Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

CHARLES BRETT[1]
REBECCA HOBERG[2]
MERITXELL PACHECO[3]
KYLE SMITH[4]

RALF BORNDÖRFER[5] RICARDO EULER[5],
GERWIN GAMRATH[5] BORIS GRIMM[5],
OLGA HEISMANN[5] MARKUS REUTHER[5],
THOMAS SCHLECHTE[5] ALEXANDER TESCH[5]

G-RIPS 2014 RailLab
–
Towards robust rolling stock rotations

[1] University of Warwick, Coventry CV4 7AL, UK c.brett@warwick.ac.uk
[2] University of Washington, 6221 25th Ave NE, 98115 Seattle, WA, USA rahoberg@math.washington.edu
[3] Universitat Politcnica de Catalunya - Barcelona Tech, Building U. C. Pau Gargallo, 5 08028 Barcelona, Spain meritxell.pacheco@gmail.com
[4] New Mexico Tech, PO Box 3384, 87801 Socorro, NM, USA ksmi02@nmt.edu
[5] Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, Germany {surname}@zib.de

ZIB-Report 14-34 (September 2014)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

G-RIPS 2014 RailLab

—

Towards robust rolling stock rotations

Charles Brett, Rebecca Hoberg, Meritxell Pacheco, Kyle Smith
Ralf Borndörfer, Ricardo Euler, Gerwin Gamrath, Boris Grimm
Olga Heismann, Markus Reuther, Thomas Schlechte, Alexander Tesch

September 27, 2014

Abstract

The Graduate-Level Research in Industrial Projects (G-RIPS) Program provides an opportunity for high-achieving graduate-level students to work in teams on a real-world research project proposed by a sponsor from industry or the public sector. Each G-RIPS team consists of four international students (two from the US and two from European universities), an academic mentor, and an industrial sponsor.

This is the report of the Rail-Lab project on the definition and integration of robustness aspects into optimizing rolling stock schedules. In general, there is a trade-off for complex systems between robustness and efficiency. The ambitious goal was to explore this trade-off by implementing numerical simulations and developing analytic models.

In rolling stock planning a very large set of industrial railway requirements, such as vehicle composition, maintenance constraints, infrastructure capacity, and regularity aspects, have to be considered in an integrated model. General hypergraphs provide the modeling power to tackle those requirements. Furthermore, integer programming approaches are able to produce high quality solutions for the deterministic problem.

When stochastic time delays are considered, the mathematical programming problem is much more complex and presents additional challenges. Thus, we started with a basic variant of the deterministic case, i.e., we are only considering hypergraphs representing vehicle composition and regularity. We transfered solution approaches for robust optimization from the airline industry to the setting of railways and attained a reasonable measure of robustness. Finally, we present and discuss different methods to optimize this robustness measure.

Contents

1 Introduction

In rotation planning, a railway company is given a train timetable which consists of fixed planned passenger trips. The problem in rotation planning is to find feasible vehicle rotations that cover all trips with respect to the minimization of the total operation costs. The complexity arises from the fact that trips must be covered by one or more train vehicles of different types. Combinations of these types are represented by vehicle configurations. There are highly complex requirements on which vehicle configuration is allowed to cover a trip. In particular, vehicle configurations can be split or coupled at designated locations. Such operations generate vastly different cost, consequently the main objective of minimizing the total cost is fairly difficult to compute.

A further ambition in rotation planning is to keep the rotation plan easy to operate. Regular trips, e.g., daily trips, are highly preferred to be executed in the same configuration at each recurrence. Potential complexities emerge when dealing with regularity and efficiency in an integrated model. Although, a cost optimal rotation plan might be found, in practice it may not be reliable in terms of delays. Without the consideration of reliability a set up may fail. Thus, it is of interest to create a plan that addresses reliability and cost.

Mathematical optimization is an opportunity for railway companies to manage the mentioned difficulties of rotation planning. In this context, we will present a mathematical optimization approach that computes efficient and robust train schedules with respect to operational cost and stochastic delays.

First, we will model the underlying problem via hypergraphs. Then a proposed mathematical model will be used to decide which vehicle configuration is chosen for each trip. Certain techniques will be discussed to solve the mathematical model as well as to speed up the computation time. After that, reliability (robustness respectively) is taken into account. We propose sensible probability models to account for random delays. Using this information robustness can be characterized by a statistic. Two methods of optimizing this statistic are presented.

2 The Deterministic Vehicle Rotation Problem

In the deterministic case of the Vehicle Rotation Problem (VRP), fixed passenger trips of one standard week, i.e. from Monday to Sunday, must be covered by vehicle rotations. At first, we give an introduction to the basic concepts of VRP.

2.1 Passenger trips and vehicle configurations

We are given a set of *passenger trips* $t \in T$ where each trip represents a collection of subsequent train stops on a line. Every trip is performed in a certain *vehicle configuration* that may consist of one or more *vehicles*. A vehicle represents the most basic form of a physical railway unit.

Each trip $t \in T$ is defined due to an origin-destination pair of locations $(O_t, D_t) \in L \times L$, a fixed planned starting time $s_t \geq 0$ in the standard week and a travel time $d_t \geq 0$ needed to pass the distance from O_t to D_t. At locations, vehicle configurations can be split, coupled or maintained to connect arriving with departing trips. Additionally, there is given a set of *vehicle types* $f \in F$. A *feasible configuration* $c \in C_t$ is a multiset over F and consist of a selection of vehicle types. For example, let $c = \{1, 3, 3\}$ with $F = \{1, 2, 3\}$ describe a configuration of three vehicles of types 1 and 3 respectively. Each trip $t \in T$ must be executed by exactly one feasible configuration $c \in C_t$.

