set of potential empty rides.

After each arrival, the train cars either wait in the station or they are transferred to another station with an empty train. According to assumption 2, train movements between any pair of stations are possible, so we should add empty car arcs from each arrival node to each station different from the arrival station. The empty car arcs can be incident to the arrival nodes because we have already delayed arrival events by the required buffer times. If the empty train movement also needs buffer times before the departure or after the arrival, those times have to be considered when the arrival time of the empty ride is evaluated. In general, the arrival event e_1 of an empty train will fall between two other events e_0 and e_2 at the respective station. In this case, we have to add a new node to the network and split the respective waiting arc ($e_0 \to e_2$) to into two arcs ($e_0 \to e_1$ and $e_1 \to e_2$). Figure 16 shows the network model for the same schedule as figure 15, but with arcs for empty rides (dashed arcs).

Let us now discuss the new model. A circulation in the network yields a valid train length assignment, we have considered the required buffer times and all other constraints. It seems that each arriving train car can be moved to any station because there is an empty ride arc from each arrival node to each station. But, at least in theory, this is not true: if an arriving scheduled ride r_{ab} has more train cars than the upper bound of some ride r_{bc}, it is not possible to transfer all train cars of r_{ab} to s_c right after the arrival of r_{ab}. The same problem occurs if there are several scheduled rides arriving at the same

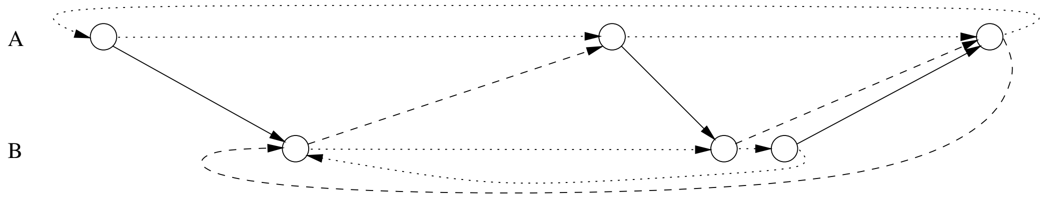


Figure 17: Empty ride arcs from arrivals to scheduled events

time in the same station. But this is only a theoretical problem: as we have seen in 2.1.6, empty trains can be longer than scheduled trains, therefore we will not change our model due to this problem.

By adding empty car arcs we have created new arrival events in our model, should we not add empty train arcs connecting those arrivals to each station? We assume that an empty train from A to B always takes the cheapest route. This means that the cheapest possibility for a train car that arrives with a scheduled ride in A and that will operate a scheduled ride departing from B, is to use an empty train movement from A to B and not several movements that could transfer the car from A to B. Note that these observations are only true for linear cost functions. Because in the linear case there will never be a car that uses two empty rides in sequence, we do not have to add more empty ride arcs. We will come back to this when discussing models for the Train Length problem with fixed costs.

But the network graph is not only large enough, it is too large. The nodes that have been added for the empty ride arrivals are not required: The network graph allows the same solution if we connect the empty ride arc to the first scheduled event node *after* the arrival of the empty ride, as shown in figure 17. So the number of nodes is at most $2n$ and the number of arcs is at most $n+2n+n\cdot(m-1) = n(m+2)$ (there are n arcs for scheduled rides, at most $2n$ waiting arcs and at most $n\cdot(m-1)$ empty ride arcs). We will normally use this network for the Minimum Cost Circulation problem. This means that we define E to be the set of scheduled events and that we allow empty rides from scheduled arrivals to scheduled events. Empty rides from departure nodes and empty rides between unscheduled events are not possible.

4.1.4 Time complexity

There are many algorithms to solve the Min Cost Circulation problem. Some of them are favorable to sparse graphs, some to networks with small costs. Many of them have time complexities depending on $U = \max\{u_{ab}\}$ and $C = \max\{c_{ab}\}$. We can assume U and C to be small constants (in the

S-Bahn Zürich example, we have chosen $U = 20$ for waiting car arcs and C can always be chosen freely, larger C allow for finer cost functions). The *double scaling algorithm* ([1], pp. 373) solves the Min Cost Flow problem in time $\mathcal{O}(|A| \cdot |V| \log U \log(|A|C))$. In our case, where $|V| = \mathcal{O}(n)$ and $|A| = \mathcal{O}(mn)$, this evaluates to $\mathcal{O}(mn^2 \log(mn))$. In our S-Bahn Zürich example, n is about 800 (if we assume daily periods) and m is about 40.

