Optimization of a Railway Hub-and-Spoke System

Routing and Shunting

Optimization of a Railway Hub-and-Spoke System: Routing and Shunting

Technical Report 477, Department of Computer Science, ETH Zurich

Michael Gatto[*] Riko Jacob[*] Marc Nunkesser[*]

Abstract

We consider the hub-and-spoke setting of the railway product "Cargo Express" of SBB Cargo, where freight cars are hauled over night from many sources to many destinations. To this aim, trains collect the cars from the source stations, along a route to the hub. The hub is a shunting yard with a hump, where the cars are rearranged and routed to their delivery stations in new trains.

We provide some basic theoretical analysis of the problem, and report first experiments with real-world data.

More precisely, we consider the routing and shunting component of the problem separately. We reduce the routing problem to a distance constrained vehicle routing problem, and derive efficient separation heuristics for a branch and cut approach. We show that the problem of sequencing the incoming and outgoing trains with the goal of minimizing the hub's capacity is NP-hard. Finally, we give a dynamic program for minimizing the makespan of shunting operations given a predefined sequence of incoming trains.

1 Introduction

Railway systems pose a multitude of interesting optimization problems. Apart from passenger railway systems also freight railway systems lend themselves well to combinatorial optimization approaches. In this paper we address interconnected algorithmic problems related to fast overnight freight transportation in a freight railway hub-and-spoke system.

Our running example is the Swiss "Cargo Express" service [19], but our results apply also to any other freight railway system that is designed similarly. Favored by the Swiss ban on night-time truck driving, the overnight transportation of mainly perishable products by freight trains has become a successful and widely used service offered by Swiss federal railways SBB Cargo Ltd. Almost 50 Swiss stations are connected to the Cargo Express railway net. At these stations, customers can deposit goods in the evening or pick up goods in the morning, provided they have signed up for this service in advance. The network is designed as a hub-and-spoke system, the hub being the central shunting yard in Däniken and the spokes the other stations. Currently, a further shunting yard (Olten) close to Däniken is used because of capacity reasons.

This short description already reveals major optimization tasks in such a system:

- Minimize the number of trains necessary to fulfill all demands in one night and also the total distance traveled.
- Minimize the shunting capacity needed at the hub.

In our model we decompose the problem into two main parts: The *routing* of trains to and from the hub and the *shunting* at the hub. We present each model together with its results in a separate section, beginning with the routing. In the shunting part we explain how the two parts are connected. The paper is structured as follows: In the next section we give a detailed description of the train routing problem. Next, we give

[*]Department of Computer Science, ETH Zürich, {gatto,rjacob,nunkesser}@inf.ethz.ch

a mathematical programming model for it and present a branch and cut solution approach. Then, we show how the shunting problem is related to a minimum cut linear arrangement problem and also consider a mathematical programming formulation. Finally, we analyze good groupings of shunting operations and present experimental results for our routing approach.

1.1 Problem Description

In this section, we describe the mode of operation of Cargo Express. The mathematical formulation of the optimization tasks we consider are described in later sections. First of all, the timetable and the served stations are fixed for longer time periods and reflect fixed customer demands. This means that customers might buy a daily transport of 50 tons of goods from A to B during six months, for example[1]. The schedules are repeated daily with the exception of Sundays where a different schedule is used.

After the goods have been placed at a station they need to be transported to the central shunting yard, the *hub*. In a first phase shunting engines transport the goods at small stations to nearby larger stations. Mostly, it is clear what the nearby larger station of a small station is. As the routing of these shunting engines is predetermined, we consider the problem after this phase, i.e., when all goods have been placed at larger stations. At some of these larger stations freight trains collect the deposited goods and start their trip to the hub. On their trip they may stop at other stations and collect the cars of these stations. More precisely, freight cars that wait at these stations are coupled to the freight trains. The process of coupling these cars takes a non negligible time[2] hence referred to as *couple time*. The length of a freight train is limited.

At the hub the freight trains are shunted. This means roughly that the incoming trains are decomposed into the cars and outgoing trains are composed from cars of incoming trains. In particular an outgoing train can only depart after all its cars have arrived at the shunting yard. A shunting yard has a limited capacity: It can only accommodate a limited number of cars respectively trains and only a limited number of shunting operations can be performed in a given time period.

1.2 Related Work

There is an enormous wealth of publications on the related *vehicle routing problem*. The vehicle routing problem itself has been studied in many variants, the closest in spirit to our problem being the distance constrained vehicle routing problem, which has received comparatively little attention. Most of the publications of exact algorithms date back to the 80ies [4, 11, 12]. There are several implementations for the general vehicle routing problem, commercial as well as free ones, see [9, 17] for surveys. One of the few free and open ones is the code by Ralphs et al. [18], on which we base our implementation. Ralphs' implementation is itself based on his SYMPHONY branch and cut framework [17]. Another branch and cut implementation for the vehicle routing problem is by Blasum and Hochstättler [3]. More on the vehicle routing problem can be found in the book edited by Toth and Vigo [20].

There are also some publications on shunting, for example [2]. Here it is necessary to check which type of shunting yard was modeled. In [2] the authors consider shunting of trams which is different from shunting freight trains. It seems that only a few contributions consider a shunting yard with a hump [6, 7]. We are not aware of any publication, in which both routing and shunting are addressed together.

2 Routing

In this section we first present our mathematical model of the routing problem, then present our solution approach.

[1]As it turns out SBB Cargo is much more flexible in accommodating for changing/new demands, but we describe the general concept of the Cargo Express product.