2.2 Hypergraph model

In our modeling approach, vehicle configurations and connections between trips are given by a hypergraph. For that purpose, let $G = (V, A)$ be a hypergraph with node set V and directed hyperarcs given by the set $A \subseteq 2^V \times 2^V$, where 2^V denotes the power set of V. Each node $v \in V$ models either the arrival or the departure of a vehicle type f in a configuration c for some trip $t \in T$, i.e., $f \in c \in C_t$. Let $t \in T$ be some trip in the following. For all $c \in C_t$ let $V_c^+ \subseteq V$ denote the set of arrival nodes to all $f \in c$ and equally $V_c^- \subseteq V$ be its set of departure nodes. In the hypergraph, each configuration $c \in C_t$ is modeled as directed hyperarc $(V_c^+, V_c^-) \in A$ that connects the arrival and the departure nodes of c. Hence, a trip $t \in T$ is represented by all its configuration hyperarcs (V_c^+, V_c^-) belonging to $c \in C_t$. Feasible connections

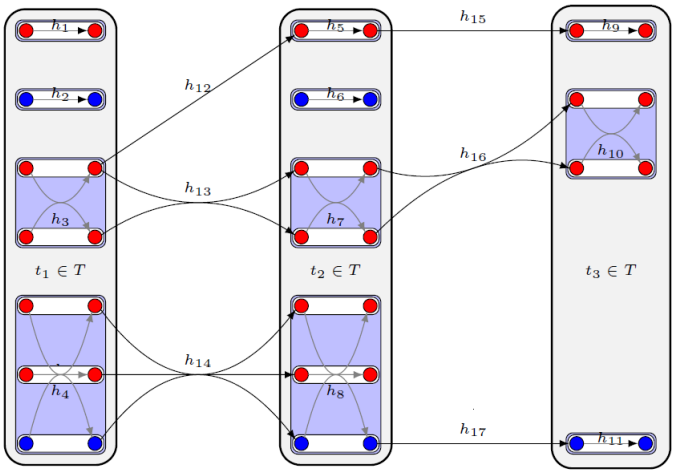

Figure 1: Hypergraph representation of trips, configurations and connections

between two configurations $c_1 \in C_{t_1}$ and $c_2 \in C_{t_2}$ corresponding to trips $t_1, t_2 \in T, t_1 \neq t_2$ are modeled as hyperarcs $(V^-_{c_1}, V^+_{c_2}) \in A$ that connect the departure and arrival nodes of c_1 and c_2 respectively, see Figure 1.

There are three trips t_1, t_2, and t_3 given. The corresponding configuration sets C_{t_1}, C_{t_2}, and C_{t_3} are modeled by hyperarcs $\{h_1, ..., h_4\}, \{h_5, ..., h_8\}$, and $\{h_9, ..., h_{11}\}$ respectively. Each configuration includes the nodes of arrivals and departures of included vehicle types $f \in F := \{blue, red\}$. The remaining hyperarcs $\{h_{12}, ..., h_{17}\}$ are the connecting hyperarcs between configurations of distinct trips.

The goal will be to compute a feasible rotation plan such that each trip is covered by a feasible configuration and each induced vehicle traverses a cycle through the underlying hypergraph, which defines a *vehicle rotation*.

2.3 Rotation plan

In a solution to the VRP, all vehicle types of a chosen feasible configuration to some trip must follow a cycle through the hypergraph network. Therefore, a *feasible rotation plan* is defined as the selection of exactly one configuration for each trip and the selection of connections between these configurations such that each vehicle induced by a selected configuration follows a cycle along selected connections. Formally, a feasible rotation plan corresponds to a *hyperflow circulation* (i.e. flow conservation for all $v \in V$) in the hypergraph. Such a hyperflow circulation is called *feasible*, if it also defines a feasible rotation plan.

In practice it is intended to make rotation plans highly operable and invulnerable against unexpected events. In common real-world instances there will be numerous trips that recur to several days of the standard week. A *regular rotation plan* aims to use only recurring connections between all trips. But in general there will be trips that occur only once or partially regular throughout the week. Hence, it will not be possible to implement full regularity in the rotation plan due to small irregularities in the given trips.

In the hypergraph, regularity is modeled by a further type of a hyperarc, a so called *regularity arc*. Together with the *connecting hyperarcs* and *configuration hyperarcs*, three types of hyperarcs are being considered. For any such hyperarc $a \in A$, we are given a cost coefficient $c_a \geq 0$. Basically, each of the mentioned objectives can be addressed by applying a certain cost coefficient to the corresponding hyperarc. A feasible rotation plan is called *optimal* if its associated hyperflow has minimum cost among all feasible hyperflows that represent a feasible rotation plan and vice versa. The objective of the deterministic VRP is to find an optimal rotation plan.

3 Hypergraph based optimization model for the VRP

As mentioned in the previous section, in the VRP a fixed set of trips must be covered by vehicle rotations. Since the problem can be modeled precisely with hypergraphs, it gives occasion to use a hyperflow model. Such a model has already been defined in [BRSW11], and is the one we are going to describe and adapt for our problem.

In order to define the mathematical model we simplify the hypergraph construction by extracting the set of configuration hyperarcs and denote it separately by H. Thus, the set of connecting arcs is now given by the set A. The hypergraph is defined as the tuple $G = (V, H, A)$.

The formulation of VRP is modeled by an integer program whose binary variables are $y_h \in \{0, 1\}$ and $x_a \in \{0, 1\}$ for each hypernode $h \in H$ and each hyperarc $a \in A$. These variables take on the value 1, if the corresponding hypernode or hyperarc is being used and 0 otherwise. The set of all hypernodes $h \in H$ covering the configurations of trip $t \in T$ is denoted by $H(t)$. Alternatively, $H(v)$ is the set of all hypernodes $h \in H$ containing $v \in V$. The set of all ingoing hyperarcs of $v \in V$ is defined as $\delta^{in}(v) = \{(V, W) \in A \mid v \in W\} \subseteq A$ and the set of all outgoing hyperarcs $\delta^{out}(v) = \{(V, W) \in A \mid v \in V\} \subseteq A$. The cost coefficients are denoted as c_a for each hyperarc and c_h for each hypernode.