4.2 Applications

We now have a model where we can work with linear costs. This allows some important optimizations. With linear cost functions we can for example minimize the number of cars, the total distance or the time spent waiting and on empty rides, as we have seen in subsection 2.4. Such optimizations may be helpful to estimate the quality of a given solution.

It is clear that it is also possible to assign monetary cost functions to each arc, but because only linear costs are allowed, it is not possible to get a very realistic model.

4.3 Fixed Costs

The real goal of a railway optimization is to minimize monetary cost; not just the number of train cars or the total distance. With the network model we can freely assign linear cost functions to arcs. But as stated above, linear functions are not very realistic for monetary cost. There are many costs like driver cost that occur only once per ride, independent on train length. In 3.2 we have shown that the Train Length problem gets $\mathcal{NP}$-hard as soon as we include fixed costs. But there are several suggestions on how to solve fixed cost flow problems. Arlt [2] proposes a branch and bound solution, Brucker, Hurink and Rolfes [3] use a method based on simulated annealing. In a first step we can therefore just use the same network graph as before and assign fixed cost functions to the arcs. We can hope that some of the methods are able to solve our problem. But it is a good idea to review the network graph and to check whether our graph is fine for fixed costs also — it isn't.

We will see below that the network model defined above does not guarantee that the optimal solution for the fixed cost case can be found. If cost functions are only linear, the model offers all the empty rides that are used by the optimal solution. But with fixed costs, this is not the case. We will discuss the problems below. At the end of this discussion, we will decide to use the same network model as for the linear case, in spite of those problems.

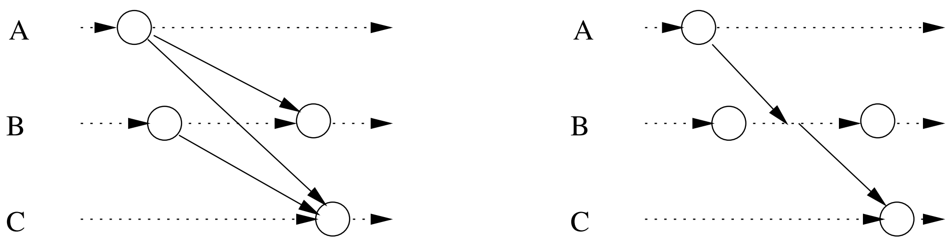


Figure 18: Left: our model, right: optimal solution

The reason for this surprising decision is that some of the problems cannot be solved with network models or only with an extremely enlarge the network.

4.3.1 Empty Rides

We have mentioned above that in the linear case, the best variant for an empty car that has to be transferred from A to B is always a direct empty ride $A \to B$. If fixed costs are allowed, there may be better possibilities: The car could be attached to already existing trains (scheduled or empty) to do the trip from A to B. When fixed costs are allowed, the cheapest possibility for a train car to be transferred from A to B could be any combination of empty and scheduled rides that enable the car to get to B within the required time interval. Imagine the situation of figure 18: There are four events in three stations (A, B and C). We assume that empty rides are possible from the event in A to the second event in B and to the event in C and that an empty ride is possible from the first event in B to the event in C. On the left part of figure 18 we see how our network graph will look like. Assume that our solution uses the empty train movements $A \to C$ and $B \to C$. But remember, empty train movements are possible at any time and therefore the optimal solution might be an empty train movement from A to B where the other car waits and the two cars ride from B to C together (as shown in figure 18 on the right). We could come back to our idea from above and add empty ride arcs from each empty ride arrival to every station. But this would again generate a lot of new arrival events and we would again have to add empty ride arcs, ... Indeed, if we want to allow for arbitrary combinations of empty and scheduled trains that transfer train cars from A to B, we would have to add extremely many empty ride arcs. It is clear that the runtime of algorithms for this problem depends on the network size and in our case (fixed charges) the runtime is not even polynomially bounded. So enlarging the network is probably not too good an idea.

We have to come back to our first idea and use the network from the linear case and assign the fixed cost functions to those arcs. If we do so, we

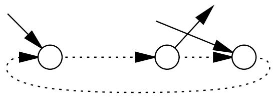


Figure 19: Arcs for waiting cars

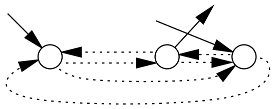


Figure 20: Arcs for waiting trains

must be aware that the solution is only an approximation that can be used as a basis for fine-tuning.