[2]The reason for this is that after coupling the cars, the brakes of every newly coupled car have to be tested.

2.1 Model

In the routing problem we search for short routes from the stations to the hub. We treat the transport from the stations to the hub and the transport from the hub to the stations separately. Obviously, these two problems are symmetric, therefore it is sufficient to analyse only one direction. Here we consider the transport from the stations to the hub. This model implies that we do not use the detailed information about point to point transports that would be given by a $(n \times n)$ supply and demand matrix but rather the row and column totals (depending on the direction) in this matrix. This is expressed by a supply (demand) value that is associated with each node. This value is the total amount of goods (in freight cars) that is to be transported from the station to the hub, or vice versa depending on the setting. The following definition gives a formalization of the problem.

Definition 2.1 (Train Routing Problem with fixed train fleet (TRP)). We are given a train network $G = (V, E)$ with a specified hub node $H \in V$, a length function $c : E \to \mathbb{R}^+$ and supplies $s(v)$ for each node $v \in V \setminus \{H\}$, a couple time t_{couple}, an average speed $\bar{v}$, a maximum train length C_{TRP}, a maximum traveled distance D_{TRP} and the number of available freight trains K. A feasible solution S consists of K *routes* $R = \{r_1, \ldots, r_K\}$, i.e. paths in the network having one endpoint in H and an association of each node $v \in V \setminus \{H\}$ in the network with a route $r = \rho(v)$ such that the following properties hold:

1. No route is longer than D_{TRP}. The *length of a route* r is defined as the length of the path plus $|\{v \in V | \rho(v) = r\}| \cdot t_{\text{couple}} \cdot \bar{v}$.
2. No train is longer than C_{TRP}. The *length of a train* for a route r is $\sum_{v:\rho(v)=r} s(v)$.
3. All nodes are associated with some route.

The cost of a solution is the sum of the lengths of the routes. The train routing problem is to find a minimum cost solution.

The restriction to a fixed train fleet is not very limiting. In practice, one wants to minimize a weighted sum of the number of used trains and the traveled distance. Reasonable values for the number of used trains are usually in a very small interval so that the optimization can be done for all of these values.

2.2 Branch and Cut Approach

In general there are many different possibilities to tackle the train routing problem, even after one has decided to use an exact approach. Out of these possibilities we could only evaluate a small subset. Natural candidates are those based on formulations that proved successful for the vehicle routing problem, i.e. *vehicle flow* models with the classical two- or three index formulations, *commodity flow* models or *set partition* models, see [20] for an excellent overview. We only tried the two and the three index formulation, out of which the two index formulation proved more successful and will be presented in the following. For convenience, we restate the distance-constrained capacitated vehicle routing problem.

Definition 2.2 (Distance Constrained Capacitated Vehicle Routing Problem (DCVRP)). Given a network $G = (V, E)$, a cost function $c : E \to \mathbb{R}$ find K circuits with minimum total cost, such that

1. Each circuit visits the depot vertex
2. Each customer vertex is visited by exactly one circuit
3. The sum of the demands on each circuit does not exceed the allowed capacity C_{DCVRP}.
4. The sum of the edge lengths on each circuit does not exceed the distance constraint D_{DCVRP}.

2.2.1 DCVRP ILP-Formulation

The two index formulation of the DCVRP uses Boolean variables x_e to indicate if a given edge $e \in E$ is chosen. We give it for undirected graphs, and explain it below.

$$\textbf{DCVRP:} \quad \min \sum_{e \in E} c_e x_e$$

$$\text{s.t.} \sum_{e=\{i,j\} \in E} x_e = 2 \qquad \forall i \in V \setminus \{H\} \tag{2.1a}$$

$$\sum_{e=\{H,j\} \in E} x_e = 2K \tag{2.1b}$$

$$\sum_{e=\{i,j\} \in E, i \in S, j \notin S} x_e \geq 2r(S) \quad \forall S \subset V \setminus \{H\}, S \neq \emptyset \tag{2.1c}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{2.1d}$$

Equations (2.1a) enforce that each node except for the hub has degree two, (2.1b) enforces that the hub has degree $2K$. Inequalities (2.1c), the *capacity cut constraints*, enforce that the graph is connected and that the capacity and length constraints are met. Note that every vehicle contributes two to the number of edges of the cut. Here, $r(S)$ is the maximum of two values $c(S)$ and $d(S)$: $c(S)$ is the minimum number of vehicles that have enough capacity to serve the set S. This value can also be interpreted as the solution of a bin packing problem with binsize C_{DCVRP}. The value $d(S)$ is the minimum value $k \in \mathbb{N}$ such that the objective value v^k_{TSP} of a k-TSP problem on S divided by D_{DCVRP} and rounded up equals k:

$$d(S) = \min \left\{ k \in \mathbb{N} \middle| k = \left\lceil \frac{k\text{-TSP}(S)}{D_{\text{DCVRP}}} \right\rceil \right\} . \tag{2.2}$$

2.2.2 Adapting the Model

Although not identical, the train routing problem defined above has many similarities with DCVRP. In the following, we give a transformation[3] such that the optimal solution of any TRP instance I_{TRP} can be derived from the optimal solution of the corresponding transformed DCVRP instance $\Psi(I_{\text{TRP}})$. This approach allows us to directly use existing software packages.