This Hyperflow Integer Program (HFIP) states:

$$
\begin{aligned}
\min \sum_{a \in A} c_a x_a + \sum_{h \in H} c_h x_h & & & \text{(HFIP)} \\
\sum_{h \in H(t)} y_h = 1 & & \forall t \in T & \text{(covering)} \\
\sum_{a \in \delta^{in}(v)} x_a - \sum_{h \in H(v)} y_h = 0 & & \forall v \in V & \text{(in-flow)} \\
\sum_{a \in \delta^{out}(v)} x_a - \sum_{h \in H(v)} y_h = 0 & & \forall v \in V & \text{(out-flow)} \\
x_a \in \{0, 1\} & & \forall a \in A & \\
y_h \in \{0, 1\} & & \forall h \in H &
\end{aligned}
$$

The objective function minimizes the total cost, composed by the sum of the costs for connecting trips and the costs for vehicle configurations for each trip. The covering constraints assign one hypernode (i.e. one configuration) to each trip and the in- and out-flow constraints are the flow conservation constraints for each node $v \in V$.

3.1 Implementation and computational results

The first approach to solve the HFIP model consists of a direct implementation of the model using the CPLEX framework for C++. Basically, in the file *train.cc* the model is built up using the CPLEX interface. There we add the binary variables, the constraints and the objective function. As soon as CPLEX proves the optimality of an incumbent solution an output file in *.xml* format is generated.

The code allows the user to run either the Integer Program (IP) or its Linear Program (LP), i.e. its relaxation. One can also choose between two available output formats: the first to check the feasibility of the computed rotation plan composed by the list of hypernodes and hyperarcs in the optimal solution, and the other for graphical purposes showing the list of trips to be covered and the hypergraphs connecting those trips.

We tested this first approach on several instances. Table 1 gives some statistics on the number of trips $|T|$ to be covered and the hypergraph size given by the number of nodes $|V|$, hypernodes $|H|$ and hyperarcs $|A|$. In order to characterize the complexity of the instances some subsets of hypernodes and hyperarcs are considered. For the hypernodes, $|H_1|$ denotes the set of hypernodes consisting of a single arc and $|H_2|$ the ones with two arcs. In the case of hyperarcs we distinguish between simple arcs ($|A_1|$), *small* hyperarcs (between 2 and 5 links, $|A_{\geq 2, \leq 5}|$), *medium* hyperarcs (between 6 and 10 links, $|A_{\geq 6, \leq 10}|$) and *large* hyperarcs (more than 10 links, $|A_{>10}|$).

Instance	$\|T\|$	$\|V\|$	$\|H_1\|$	$\|H_2\|$	$\|H\|$	$\|A_1\|$	$\|A_{\geq 2,\leq 5}\|$	$\|A_{\geq 6,\leq 10}\|$	$\|A_{>10}\|$	$\|A\|$
HG2_2300	64	160	160	0	160	11622	908	1056	0	13586
HG2_2301	86	446	226	220	446	44530	7176	2064	0	53770
HG2_2201	177	1427	147	640	787	657393	59002	73312	1440	791447
HG2_2302	199	2402	450	1952	2402	1637548	340936	133064	10808	2122356
HG2_2303	164	2690	230	2460	2690	2353600	199498	289990	11380	2854468
HG3_2201	181	2316	536	1780	2316	1458406	258902	109362	6720	1833390
HG3_2202	209	2764	584	2180	2764	2107972	358938	164490	7780	2639180
HG3_2203	173	2590	350	2240	2590	2002218	293604	197844	12840	2506506
HG3_2204	162	2468	308	2160	2468	1816194	22144	193796	12840	2244974
HG3_2205	159	2296	356	1940	2296	1610254	255546	143140	6020	2014960
HG3_2206	157	2186	386	1800	2186	1380226	221660	117614	6780	1726280
HG3_2207	154	2364	284	2080	2364	1715130	163676	209576	11880	2100262
HG3_2208	143	2270	230	2040	2270	1652824	159106	195194	8040	2015164
HG3_2209	177	2544	404	2140	2544	1854576	218202	194752	7020	2274550

Table 1: Characteristics of the tested instances

Instances HG2_2300 and HG2_2301 can be solved very fast, since the number of trips is small and we have many simple arcs. Although the number of trips is not large at all, the number of hyperarcs can be larger than two million, making the problem much more difficult.

All our computations were performed using CPLEX 12.5.0 with up to 4 threads. Table 2 shows the obtained results, that is, the objective value and the total running time for both the linear relaxation and the integer program, as well as the gap between the objective values. The running times marked in red correspond to instances that are computationally expensive. HG3_2202 and HG3_2205 were running for more than 16 and 22 hours respectively. As the running time seems too consuming for practical purposes we will develop an approach to speed up the computation in the next section.

		LP		IP		
Instance	$\|A\|$	Obj. val.	CPU (s)	Obj. val.	CPU (s)	Gap (%) [1]
HG2_2300	1358	617392.00	0.99	617536.12	1.61	0.02
HG2_2301	53770	1279766.52	3.12	1279869.19	7.55	0.01
HG2_2201	791147	1194348.37	110.52	1208161.44	540.94	1.16
HG2_2302	2122356	2008527.50	175.11	2010281.75	2691.53	0.09
HG2_2303	2854468	1171960.28	294.59	1173850.10	29911.33	0.16
HG3_2201	1833390	1477097.32	233.49	1482425.00	3446.62	0.36
HG3_2202	2639180	1469527.28	350.41	1483895.97 [2]	$>$ 79000.00	1.00
HG3_2203	2506506	1656310.76	273.07	1656429.00	1367.69	0.01
HG3_2204	2244974	1654993.59	247.46	1655062.96	1376.05	0.00
HG3_2205	2014960	1415821.05	190.12	1419742.37 [3]	$>$ 60000.00	0.28
HG3_2206	1726280	1235791.48	1381.84	1235008.71	133.01	0.06
HG3_2207	2100262	1471733.57	1091.26	1471591.29	264.05	0.01
HG3_2208	2015164	1095815.92	159.08	1102262.62	47096.34	0.60
HG3_2209	2274550	1363237.09	219.95	1363451.00	1433.30	0.02