4.3.2 Fixed Costs on Waiting Arcs

Above we proposed to assign fixed cost functions to the arcs to get a model that takes fixed charges into account. If a train has to wait in a station, there are also fixed charges occurring, independent on the number of cars on the train. But the waiting car arcs in our model only represent *waiting cars* instead of *waiting trains*. If only linear cost functions are allowed, it is sufficient to know the number of cars in a station, but with fixed costs we should have arcs for waiting trains. Figure 19 shows an example for a station with two arrivals and one departure. The arcs for waiting cars connect each event to the subsequent one.

If we need arcs for waiting trains, we need arcs between any pair of events where a train could start and stop waiting. Waiting begins after the arrival of a train and ends when a ride departs with the respective cars. In arrival nodes, there is an arrival of a scheduled ride, there are departures of empty rides and there may be empty ride arrivals. In departure nodes, there is the departure of a scheduled ride and there may be arrivals of empty rides. This means that a train can start and stop to wait in any node. So waiting arcs between each pair of nodes of the same station are required. Figure 20 shows the same timetable as figure 19, this time with arcs for waiting trains.

In 2.4.2 we have seen that we can count coupling and decoupling with fixed cost functions on waiting train arcs. But this is only possible if for each each event e either $|\operatorname{arr}(e)| = 1$ or $|\operatorname{dep}(e)| = 1$, which is not the case in our network. So one important application for fixed costs on waiting train arcs cannot be used with our network. We could think about modifying the network, such that there is only one arrival or only one departure per ride. If we allow only the scheduled arrival in arrival nodes and only the scheduled departure in departure nodes, empty and waiting rides are only possibe from arrival to departure nodes. But in such a network, each car uses exactly one empty or waiting ride between two scheduled rides, this is not what we want (we want to allow combinations of empty and waiting rides between scheduled rides).

Also, the number of arcs would be increased drastically by choosing empty train arcs instead of empty car arcs (from $\mathcal{O}(mn)$ to $\mathcal{O}(n^2)$). For schedules of the size of our S-Bahn Zürich example (about 800 rides and 40 stations), the network would be enlarged by a factor of about 20. This is certainly a problem for algorithms with runtimes that are not even polynomially bounded.

Having waiting train arcs instead of waiting car arcs has the big disadvantage of enlarging the network size and it does not even allow to model coupling and decoupling costs. Again, we decide to use the network from the linear case. With this network we only solve the pure Train Length problem: Only the lengths of scheduled and empty trains are assigned, everything that happens within stations has to be solved in a second step. In this second step the exact routes for each car are defined in order to satisfy the train length assignment. We will shortly discuss the Train Car Routing problem in section 6.

4.4 Network Size

We have chosen to use the same network for the linear and the fixed charge case. The network consists of at most $2n$ nodes and $n + 2n + n \cdot (m-1) = n \cdot (m+2)$ arcs (n scheduled ride arcs, $2n$ waiting car arcs and $n \cdot (m-1)$ empty ride arcs). We expect problem sizes in the order of 1000 rides and 50 stations, this leads to graphs with about 50'000 arcs. We have a C++ implementation that uses LEDA to solve the linear Min Cost Circulation problem. On a Sun Ultra 10, the algorithm takes a few seconds (2 – 10, depending on cost functions) to solve this problem.

We have not implemented an algorithm for the fixed cost case. Brucker, Hurink and Rolfes [3] have solved a Train Length problem with fixed charges for a timetable that led to a network of about 1000 arcs, it took them 20 minutes on a slow PC (486, 33 MHz). Although PCs are faster today, we cannot expect to be able to solve networks that are 50 times larger with algorithms not having running times polynomially bounded in network size.

It is clear that the running time of fixed cost flow algorithms mainly depends on the number of fixed cost arcs. Fixed costs on scheduled rides do occur, but we can leave them away because they just shift the cost of a

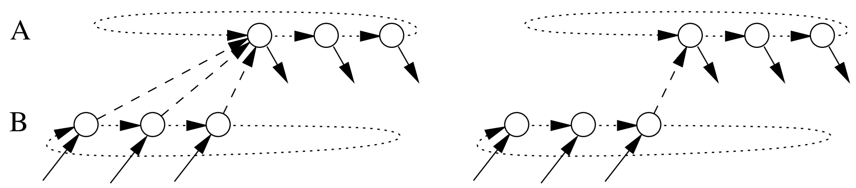


Figure 21: Our network (left) and the version of Brucker et. al. (right)

solution by a constant, as the flow on a scheduled ride arc is never 0. Fixed costs on waiting car arcs make no sense, as we have seen in 4.3.2. So we "only" have to consider empty ride arcs. But we have $n \cdot (m-1)$ empty ride arcs, which is still in the order of 100'000 for our problems.