Given an instance I_{TRP} of TRP with network G_{TRP} the transformation Ψ applies the following three types of modifications to it: First it adds all missing edges to G_{TRP}. The length of such a new edge $e = (u, v)$ is set to the length of the shortest u, v path in G_{TRP}. In the next step it adds $t_{\text{couple}} \cdot \bar{v}$ to the weight of each edge of the network. Finally, it puts a gadget on top of the network. Figure 2.1 shows how the TRP-instance after the first two modifications, represented by the circular nodes is transformed into a DCVRP instance by adding extra nodes and edges. The extra nodes are shown as rectangular nodes. The underlying idea is to allow each vehicle to "jump" from the depot to a start point in the network. To that purpose, we add K extra nodes $\{v^e_1, \ldots, v^e_K\}$ to the network. These nodes are all connected to the depot H with edges of length $-M$, with $M \gg d(e)\ \forall e \in E$. The extra nodes are connected to the rest of the network via the complete bipartite graph. The length of each such edge is zero. The extra nodes are not interconnected. Each extra node has an associated demand of M', with $M' > \sum_{v \in V} s(v)\ \forall v \in V$. We set the capacity C_{DCVRP} of the vehicles in the DCVRP instance to $C_{\text{DCVRP}} = M' + C_{\text{TRP}}$, finally we set $D_{\text{DCVRP}} = -M + D_{\text{TRP}}$. The correctness of this transformation is established in the following lemma.

Lemma 2.3. *Given an optimal solution S_{DCVRP} of cost c to DCVRP on $\Psi(I_{TRP})$ for a TRP instance I_{TRP}, the cost of an optimal solution S_{TRP} to I_{TRP} is $c + K \cdot M$ and S_{TRP} can be reconstructed from S_{DCVRP} in linear time.*

[3]Note that the common transformation $c_{\{i,j\}} \longleftarrow c_{\{i,j\}} - c_{\{H,i\}} - c_{\{H,j\}}$ by Clarke and Wright savings [5] only works in the other direction, in the sense that it transforms a problem with cycles into a problem with paths.

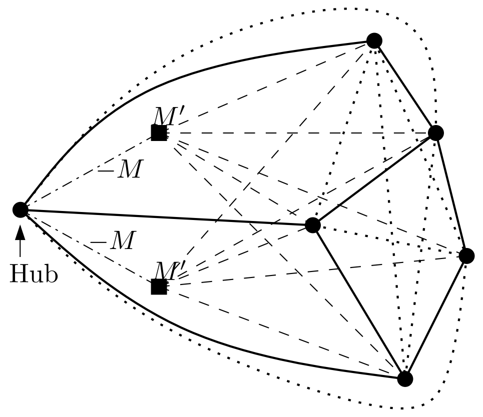


Figure 2.1: Transformation from TRP to DCVRP for two trains. The new square nodes have demand M. The original graph consists of all circular nodes together with the solid edges. The dashed edges have length zero. The dotted edges are introduced in an earlier phase of the transformation.

Proof. We first show that S_{TRP} has the following form: It consists of K cycles, such that the ith cycle can be written as $(H, v_i^e, v_{i,j_1}, \ldots, v_{i,j_{l_i}}, H)$. The reason for this is that the supply of M of each extra node together with the capacity constraints enforce that exactly one extra node is on each circuit. The negative lengths of the edges (H, v_i^e) enforce that the extra nodes must be directly after (or before) H on the circuits. To construct S_{TRP} we simply set the ith route to $(v_{i,j_1}, \ldots, v_{i,j_{l_i}}, H)$. From the description of the transformation it follows that the feasibility of S_{DCVRP} guarantees that S_{TRP} is feasible. Note that we assume a couple time also at the stations where trains start their journey. As for the optimality, assume S_{TRP} is not an optimal solution. Then let S' be the optimal solution to the TRP consisting of K routes. This solution can be transformed into K cycles by reverting the above construction. It is straight-forward to check that these cycles form a cheaper feasible solution for the DCVRP instance. □

Note that the bipartite component of the gadget can be slimmed down: All we need is that there is a perfect matching between each subset $S \subset V$ of size K and the extra nodes. Thus, it is enough to insert, for K nodes in N, only the edges $(v_i, v_i^e), i \in \{1, \ldots, K\}$ in the bipartite component.

2.2.3 Separation Heuristics

The core part of every branch and cut algorithm is the design of a separation algorithm that effectively separates a given fractional point from the convex hull of integer solutions. The general separation problem is NP-complete and this still holds for most known classes of valid inequalities including the ones of type (2.1c), see [1]. For this reason we focus on effective *separation heuristics* that try to find violated inequalities of type (2.1c). These inequalities comprise two subtypes, capacity and distance constraints, depending on which of the values $d(S)$ and $c(S)$ is larger.

We have based our implementation on the branch and cut code by Ralphs et al. [18] for the capacitated vehicle routing problem. All separation heuristics dealing with capacity constraints are described in [18]. Therefore, we focus on our separation heuristics for the new distance constraints.

Let $\hat{x}$ be a fractional solution, and denote by $\hat{G} = (V \setminus \{H\}, \hat{E}), \hat{E} = \{e : \hat{x}_e > 0\}$ the *support graph* corresponding to $\hat{x}$. The general idea of the inserted cuts is as follows: If the length ℓ of the route required to serve a connected component $S \in \hat{G}$ exceeds D_{TRP} the component must be served by more vehicles. In this case we introduce a constraint of type (2.1c). This proceeding remains valid if ℓ is a lower bound on the length of the route.