Table 2: Results for the tested instances

[1] Calculated as $1 - \frac{obj(IP)}{obj(LP)}$

[2] Value obtained in the specified running time

[3] Value obtained in the specified running time

4 Speeding up optimization

Since a railway company would need to optimize vehicle rotation frequently and with standard computers, we needed to speed up this optimization. Solving the linear program is significantly less expensive than the integer program, so we looked for a way to use the LP solution in a clever way to find an integer solution more quickly.

To visualize the LP solutions, we created a graphical representation where each trip is represented by a node, colored according to its start day, and every link in every hyperarc is represented as an edge between the two trips that it connects. Figure 2 shows our visualizations of the LP solution of file HG2_2201.

With this visualization, we noticed that the LP solutions tend to have clearly defined connections between trips. In fact, Figure 3 shows that the trip connections appearing in the LP and IP solutions are extremely similar.

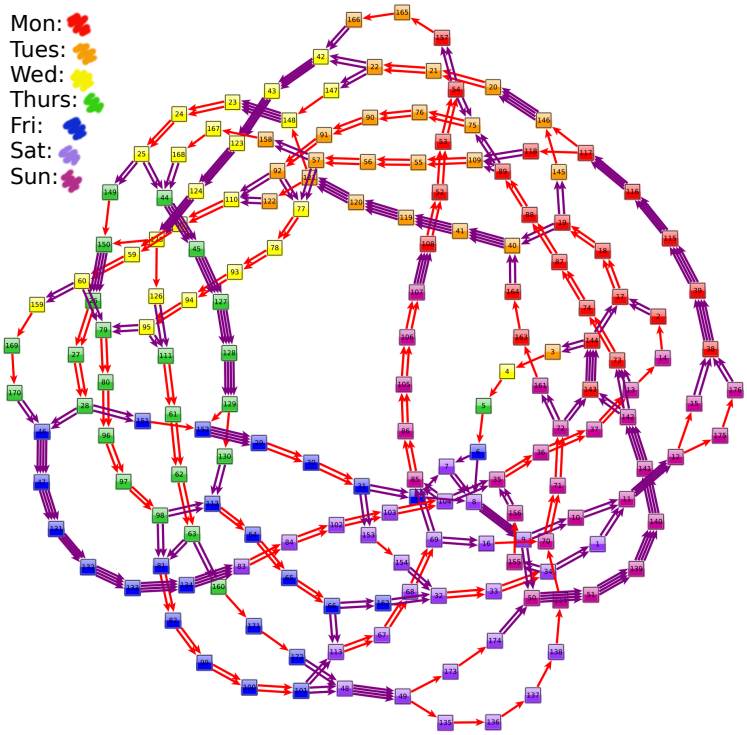

Figure 2: A graph of the LP solution for HG2_2201. The nodes represent trips and edges between them represent links in hyperarcs that connect the corresponding trips. The edges are colored by value in the LP solution, with red as one, and more blue as we approach zero.

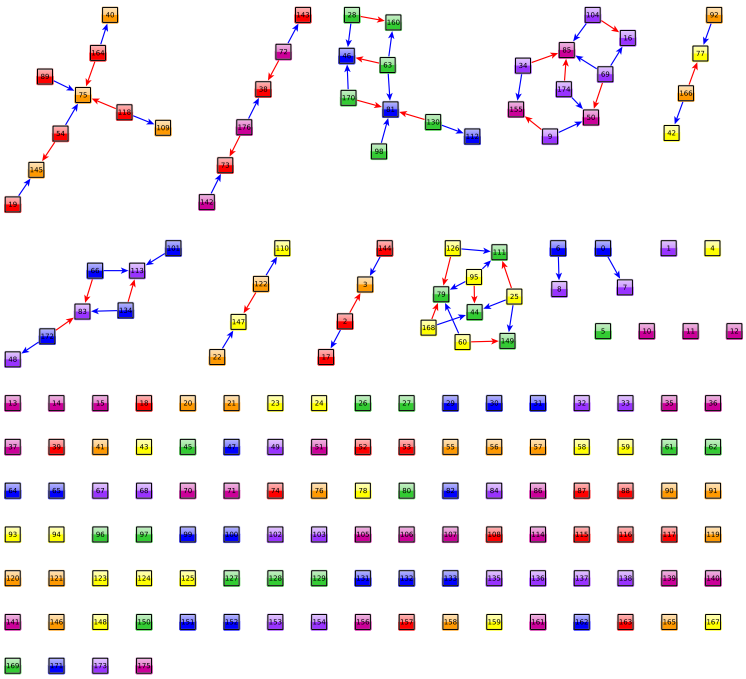

Figure 3: A graph of the symmetric difference of the LP and IP solutions for HG2_2201.xml. Blue edges are any connections between trips that are in the LP solution but not IP. Red edges are trip connections that appear in the IP solution but not LP.

This observation led us to our first strategy - to remove all hyperarc variables whose trip connections do not appear in the LP solution, and then solve the reduced IP. We can see from Table 3 that this approach led to a significant speed-up of computations The results also tend to be extremely close to optimal. However, there are still cases when we remove too many variables - that is, there are actually no feasible integer solutions with the available hyperarcs.