We have to reduce network size somehow, if we want to use a fixed cost network flow algorithm. Brucker, Hurink and Rolfes [3] reduce the network size by not allowing empty rides r_{ab} in case there is also an empty ride $r_{a'b}$ with $s_a = s_{a'}$ and $t_a < t_{a'}$. Figure 21 shows a situation in the two stations A and B: there are three scheduled arrivals in B and later three scheduled departures from A. The empty ride departing from B after the last arrival reaches A before the first departure. Our model (on the left) allows empty rides from each arrival, Brucker, Hurink and Rolfes only allow the last one. If there are many situations as in figure 21, this can reduce network size, but if there are many situations of that kind, this is also a problem. It is possible that all the cars that arrive in B are used on the trains departing from A. Transferring all those cars using a single empty train ride would require huge empty trains. It is clear that we could split empty trains that are too long into several trains, but then assigning fixed charges to empty rides would not make much sense.

We have another proposition to reduce the number of empty ride arcs (although we have to admit that this proposition has not been tested by now). We propose to solve the Train Length problem with linear cost functions to get a feasible solution x. We assume that we can find linear cost functions such that the train length assignments are pretty good approximations (this assumption is supported by the results of our tests, see section 5). From x we can extract all routes (that is, pairs of stations) where there are empty rides r_{ab} with $x_{ab} > 0$. We can then create a new network by only allowing empty rides on those routes. This will considerably reduce network size and it is sure that there exists a legal realization to the Train Length problem with the new set of empty rides. On this smaller network, we can run a Fixed Cost Flow algorithm and the solution will not be worse than x, because x is a valid flow for the smaller network.

4.5 Conclusion

To minimize linear cost functions like the total distance or the number of train cars, the network approach allows to find optimal solutions. For all other cost functions, we can only find approximations. We propose to use linear cost functions to find an initial solution, based on which we can then define a network that should be small enough for fixed cost flow algorithms.

5 Results

To test our network model we have implemented a small C++ program. The program uses LEDA to solve the Minimum Cost Circulation problem with linear costs only. The data we are using is the real timetable of the entire S-Bahn Zürich system.

5.1 Data Preparation

From the timetable we received from SBB we have extracted the trains that are running on Mondays to Fridays, this has two reasons:

- On these days, we have the largest number of trains and the most passengers, and
- from Monday to Friday the same schedule is repeated five times, this is the best approximation of a periodic schedule we can find (if we look at daily schedules).

Our data from SBB is not just the timetable containing the scheduled trains but it is SBB's solution to the Train Length problem. The timetable defines all the scheduled rides and all the empty rides that are used. For the scheduled rides we were given a table with expected passenger numbers.The timetable consists of 835 scheduled and 96 empty rides. To compare the solutions of our approach to the SBB solution we will use the 835 scheduled rides and the passenger demand as input to our program and compare the two solutions.

When comparing our solution to the SBB solution, we will have to do so with respect to a certain cost function f. We will refer to SBB's solution as x_{SBB} and to our solution as x.

A comparison of $f(x_{\text{SBB}})$ to $f(x)$ would be unfair because SBB's schedule includes rides to and from maintenance stations. We have data about maintenance that says when and where maintenance happens (there are 260 maintenance operations each day). For each maintenance operation in station M, we define a "maintenance ride", a ride with departure and arrival in M and with departure and arrival times as defined in the maintenance data. Those 260 maintenance rides are added to the 835 scheduled rides. We use this new set of 1104 rides as input to our program. With this input, a solution will satisfy passenger demand on the scheduled rides and it will also have as much maintenance as the solution from SBB. It is clear that maintenance rides are only a compromise: flexible maintenance times and places could allow for better solutions, but on the other hand we can not guarantee that in our solution each car gets maintained (it is possible that only some of the cars are maintained). Also, if our solution uses more or less

	train cars	distance (km)	empty rides
x_{SBB}	123	55'953	96
lower bounds	118	44'184	6
x_{123}	123	44'379	49
x_{122}	122	44'403	50
x_{121}	121	44'436	51
x_{120}	120	44'476	52
x_{119}	119	44'514	56
x_{118}	118	44'569	57

Table 4: Comparison of solutions

train cars than x_{SBB}, then the amount of maintenance is not appropriate. We will therefore only consider consider solution that use at most as many train cars as x_{SBB}.