Given a fractional solution $\hat{x}$ and a connected component $S \in \hat{G}$, the length of the fractional route is computed as the sum over all weights of edges in S, weighted by $\hat{x}$, excluding edges of weight $-M$. If this length exceeds D_{TRP}, we introduce a valid inequality of type (2.1c) for S. Let c be the value of the (graph theoretic) $(S, \{H\})$-cut. Then, setting $r(S)$ to $\lfloor \frac{c}{2} \rfloor + 1$ gives a locally valid cut. Next, we apply the shrinking heuristic described in [18] to enforce stronger cuts if the previous search was unsuccessful. All these cuts have only local validity in the branch-and-cut search tree, because the branching decisions enforce or forbid some edges.

Given an integral solution, the length ℓ of a route is computed in the obvious way by summing up the length of the edges in the route, but neglecting edges of weight $-M$. If the route length exceeds D_{TRP}, we introduce a cut (2.1c) with right-hand-side $2\kappa(S)$. The value $\kappa(S)$ is a global lower bound on the number of vehicles needed to serve S, and is explained later in the section. Similarly to [18], we try to enforce stronger cuts by considering only parts of each route: we sequentially add edges along a route until the distance-constraint is violated, and enforce a cut on this smaller subset of vertices. Next, we shorten the route from its source, and add a cut for each subset violating the distance constraint. This procedure is iterated by adding one edge to the previous prefix of the route. As before, these cuts use $\kappa(S)$ as lower bound on the number of vehicles.

The value $\kappa(S)$ is computed by adapting two standard relaxations of the TSP to our needs, the relaxation to 1-trees and the one to the assignment problem, see [13] for details. Given an integral connected component S, we compute the minimum weight spanning tree on S. If the cost of the spanning tree exceeds D_{TRP}, we introduce a cut. In order to find the best possible bound on $r(S)$, we proceed as follows. Let w be the weight of the tree, $r = 1$. As long as $\frac{w}{D_{\text{TRP}}} > r$, we increase r by one, decrease w by the weight of the heaviest edge in the tree and increase it by the weight of the cheapest not yet considered edge from S to the depot node H. The idea is to subdivide the component in many components, each served by one vehicle. The updated value of w provides a lower bound on the length of the route needed to serve these new components. Hence, the final value of r is a global lower bound on the number of vehicles needed to serve the nodes in S. Should this procedure fail and not lead to $r > 1$, we apply a final heuristic based on the TSP-relaxation to an assignment problem. We build a bipartite graph by duplicating the nodes in S into A and B. For every original edge (u, v) we introduce the two edges (u_A, v_B) and (u_B, v_A) with the weight of (u, v). Furthermore, we introduce a new node for each partition. This node represents the depot node. We connect the depot node of partition A to all nodes in B excluding the depot node with edges of weight as in the original graph. These edges represent the trip from the last node to the depot. Similarly, we connect the depot node of the partition B to all vertices in A (excluding the depot) with edges of weight zero. These edges represent a zero cost edge from the depot to the start of the path. The weight of the minimum weight bipartite matching is a lower bound on the length of the minimum route needed to serve the nodes in S. Hence, if the weight of the bipartite matching exceeds D_{TRP}, at least two vehicles are needed to serve the nodes in S.

These two relaxations lead to the value $\kappa(S)$.

3 Shunting

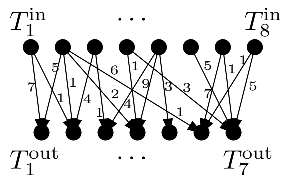
At the center of the operation in a hub-and-spoke setting of a railway system, there is the shunting operation at the hub. So far we have only a very basic understanding of this process. It seems that the problem is not very well studied in the literature, only a few contributions consider a shunting yard with a hump [6, 7].

As a first and most simplistic modeling of the hub we focus on the capacity of the shunting yard. Assuming that the composition of incoming and outgoing trains is given, it remains to be decided in which sequence the trains should be scheduled to arrive and depart, such that the capacity of the shunting-yard at the hub is not exceeded. In this section we first show that this problem is NP-complete, even in a very restricted setting. Next we discuss how to solve the problem in practice by an ILP-formulation. Finally, we consider the problem of optimally grouping the shunting operations in the shunting yard.

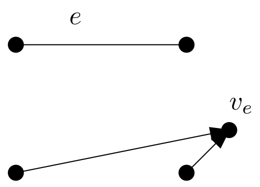
3.1 Hardness of Directed Minimum Cut Linear Arrangement

The sequencing problem for incoming and outgoing trains turns out to be NP-hard, already in a very simple version.

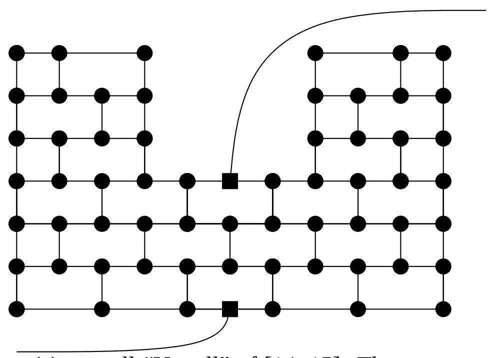
Corollary 3.1 (of Theorem 3.3). *It is NP-hard to decide if a collection of incoming and outgoing trains can be sequenced such that the capacity of the shunting yard is sufficient, even if every incoming train consists of precisely 3 cars, and every outgoing train of precisely 2 cars.*



(a) in-out graph G_{io}



(b) the transformation of an undirected edge to a pair of directed edges to a new vertex



(c) a small "U-wall" of [14, 15]. The square vertices connect to other square vertices of U-walls