The more conservative strategy comes from an additional observation of Figure 3. We noticed that most of the trip connections that appear in the IP but not LP solution are connecting trips with different starting days. To account for this, we only removed arcs with the same starting day whose trip connections do not appear in the LP solution. As seen in Table 4 this approach was more consistent (none were infeasible) and gave close results. Moreover, for the two cases where the aggressive approach failed, the conservative approach sped up the direct approach. However, the speed-up was not nearly as significant, and the computation of some other instances actually slowed down, so this approach is still too computationally expensive. Since our data was for intercity trains, a large percentage of trip connections crossed the day, so the problem wasn't nearly as reduced as the first approach. Two questions therefore arise, which we do not answer in this report. First, what attributes of the data made them fail in the first approach? Second, what subset of trip connections can we remove so that the problem remains feasible, but which is still substantial enough to speed up the problem in a significant way?

		Original IP		Aggressively Reduced IP			
Instance	$\|A\|$	Obj. val.	CPU (s)	Obj. val.	CPU (s)	Gap (%) [4]	Improv (%) [5]
HG2_2300	1358	617536.12	1.61	617709.20	1.18	0.03	26.71
HG2_2301	53770	1279869.19	7.55	1279869.19	4.20	0.00	44.37
HG2_2201	791147	1208161.44	540.94	1275531.22	53.38	5.58	90.13
HG2_2302	2122356	2010281.75	2691.53	2014133.33	231.55	0.19	91.40
HG2_2303	2854468	1173850.10	29911.33	1174088.80	430.84	0.02	98.56
HG3_2201	1833390	1482425.00	3446.62	1492781.25	215.16	0.70	93.76
HG3_2202	2639180	1483895.97	> 79000.00	Infeasible	NA	NA	NA
HG3_2203	2506506	1656429.00	1367.69	1660023.97	1473.36	0.22	-7.73
HG3_2204	2244974	1655062.96	1376.05	1658546.36	370.50	0.21	73.08
HG3_2205	2014960	1419742.37	> 60000.00	Infeasible	NA	NA	NA
HG3_2206	1726280	1235791.48	1381.84	1239271.05	184.39	0.28	86.66
HG3_2207	2100262	1471733.57	1091.26	1472019.45	250.70	0.02	77.03
HG3_2208	2015164	1102262.62	47096.34	1102780.15	262.09	0.05	99.44
HG3_2209	2274550	1363451.00	1433.30	1363612.60	225.15	0.01	84.29

Table 3: Results for Aggressive Approach

		Original IP		Conservatively Reduced IP			
Instance	$\|A\|$	Obj. val.	CPU (s)	Obj. val.	CPU (s)	Gap (%) [6]	Improv (%) [7]
HG2_2300	1358	617536.12	1.61	617536.12	2.62	0.00	-62.32
HG2_2301	53770	1279869.19	7.55	1279866.66	7.87	0.00	-4.30
HG2_2201	791147	1208161.44	540.94	1208162.34	124.96	0.00	76.90
HG2_2302	2122356	2010281.75	2691.53	2010578.65	3301.68	0.02	-22.67
HG2_2303	2854468	1173850.10	29911.33	1173890.20	11422.83	0.00	61.81
HG3_2201	1833390	1482425.00	3446.62	1482427.70	2064.13	0.00	40.12
HG3_2202	2639180	1483895.97	> 79000.00	1489951.32	48870.30	0.41	NA
HG3_2203	2506506	1656429.00	1367.69	1656433.00	2340.18	0.00	-71.10
HG3_2204	2244974	1655062.96	1376.05	1655144.46	1587.40	0.00	-15.36
HG3_2205	2014960	1419742.37	> 60000.00	1421541.27	9109.58	0.13	NA
HG3_2206	1726280	1235791.48	1381.84	1235916.48	2946.04	0.01	-113.20
HG3_2207	2100262	1471733.57	1091.26	1471733.73	610.25	0.00	44.10
HG3_2208	2015164	1102262.62	47096.34	1102700.35	12026.70	0.04	74.46
HG3_2209	2274550	1363451.00	1433.30	1363451.21	1014.03	0.00	29.25

Table 4: Results for Conservative Approach

5 Delays and disruptions

In the previous sections we have minimized the cost of operating a VRP (i.e. maximized efficiency), assuming that all trains run on time. In practice this is rarely the case. Trips are regularly delayed for a variety of reasons, and delays have a cost associated with them. This cost is harder to quantify than the operating cost. For example, if trains are regularly delayed this will reduce demand for railway transportation and complicate the logistics of operating the network. Significantly delayed trains will also directly reduce profits as passengers on such trains are eligible for refunds. Therefore it is important to minimize delays as well as the operating cost.

[4] Calculated as the objective function value of the Aggressively Reduced IP devided by the Original IP objective function value.

[5] Calculated as $1 - (\frac{\text{Aggresively Reduced IP CPU time}}{\text{Original IP CPU time}})$

[6] Calculated as the objective function value of the Conservatively Reduced IP devided by the non Original IP objective function value.

[7] Calculated as $1 - (\frac{\text{Conservatively Reduced IP CPU time}}{\text{Original IP CPU time}})$

To begin we will introduce an important classification of delay types. *Primary delays* are delays that are not caused by other trains on the network. Types of primary delay include:

- Breakdown delays (e.g. due to an engine failure)
- Track delays (e.g. due to a tree falling on the track)

We cannot do anything about this kind of delay in our optimization. *Secondary delays* are caused by other trains on the network due to the knock on effects of primary delays. Types of secondary delay include:

- Departure delays (there may not be sufficient buffer time between two trips to soak up a primary delay in the first trip)
- Safety delays (if a train is delayed then trains behind it on the same track will have to wait)
- Station delays (stations have maximum capacities so trains may have to wait to use them)

We can reduce secondary delays, for example by finding VRPs which do not propagate delays as much. We will describe such VRPs as *robust* and say that our goal is to improve *robustness* of VRPs. This is not standard terminology. In the context of reducing delays, one could investigate:

- Worst case delays - This is often called robust optimization
- Average case delays - This is often called stochastic optimization

We will investigate average case delays i.e. do stochastic optimization. As a result we will consider probability distributions of delays. Note that more robust VRPs will likely have higher operating costs, so it will be important to find a balance between robustness and efficiency.