5.2 Test Runs

Because we have no access to SBB's cost functions we will compare solutions by the number of train cars, the total distance and the number of empty trains that are used. Table 4 shows some examples. In the first row we see the solution that is currently used by SBB, they need 123 train cars and 96 empty train rides. We have computed lower bounds for the number of train cars, the total distance and the empty rides. The lower bounds are shown in the second row. The lower bounds for the total distance and the number of empty rides are reached with train length assignments that require 204 and 183 train cars respectively.

Because we only allow train length assignments that use at most as many train cars as x_{SBB} we have to find solutions with 118 to 123 train cars. We use cost functions that combine costs for the number of train cars and costs for the distance. Although in reality the cost for riding an empty train is less than the cost for a scheduled train, we assign higher costs to empty trains. With such cost functions, empty car movements will only require an empty train if the car cannot be attached to a scheduled ride. If the cost for empty rides were less than the cost for scheduled rides, every empty car movement would use an empty train.

As a rule of thumb we can say that the more train cars there are, the less empty rides are used. Therefore it is possible to find solutions with different numbers of train cars by balancing the cost functions: By assigning high costs to empty rides we find solutions that use few empty rides, by assigning high costs to overnight trains we find solutions that use few train cars.

In table 4 we list a solution for each number of cars from 118 to 123. It is very surprising to see the large difference between our solutions and x_{SBB}: our solutions have riding distances of about 80% of x_{SBB} and they use less than 60% of the empty rides.

This is very astonishing because we are only using linear cost functions. It is clear that for minimizing the number of train cars and minimizing the total distance, linear cost functions are sufficient, but with respect to those costs, our solution and x_{SBB} are similar, but our solution is much better with respect to empty train rides. With the use of an algorithm for Fixed Cost Flow problems as proposed in 4.4 or by using an Improvement Algorithm as we will propose in section 6, we expect to get even better solutions.

We are aware that there might be some constraints that are met by x_{SBB} that are not met by our solution and that our cost functions are extremely simplified compared to those that SBB uses. But we expect that our solutions can be used as a base solution for human experts. To meet all the constraints, empty rides may have to be added to the system; but the human experts may add many empty rides without getting worse solutions than x_{SBB}.

We assume that the number of train cars, the number of empty rides and the total distance are the most important cost factors and therefore our solutions will also be better than x_{SBB} for the more sophisticated cost functions used by SBB.

6 Future Work

In this section we would like to give brief descriptions of what could be done based on our results.

6.1 The Train Car Routing Problem

Given a solution to the Train Length problem, we still have to define exact routes for each car. Routes for cars are found by dividing the train length assignment into cycles; each cycle defines the route of a car. Such a route may last several periods, for each cycle we need as many cars as it takes periods to ride the full cycle (because each ride on the cycle must be operated in every period).

When defining the cycles, several aspects have to be considered:

- each cycle should be connected to the required maintenance stations, otherwise it would not be possible to maintain the cars on such cycles.
- Cycles with common rides should be defined such that their common paths are as long as possible, this minimizes coupling and decoupling.
- Cycles should balance the load, such that all the cars drive about the same distance a day, independent on the cycle they are on.

Optimal car routing on the optimal solution to the Train Length problem does not necessarily lead to the optimal train car routing for the given timetable. For example it is possible that the train length assignment requires so many coupling and decoupling operations that there exists a cheaper car routing for a different train length assignment. It would be interesting to discuss algorithms that do the whole car routing in one step or, as a variant, algorithms for the Train Car Routing problem that are allowed to modify the train length assignment.

6.2 Improvement Algorithms

The use of an improvement algorithm could be an interesting (and much faster) alternative to using a Fixed Cost Flow algorithm. In 4.4 we have proposed to define the network for the Fixed Cost Flow algorithm based on a solution for the linear case.

We propose another approach here: based on a train length assignment x for the linear case, perform improvement steps to get a better solution instead of running a Fixed Cost Flow algorithm. The main problem of a linear case solution is that it uses too many empty rides. Based on the following idea, it is possible to define several improvement algorithms: We define the length of a path between two nodes in the network as the cost that would occur if we would add one empty car to the rides along this path. If we add one car to a ride with $x_{ab} = 0$, the additional cost is $c_{ab} + d_{ab}$, for rides with $x_{ab} > 0$ it is only c_{ab}. If adding a car to a path is not allowed because this would violate the bounds, we define the length of the path to be ∞. We will refer to the shortest path from a to b as $s_{ab}(x)$. $s_{ab}(x)$ is a sequence of rides $[r_{ac}, r_{cd}, \ldots, r_{gh}, r_{hb}]$.