Figure 3.1: Illustrations for Minimum Cut Linear Arrangement

Given the composition of the incoming trains $\mathcal{T}^{\text{in}} = \{T_1^{\text{in}}, \dots, T_m^{\text{in}}\}$ and the outgoing trains $\mathcal{T}^{\text{out}} = \{T_1^{\text{out}}, \dots, T_n^{\text{out}}\}$, the sequencing task at hand can be depicted by the bipartite graph $G_{\text{io}} = (U \cup V, E)$ in Figure 3.2(a), the *in-out graph*. The incoming trains correspond to vertices in U, the outgoing trains to vertices in V. We model precedence constraints by directed edges, for every car from its arriving train to its departing train, expressing that a car needs to arrive (with its train) before it can depart. We call G_{io} a *uniformly directed bipartite graph*, because all edges are directed from U to V. An in-out graph is allowed to have parallel edges, or alternatively it can have weights c on the edges, indicating how many cars of an incoming train an outgoing train receives. The sequencing task corresponds to finding a *linear arrangement* of the graph G, i.e., an embedding of the graph onto the horizontal line, such that all edges are directed from left to right. For such an arrangement, the maximal number of edges crossing any vertical line is the *(cut-) width*, and it corresponds to the maximal number of cars residing in the shunting-yard. The width of a graph G is given by the minimal width of a linear arrangement of G. Conversely, any uniformly directed bipartite graph can be understood as an in-out graph. Hence Corollary 3.1 follows indeed from Theorem 3.3 below.

Let $L\colon U \cup V \to \{1, \dots, n\}$ be an optimal linear arrangement of the uniformly directed bipartite graph $G = (U \cup V, E)$. We can assume that in L every outgoing train departs as early as possiblethat is, as soon as all its cars are available. Conversely, there is no use in scheduling an incoming train to arrive before some of its cars are needed. Together this means that given the sequence of the incoming trains it is easy to compute an optimal sequence of the outgoing trains, and vice versa.

Without the directions and the restriction to bipartite graphs, this problem is known as the "minimum cut linear arrangement", a well studied NP-complete problem [8, GT44] that was shown to remain NP-hard for graphs of degree 3 [14], and even planar graphs of degree 3 [15]. We extend these results in the following way.

Lemma 3.2. *For any constant $c > 0$ it is NP-hard to approximate minimum cut linear arrangement with an absolute error of c, even on planar graphs with degree 3.*

Proof. By reduction from the NP-hard problem "minimum cut linear arrangement for planar graphs" [15]. We follow closely the reduction presented in [15]. Let G be a planar graph, and ℓ the bound on its width. We construct G' by taking a U-wall (see Figure 3.1(c)) of nodes with degree 3 for every vertex of G. G' has the property that no two U-walls can significantly overlap in any linear arrangement. (This idea goes back to [14].) The edges of G are replaced by edges in G' connecting vertices of the inner parts of the two U-walls, the square vertices in Figure 3.1(c). As limit L for the width of G' we use the height of the U-walls plus the bound ℓ on the width of G. Now from any linear arrangement that obeys this limit L we can reconstruct an arrangement of the original graph G that has width ℓ. To extend the result in the sense of the lemma, we "multiply" the construction by a factor c, i.e., we use c-times bigger U-walls, and replace every original edge by c new edges. If there is a linear arrangement of the original graph of width ℓ, the constructed graph G' has width cL. Conversely, even from an arrangement of the new graph of width $cL + c - 1$, we can reconstruct a linear arrangement because the U-walls still cannot overlap significantly, and this linear arrangement has width $c\ell + c - 1$. Because every original edge is represented by c parallel edges, every cut is divisible by c, and hence this linear arrangement actually has width $c\ell$, hence the original graph G has width ℓ. $\square$

Theorem 3.3. *It is NP-hard to decide if a uniformly directed bipartite planar graph of out-degree 3 and in-degree 2 admits a linear arrangement of width ℓ.*

Proof. By reduction from the problem of approximating the width of a planar graph with an additive error of 7 of Lemma 3.2. Let G be the undirected planar graph and L be the width limit defining an instance of that problem. Then G either has width $\leq L$ or $\geq L + 7$, and it is NP-hard to distinguish these two cases. We construct a graph G' by replacing every edge e with a pair of edges directed toward a new vertex v_e, see Figure 3.1(b). This graph G' is also known as the node-edge incidence graph with the links directed from nodes to edges (or vice-versa, this is just symmetric). We set the width limit $\ell = L + 6$.

Any optimal linear arrangement of G' will place all the edge-vertices v_e as far left as possible, because not doing so can only increase the width. The vertices of G are also vertices of G', such that the above observation allows us to directly map arrangements of G to arrangements of G' and vice versa. Then directly to the left of an original vertex v, the width of G' is the same as the width of G. Only to the right of it, it is increased by twice the number of neighbors of v in G that are arranged left of v: For a neighbor u of v in G that is arranged left of v, the directed edge (u, v_e) continues up to v_e, and there is the additional edge (v, v_e).

Concluding we see that if G has width $\leq L$, then G' has a linear arrangement of width $\leq \ell = L + 6$, but if the width of G is $\geq L + 7 > \ell$, then G' has width $\geq L + 7 > \ell$. $\square$

This hardness result is complemented by the following consideration.

Theorem 3.4. *Every uniformly directed bipartite graph with maximum degree 2 admits a linear arrangement of width 4, and it takes linear time to determine the minimal width of such a graph.*

Proof. A graph of maximum degree 2 decomposes into cycles and paths. A single edge has width 1, two directed edges have width 2, a path has width 3, and a cycle has width 4 (consider the last incoming train, it adds 2 cars to a shunting-yard containing two cars). $\square$

3.2 Solving the Arrangement Problem in Practice

As the instances of the arrangement problem that arise in our setting are not too large, we can solve them by a simple ILP formulation.