A review on different approaches to computing delays in railway networks can be found in [MM07]. We will use ideas from [BDNS10], many of which are expanded upon in [Dov14]. This work is for the tail assignment problem of airplanes. See also the literature [CCG+08] and [CCG+12], which is on robust railway rotation planning.

The structure of the rest of this section will be as follows. We will first introduce a model for delay propagation. Next we will discuss how to use this model to compute delay probabilities. We will then devise a statistic that characterizes the robustness of a VRP. This and visualizations of delays will enable us to understand why some VRPs are more robust than others. We can use this knowledge to design methods for finding VRPs with good trade offs between robustness and efficiency.

5.1 Model of delay propagation

The basic mechanism of delay propagation is: A train experiences a primary delay that is larger than the *minimum turn over time* between trips (the time it takes to clean the train, swap crews etc.), so it departs late on its next trip. We now formulate this more precisely. For a trip $t \in T$:

- AD_t denotes the arrival delay
- C_t denotes the set of connecting trips (which are the trips containing vehicles needed for trip t)
- $b_{s,t}$ for $s \in C_t$ denotes the stop over time (the timetabled arrival time of s minus the timetabled departure time of t)
- D_t denotes the delay experienced during the trip (both primary and secondary delays)
- c denotes the minimum turnover time

These are all real numbers. Using this notation, delay propagation can be captured with the following model. Initially suppose $\mathrm{AD}_t = 0$ for all $t \in T$. Then iterating through the trips in order of increasing start time, set

$$\mathrm{AD}_t = \underbrace{\max_{s \in C_t} (\overbrace{\max(\mathrm{AD}_s + c - b_{s,t}, 0)}^{\text{Connecting delay}})}_{\text{Departure delay } (\mathrm{DD}_t)} + D_t.$$

This model is well defined since iterating over the trips in this order ensures that AD_s for all $s \in C_t$ are set before AD_t is computed. So given the delay D_t for each trip t we know the departure and arrival delay for all trips.

This is a very simple model that omits some important details. In order for the model to be realistic we also include:

- Different types of primary delay e.g.
$$D_t := B_t + T_t,$$
where B_t is the breakdown delay and T_t is the track delay. These should depend on the length of the trip
- Minimum safety distances
- Times for splitting and joining vehicles

These are outlined in more detail in our problem brief.

5.2 Computing delays probabilities

The model of the previous section explains how delays propagate through the network given the primary delays for each trip. However the delays are caused by disruptions, which are random events, and therefore delays are random events. It is important to know the probability distribution of delays for each trip. In this section we consider two approaches for computing this. They both have different advantages and disadvantages.

5.2.1 Simulation

By *disruptions* we refer to the underlying cause of primary delays. Depending on the VRP the disruptions may affect trains in the network in different ways and result in different trip delays. We suppose track delays are caused by *track disruptions*. They happen at specified times and locations, and last for given durations. Similarly we suppose train delays are caused by train disruptions. For each train these occur after the train has travelled predefined distances and they last for specified durations.

Given a set of disruptions we can use the model from Section 5.1 to compute the departure and arrival delays of all trips. So we can randomly generate sets of disruptions (based on assumptions on their frequency, locations etc.) and compute these delays. By doing this many thousands of times we can construct an approximate probability distribution for the departure and arrival delay for each trip.

The advantages of the simulation approach are:

- It is easier to make assumptions about disruptions than delays e.g. using intuition, expert opinion, or historic data on disruptions. The delays themselves are affected by how the network is managed and so are hard to make assumptions on.
- It allows the use of a more realistic model that includes minimum safety distances, station capacities etc. The features are hard to implement in the propagation of probabilities approach we consider in Section 5.2.2.

The main disadvantage is that simulation can be computationally expensive; we may need to run many thousands of simulations with different randomly generated input disruptions in order to accurately approximate the true departure and arrival delay probability distributions.

The assumptions we use to randomly generate the disruptions are the following:

- For each train breakdown disruptions occur at exponentially distributed time intervals, which is a reasonable model for component failure. We want the support of the distribution of the durations to be $\mathbb{R}^{\geq 0}$, so we assume an exponential distribution for this too.
- Track disruptions occur at exponentially distributed time intervals and uniformly in space across the railway network (i.e. are more likely to occur on longer sections of track). For the duration of the disruption we again take an exponential distribution.

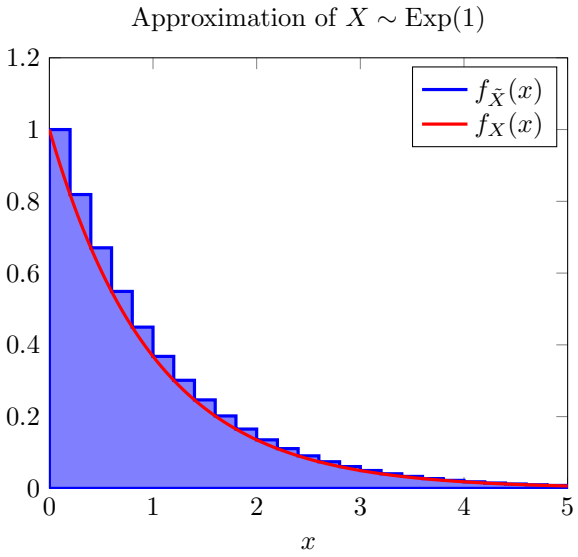

Figure 4: A pdf $f_{\tilde{X}}$ of a discretized random variable that approximates the pdf f_X of an Exponential random variable.

We do not currently have values for the parameters in these distributions based on real data. We just used intuition (gained from using railways ourselves) to choose reasonable values. We observe that different parameter values change various measures of delay (see Section 5.3) in absolute but not relative terms. So the computational results later in this report should still be valid even if our parameter values turn out not to be realistic.