Starting with any valid train length assignment x, we can run the following algorithm:

$f_{old} \leftarrow \infty$
while $f_{old} > f(x)$ **do**
  $f_{old} \leftarrow f(x)$
  for all $r_{ab} \in R$ **do**
    if $x_{ab} > l_{ab}$ **then**
      $x_{ab} \leftarrow x_{ab} - 1$
      for all $r_{ij} \in s_{ab}(x)$ **do**
        $x_{ij} \leftarrow x_{ij} + 1$
      end for
    end if
  end for
end while

From a ride r_{ab} we remove one car and add it to the rides on the shortest path from a to b, this gives us a better solution, unless the shortest path is $[r_{ab}]$. We can repeat this step until a local minimum is found (no further improvement is possible).

This approach allows for every kind of cost function and constraint, it is even possible that paths have negative weights, for example if fewer coupling and decoupling is required.

A variant would be to remove a car from a *sequence* of rides and then to find the shortest path from the first to the last event of this sequence. It is clear that removing a car from a sequence of rides is only allowed if this does not violate the bound inequations.

As with all improvement algorithms, the advantage of this approach is that the solution only gets better while the algorithm is running. It is also possible to perform only a certain number of steps and stop before a local minimum is found, every intermediate solution is a valid train length assignment.

The quality of the final solution depends on the order in which rides are chosen to remove cars. There are many variants: random, expensive rides first, cheap rides first, rides with many cars first, ...

6.3 Train Types

The Train Length problem is defined for one type of trains only. We are also interested in optimizing full schedules with rides for several train types[3]. We assume that for each ride we are given the minimum number of cars of each type and whether a locomotive is required.

Brucker, Hurink and Rolfes [3] use a Multicommodity Flow formulation for the case of several train types. Because timetables for many types of trains are larger than timetables for one type of train, it would be interesting to evaluate another approach.

Our idea is to split the timetable into timetables for each type of car and to solve the Train Length problem seperately for each type. Then we can join the solutions of car types that can be coupled and run an improvement algorithm on those set of rides. Finally we have to assign locomotives where required: this is a new Train Length problem, whith all the rides r_{ab} with $x_{ab} > 0$ as scheduled rides. If different types of locomitives are used for different types of trains it is again possible to split the timetables.

[3]like intercity and regional trains

7 Conclusion

Although we are using a very simplified model for the Train Length problem, our solutions seem to outrange current solutions used by SBB. Currently, SBB solves the Train Length problem "by hand" and their experts spend a lot of time to find the train length assignment for next year's schedule.

Our train length assignments that are valid for a simplified model could be used as a basic solution for the experts at SBB, they could then use there skills to modify this solution in order to meet all the constraints.

With our model it would also be possible to get first hints about the quality of a new timetable. Given a new timetable we could for example find out the minimum number of train cars that would be required to operate all the trains. If this number of cars is not available, the experts can refuse the new timetable immediately without having to check the new timetable for hours or even days.

We expect that a computer program basing on our propositions could greatly support the work of the experts. Because we can only optimize a very simplified model, the experts could then concentrate on the real hard problems and leave the simple things to the program.

7.1 Credits

I would like to thank SBB, especially Mr. Strüby, for giving me much of their data and for helping me understanding them. Mr. Strüby has patiently answered lots of questions (also stupid ones) and he was a great help.

A "thank you" also to my advisors, Peter Widmayer and David Taylor from the Institute for Theoretical Computer Science at ETH Zürich, they always supported me whenever I needed help and their support helped me to resolve my doubts in the middle of the work.

References

[1] Ahuja R.K., Magnanti T.L. and Orlin J.B., *Network Flows.* Prentice Hall Inc., Englewood Cliffs, New Jersey, 1993

[2] Arlt Ch., *Die Lösung grosser Fixkosten-Netzwerkflussprobleme.*, Gabler Verlag, Wiesbaden, 1994

[3] Brucker P., Hurink J.L. and Rolfes T., *Routing of railway carriages: A case study.* University of Twente, Enschede NL, 1999

[4] Schrijver A., *Minimum circulation of railway stock.* CWI Quarterly 6(3), 205 – 217, 1993

[5] Hochbaum S. and Segev A., *Analysis of a Flow Problem with Fixed Charges.* NETWORKS, Vol. 19, 291 – 312, John Wiley & Sons, Inc., 1989