For this formulation we discretise the time horizon into $t_{\max}$ points in time $Z = \{0, \ldots, t_{\max} - 1\}$. We model the problem by Boolean variables $a_{T,t}$ and $d_{T',t}$ that model arrival (and departure) of the trains $T \in \mathcal{T}^{\text{in}}$ ($T' \in \mathcal{T}^{\text{out}}$, respectively) at time $t \in Z$. Here, we assume that it takes a constant number σ of time units to compose an outgoing train after its last cars have arrived at the shunting yard. We refer to E as the edge set of the in-out graph G_{io}.

Equations (3.1a), (3.1b) and (3.1h) impose that, for every edge e, the variables $a_{e,\cdot}$ and $d_{e,\cdot}$ form a monotone sequence starting with 0 and ending with 1. The idea is that the train arrives (or departs, respectively) at the time when the 0-1 transition takes place. Constraints (3.1c) and (3.1d) enforce that an outgoing train can only depart if all its cars have arrived and that σ time units are needed for shunting those cars. Constraints (3.1e) enforce the capacity constraints. Constraints (3.1f) and (3.1g) introduce time constraints for the earliest arrival / latest departure of trains.

The constraints (3.1f) and (3.1g) allow us to construct a solution to the entire problem as follows: First construct, by the methods of Section 2, the optimal routings to and from the hub. Then inspect the length of the routes. These lengths together with the earliest possible departure times (latest possible arrival times) at the stations yield earliest possible arrival times (latest possible departure times) at the hub.

Our experiments show that for our problem instances we can calculate a shunting schedule for a given routing to and from the hub in a few minutes.

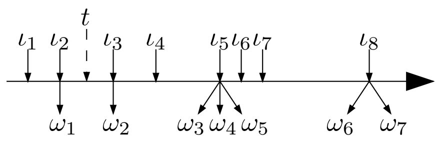
$$
\begin{array}{lllll}
\textbf{ARR:} & \text{Find} \quad (a,d) & & & \\
\text{s.t.} & a_{T,t} & \le a_{T,t+1} & \forall T \in \mathcal{T}^{\text{in}}, t \in Z & (3.1a) \\
& d_{T,t} & \le d_{T,t+1} & \forall T \in \mathcal{T}^{\text{out}}, t \in Z & (3.1b) \\
& a_{T,t} & \ge d_{T',t+\sigma} & \forall t \in \{0,\dots,t_{\max}-\sigma\},\ e=(T,T') \in E & (3.1c) \\
& d_{T',t} & = 0 & \forall T \in \mathcal{T}^{\text{out}}, t \in \{0,\dots,\sigma-1\} & (3.1d) \\
& \sum_{e=(T,T')\in E} c_e(a_{T,t}-d_{T',t}) & \le C & \forall t \in Z & (3.1e) \\
& a_{T,t_i} & = 0 & \forall T \in \mathcal{T}^{\text{in}} \text{ s.t. } T \text{ must arrive after } t_i & (3.1f) \\
& d_{T',t_i} & = 1 & \forall T' \in \mathcal{T}^{\text{out}} \text{ s.t. } T' \text{ must depart before } t_i & (3.1g) \\
& \multicolumn{3}{l}{a_{T,0}=0, \quad a_{T,t_{\max}-1}=1, \quad d_{T',0}=0, \quad d_{T',t_{\max}-1}=1 \quad \forall T \in \mathcal{T}^{\text{in}}, T' \in \mathcal{T}^{\text{out}}} & (3.1h) \\
& \text{all } a_., d_. \in \{0,1\} & & & (3.1i)
\end{array}
$$

3.3 Optimal Grouping of Shunting Operations

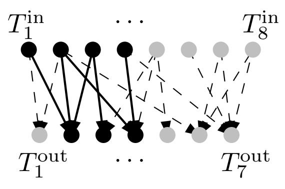
In this section, we take a closer look at the shunting operation. Assume that we have already found a good order for the trains to arrive at the central shunting yard. More precisely, let us assume that we have computed targeted arrival times $\mathcal{I} = \{\iota_1, \dots, \iota_m\}$ of the incoming trains $\mathcal{T}^{\text{in}}$, for example by the methods of the last section.

From this, we can compute the earliest possible departure times $\mathcal{O} = \{\omega_1, \dots, \omega_n\}$ of the outgoing trains $\mathcal{T}^{\text{out}}$ as follows. The earliest possible departure time ω_i of an outgoing train T_i^{out} is the latest arrival time ι_j of an incoming train T_j^{in} that has cars for T_i^{out}: $\omega_i = \max_{j:(T_j^{\text{in}},T_i^{\text{out}})\in E} \iota_j$ with respect to the edge set E of the in-out graph G_{io}. Note that these earliest possible departure times do not include the time needed for shunting in contrast to the actual departure time that are calculated by the algorithm the we present in this section. It follows that the earliest possible departure times are a subset of the arrival times ($\mathcal{O} \subseteq \mathcal{I}$), see Figure 3.2(a). We denote the total number of cars of T_j^{out} by $|(T_j^{\text{out}})|$. The trains are indexed w.l.o.g. in the order of their arrival times/earliest possible departure times.