5.2.2 Propagation of probabilities

For this approach we suppose that DD_t, AD_t and D_t are random variables. Initially that the AD_t are the random variable taking the value 0 with probability one. Then given delay random variables D_t, the model in Section 5.1 iteratively defines new random variables. This is well defined mathematically, as the operations of summation and taking the maximum are defined for random variables, however it cannot be done analytically. There is no realistic choice of distribution for D_t such that the AD_t remain in the same finite dimensional class of distributions. Therefore, in order to compute these distributions an approximation is necessary. Note that with this approach we are not able to include minimum safety distances in our model. It produces the same output as the simulation approach; delay distributions for all trips.

A number of approaches to propagating random variables are considered in the literature, such as using phase-type distributions and theta-exponential polynomials (see the review in [MM07]), but we chose the approach of discretized random variables (see [BDNS10] and [Dov14]). This approach has been demonstrated to be accurate and computationally efficient, and it is conceptually simple. It also allows the choice of arbitrary delay distributions.

We refer the reader to the cited papers for the precise details as our implementation is the same. But the approach is roughly speaking the following. Random variables are approximated by what we call *discretized random variables*. They are random variables which have probability density functions (pdfs) that are step function with uniform sized steps. This can be seen in Figure 4.

The summation of random variables is computed by taking the convolution of pdfs. Therefore summation for discretized random variables is simple to compute as it corresponds to the convolution of step functions i.e. just the summation and multiplication of step values. Similarly the max operation for discretized random variables leads to a straightforward computation.

An advantage of propagation of probabilities is that it is computationally cheaper than simulation (using discretized random variables). In [Dov14] this allowed a random variable representing delays to be optimised directly. The disadvantage of propagation of probabilities is that it requires a simplified model; features such as minimum safety distances and station capacities are hard to include.

Where as with the simulation approach we were able to make intuitively reasonable assumptions on disruptions, it is harder to make these on delays. The literature [MM07] suggests that the cumulative density function of delay d given by

$$\mathbb{P}(D_t \leq d) = \begin{cases} 0 & d < 0 \\ 1 - \theta\lambda \exp(\lambda d) & d \geq 0, \end{cases}$$

has been observed experimentally and justified theoretically. Here θ is the probability of delay and the delay has expected duration λ when there is one. We suppose

$$D_t := B_t + T_t + S_t$$

where B_t, T_t and S_t are random variables for breakdown, track and safety delays that have this form. The parameter values in these distributions are chosen heuristically depending on the length of the track and how many trips use the track. Alternatively we can use the simulation approach with assumptions on the disruptions to suggest or even calibrate assumptions on delays.

We have implemented both the simulation and propagation approach for computing delay distributions. Given equivalent assumptions, both implementations calculate delay distributions to within an arbitrarily small discrepancy. This gives strong evidence that both approaches have been implemented correctly.

5.3 Characterizing robustness

We have two methods of computing departure and arrival delay distributions of all trips. We want to combine all this information into a single random variable or number which allows us to easily decide whether one VRP is more robust than another. Some possible statistics are the following:

- $S_1 := \frac{1}{|T|} \sum_{t \in T} \mathbb{E}[\mathrm{DD}_t]$ - Average expected departure delay
- $S_2 := \frac{1}{|T|} \sum_{t \in T} \mathbb{P}[\mathrm{DD}_t > 0]$ - Average propagated delay (used in Robust Tail Optimization paper)
- $S_3 := \frac{1}{|T|} \sum_{t \in T} \mathbb{P}[\mathrm{AD}_t > 5]$ - Average probability that arrival delay is greater than 5 minutes

The statistics S_1 and S_2 ignore the primary delays (that we have no control over) and just measure the secondary delays. The statistic S_3 includes primary delays but tells us something that is more meaningful to a train company or passenger. In particular we could just look at the expectation of these statistics.

We see in Figure 5 that $\mathbb{E}[S_1]$, $\mathbb{E}[S_2]$ and $\mathbb{E}[S_3]$ all contain the similar information about robustness; whichever statistic we compute all of them will likely agree which VRP is more robust. Therefore we favour using the statistic S_1 as it only measures secondary delays and it is easy to interpret.

5.4 Visualization

We have characterized robustness, enabling us to decide whether one VRP is more robust than another by comparing $\mathbb{E}[S_1]$. We have created a visualisation that helps us better understand why this is. An example is shown in Figure 6. This understanding has motivated our approaches to optimising robustness in the next section.

Our computational results show that delays are predominantly the result of short buffer times, which lead to departure delays. Safety delays are not a big source of delay. From the visualisations such as Figure 6 we also notice that multiple consecutive short buffer times are a common in the VRPs that minimise operating cost. This allows delays to build up to high levels.

6 Optimizing for robustness

The most natural way to improve robustness would be to minimize our measure of robustness $\mathbb{E}[S_1]$ directly. This may be possible using the ideas from [Dov14], however this is hard. Similarly, our observations in the previous section suggest that consecutive short buffer times are the main problem,