The literature addresses the task of static shunting [10, 16, 6, 7], i.e., the situation where all trains arrive before the shunting operation starts, and leave only after it has been completed. The dynamic aspect of the situation where the shunting operation can (and has to) start with only some of the incoming trains available, seems not to have been investigated. Here, we consider a simple model, the *grouped shunting*, in which we periodically shunt the outgoing trains for which all cars have arrived at the shunting yard. At time t all cars that are in the shunting yard correspond to a subgraph G_t of G_{io}, see Figure 3.2(b) for an example. If we start a shunting phase at time t, the set of all cars O_t in the shunting yard at time t belonging to complete outgoing trains are composed. In Figure 3.2(b) the set O_t corresponds to all nodes in the bottom partition that have



(a) time line with times of incoming and departing trains



(b) $G_t \subset G_{\text{io}}$ represents a possible configuration of cars in the shunting yard at time t (solid edges), cf. Fig. 3.1(a).

Figure 3.2: grouped shunting example

all their adjacent edges in G_t, i.e T_2^{out}. The remaining cars are left in the shunting yard. We further assume that the time needed for shunting O_t depends on the number of cars in O_t, denoted by $|O_t|$. We assume that the shunting time is given by a monotone concave function $f : \mathbb{N} \to \mathbb{R}^+$, where $f(n)$ is the time needed to shunt n cars. Note that the concavity just states that a static shunting task for some set of cars cannot take longer than breaking this set up into subsets and sequentially perform the static shunting on these subsets. This property trivially holds for all sensible static shunting methods. Given a shunting operation starting at time t, the outgoing trains are composed in the time interval $[t, t + f(|O_t|)]$, and during that time no other shunting operation can take place.

Our task is to decide how to group the shunting operations, i.e., at which points in time we should start to shunt. Observe that the cars that are in the shunting yard at time t depend on the grouping decisions before t. For the objective of makespan minimization ($C_{\max}$) we call the problem the *optimal grouping problem with makespan objective*. It can be solved by dynamic programming as follows:

For each $\omega_i \in \mathcal{O}$, the algorithm maintains a state $W(\omega_i) = (t', v')$ with the following properties. Time t' is a point in time within the interval $I_i = [\omega_i, \omega_{i+1})$ and v' represents the minimum number of cars available for shunting at t'. Together, the pair (t', v') represents a *partial solution until (interval)* i, that is, a solution to the problem restricted to the intervals up to I_i, which has ended shunting before or at time t' and has v' cars available for shunting. The interval for the last point $\omega_{\max} \in \mathcal{O}$ is defined as $I_{\max} = [\omega_{\max}, \infty)$. For convenience, we write $t = W^t(\omega)$ and $v = W^v(\omega)$ for $W(\omega) = (t, v)$.

The key idea of the algorithm is that it suffices to store a single state for each interval. We express this by a dominance rule for two states of the same interval. The state (t, v) *dominates* (t', v') if and only if $t + f(v) < t' + f(v')$.

```
Procedure DOMINANCE(t,v,t',v')
    if $t + f(v) < t' + f(v')$ then return $(t, v)$ else return $(t', v')$
```

The following lemma makes the usefulness of dominance precise.

Lemma 3.5 (Dominance Rule). *Consider a solution S with makespan $C^S_{\max}$ which starts a shunting phase with v cars at time t of interval I_i. Assume further that a state (t', v'), $t' \in I_i$ exists that dominates (t, v). Then, a solution S' exists, which starts shunting with v' cars at time T' and has a makespan of less than or equal to $C^S_{\max}$.*

Proof. As (t', v') dominates (t, v), we can construct a solution S' as follows. The existence of (t', v') guarantees that a partial solution P until i exists. We build S' by using P up to t'. At t' we start a shunting phase that ends at $e' = t' + f(v')$, i.e., before $e = t + f(v)$, where the corresponding shunting phase in S ends (by definition of dominance). If t is the start of the last shunting phase in S we already have a complete solution with a shorter makespan. Otherwise, the rest of the new solution consists of the grouping decisions in S at or after e'. Let s_{next} be the start of the first shunting phase at or after e'. Note that $s_{\text{next}} \geq e > e'$ since (t', v') dominates (t, v). At s_{next}, S' has exactly the same number of cars available for shunting as S has, since t and t' are in the same interval. The cars available at s_{next} in S and S' are just the weights of outgoing trains in the interval from $[t, s_{\text{next}})$ resp. $[t', s_{\text{next}})$. These two values are identical. □

A solution S induces a set of states in the intervals in which it starts and ends shunting and in the intervals in which it waits. For the starting and ending phases this is the exact time at which the shunting starts or ends together with the number of cars available for shunting at these times. A solution that waits in an interval I_i induces the state (ω_i, v), where v is the number of cars available for shunting at ω_i. We say that a *solution dominates a state s* if it induces a state s' in the interval of s that dominates s. Similarly, we say that a solution S dominates a solution S' if S induces a state that dominates a state of S'. Because of Lemma 3.5, it is sufficient to consider undominated solutions when searching for an optimal solution. We introduce the same notation for partial solutions until i, which only induce states in intervals $I_j,\ j \leq i$.

The dynamic program proceeds as follows, see Algorithm 2 for a precise formulation. First, we initialize prefix sums $S(\omega)$ for each event point. These sums stand for the cumulated number of cars of all outgoing trains up to time ω. Then we iterate over the events chronologically and update the W values. The crucial observation is that shunting at time $t = W^t(\omega)$ means that we keep the shunting yard busy for at least $\sigma = f(W^v(\omega))$ time. Let ω' be the event point directly before $t' = t + \sigma$. To find this event ω', we need a dictionary on $\mathcal{O}$ that supports predecessor queries. If we decide to start a shunting phase at t, then there is a feasible solution with state $(t', S(\omega') - S(\omega))$ in the interval of ω'. We use the dominance rule to find out if this state should replace the current state in the interval of ω'. In order to account for the possibility of not shunting directly after t', we also have to update all states in intervals after ω', which we do implicitly in line 2 before accessing $W(\omega)$. After the last iteration, the values W reflect an optimal solution. In order to find the minimum makespan, we need to add one extra state after the last event. In this state we calculate the finish time after the additional shunting operation at the end, i.e. the makespan.