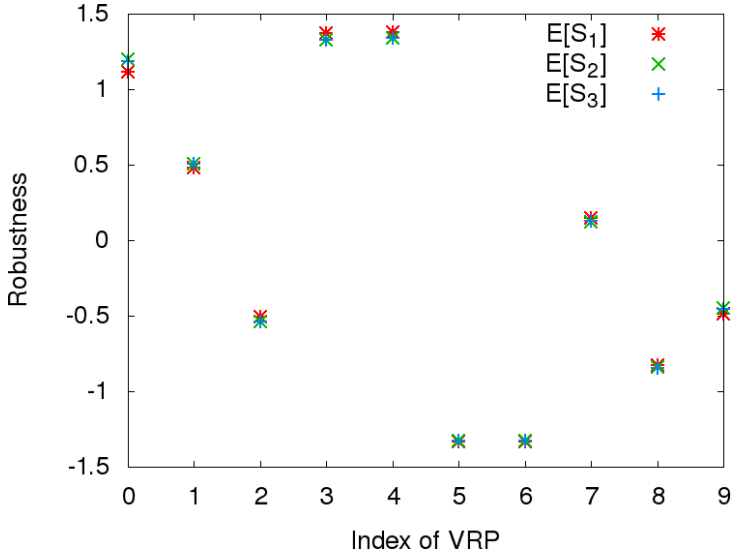

Figure 5: Let $S_{i,n}$ denote the statistic S_i for one of ten feasible VRPs which are indexed by n. Set $V_i = \{\mathbb{E}[S_{i,n}]\}_{n=0}^{9}$. For each of $i = 1, 2, 3$ we have plotted $(\mathbb{E}[S_{i,n}] - \text{mean}(V_i))/\text{s.d}(V_i)$ for $n = 0, ..., 9$. So we have plotted shifted and scaled versions of $\mathbb{E}[S_i]$ for different VRPs. We see that the data points for a given VRP are all very close together, suggesting that all the statistics contain the same information.

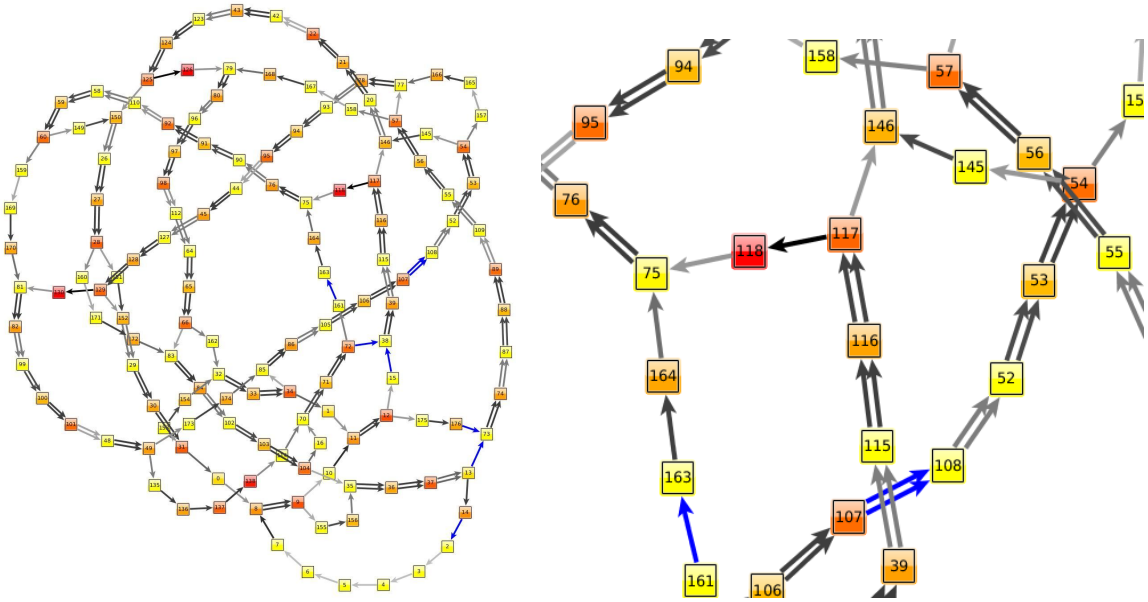

Figure 6: Part of a visualization of the delay in a rotation plan. It was created in the graph visualization program yEd. The nodes are trips, which are colored from yellow to red as the expected departure delay $\mathbb{E}[S_1]$ increases (mousing over them in yEd shows the numerical value). The edges show connecting and splitting of vehicles with darker edges indicating shorter buffer times (mousing over them in yEd shows the numerical value). The blue edges mark the end point in the week i.e. where we stop propagating delays.

but reducing these is still quite technical. Instead, we begin with the simplest possible approaches to optimizing for robustness, however they turn out to be effective.

In particular we will investigate a local search around the VRP with minimal operating cost and a heuristic optimization approach which penalises short buffer times.

7 Project Outcomes

We finish with a summary of the outcomes of our project. We have achieved:

- A better understanding of the VRP problem and its complexities
- Code for:
  - Optimizing cost - speed up using heuristics
  - Computing delay probabilities. Two methods - simulator and propagator
  - Optimizing robustness - Local search approach and optimization using a heuristic
- Visualizations to help develop intuition for:
  - IP and LP solutions
  - Delays

References

[BDNS10] R. Borndörfer, I. Dovica, I. Nowak, and T. Schickinger. Robust Tail Assignment. ZIB Report 10-08, ZIB, 2010.

[BRSW11] Ralf Borndörfer, Markus Reuther, Thomas Schlechte, and Steffen Weider. A Hypergraph Model for Railway Vehicle Rotation Planning. In Alberto Caprara and Spyros Kontogiannis, editors, *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 20 of *OpenAccess Series in Informatics (OASIcs)*, pages 146–155, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[CCG+08] V. Cacchiani, A. Caprara, L. Galli, L. Kroon, and Gábor Maróti. Recoverable Robustness for Railway Rolling Stock Planning. In *OASIcs-OpenAccess Series in Informatics*, volume 9. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2008.

[CCG+12] V. Cacchiani, A. Caprara, L. Galli, L. Kroon, G. Maróti, and P. Toth. Railway Rolling Stock Planning: Robustness Against Large Disruptions. *Transportation Science*, 46(2):217–232, 2012.

[Dov14] I. Dovica. *Robust Tail Assignment*. PhD thesis, Technical University of Berlin, 2014.

[MM07] L. E. Meester and S. Muns. Stochastic Delay Propagation in Railway Networks and Phase-type Distributions. *Transportation Research Part B: Methodological*, 41(2):218–230, 2007.