Algorithm 2: Optimal grouping

```
   //Initialize Prefix Sums (in linear time in the obvious way)
   forall $\omega \in \mathcal{O}$ do $S(\omega) \longleftarrow \sum_{i:\omega_i \le \omega'} |T_i^{\text{out}}|$
   //Initialize States
   $W(\omega_1) \longleftarrow (\omega_1, |T_1^{\text{out}}|)$
   $C_{\max} \longleftarrow \infty, v_{\text{old}} \longleftarrow 0$
   //Iterate
1  forall $\omega \in \mathcal{O}$ in chronological order do
2      $(t, v) \longleftarrow W(\omega) \longleftarrow$ DOMINANCE$\big((\omega, v_{\text{old}} + |T_i^{\text{out}}|), W(\omega)\big)$
       $t' \longleftarrow t + f(v)$
3      if $t' < \omega_{\max}$ then
4          $\omega' \longleftarrow$ PREDECESSOR$(t')$
5          $W(\omega') \longleftarrow$ DOMINANCE$\Big(W(\omega'), \big(t', \omega + f(S(\omega') - S(\omega))\big)\Big)$
       else
6          $C_{\max} \longleftarrow \min\big\{C_{\max}, t' + f(S(\omega_{\max}) - S(\omega))\big\}$
       $v_{\text{old}} \longleftarrow v$
   return $C_{\max}$
```

Theorem 3.6. *Algorithm 2 solves the grouping problem with makespan objective in $\mathcal{O}(n \log n)$ time.*

Proof. We prove the correctness of the algorithm by the following invariant:
At the end of the i-th iteration of the forall loop 1 the following two properties hold:

INV$1(i)$ *For all intervals I_j, with $j \le i$ the state $(t, v) = W(\omega_j)$ is not dominated by any partial solution until i.*

INV$2(i)$ *No undominated partial solution until k which starts to shunt the last time before ω_i and ends the shunting in interval $I_k, k > i$ exists which dominates $W(\omega_k)$.*

Note that dominance for the last interval $[\omega_{\max}, \infty)$ is equivalent to a better makespan. Therefore, the correctness of the invariant implies the correctness of the algorithm.

To prove the invariants, consider iteration i and the corresponding state $(t, v) = W(\omega_i)$. For INV1(i) we have to set (t, v) to a state that is not dominated. Such a state corresponds to a specific partial solution until i. If that solution ends a shunting phase in I_i then $W(\omega_i)$ is already set correctly by INV2$(i-1)$. If this is not the case, then this solution ends a shunting phase before I_i and waits in I_i. In this case $(\omega_i, W^v(\omega_{i-1}) + |T_i^{\text{out}}|)$ is a non dominated state and it is assigned to $W(\omega_i)$ in line 2. This makes use of the fact that $W^v(\omega_{i-1})$ is calculated correctly because of INV1$(i-1)$. After line 2 the state $W(\omega)$ cannot be dominated by another state and INV1(i) holds.

Line 5 creates the state that corresponds to a start of a shunting phase in i and updates the interval in which the phase ends. After this update INV2(i) holds: We have to check the property for all undominated partial solutions until k that start a shunting phase in I_i, for the others it is clear from INV2$(i-1)$. We know from INV1(i) that in I_i the state $W(\omega_i)$ cannot be dominated. Therefore, any undominated solution that starts shunting in I_i has to end this shunting phase in the interval of ω', see Lines 4,5. This implies that we do the only necessary update to preserve the second property.

We can use a balanced search tree for the predecessor queries which guarantees a running time of $\mathcal{O}(n \log n)$. □

4 Experiments

We implemented our routing approach to evaluate whether we can solve practical instances with it. To that purpose, we created test instances that are based on the real SBB Cargo network and on their supply-demand matrix. The couple time was set to a realistic value, the distance constraints were set to the maximum distance from the hub increased by a specific percentage. As for the other parameters, we could not set all of them to their real values. The major deviation from the actual problem setting was in the capacity constraint. Currently we do not consider that a single supply can be collected by two trains if it exceeds the train capacity. Hence we had to choose a high excess capacity, i.e. the total available capacity exceeds the total necessary capacity. This choice is partially justified because the emphasis on fast transportation usually makes the Cargo Express freight trains much shorter than normal freight trains. On the other hand, note that for a prescribed subdivision of a big supply it is possible to incorporate this feature by splitting nodes in the graph. We did not exploit that some supply points can be discarded due to the transport to other stations with the shunting engines. Without this assumption, we would have to set the capacity constraint to completely unrealistic values. On the other hand, the created instances become bigger than the instances with shunting engines.

We implemented the branch and cut approach of Section 2.2 in SYMPHONY 5, and extended the existing VRP module by Ralphs et al. [17]. We used CPLEX 9 as LP solver for SYMPHONY, and LEDA 4.5 for computing the minimum spanning trees and the assignment problems. It allowed us to carry out experiments on a linux workstation running a 3 GHz Pentium 4 processor with 2 GB RAM.

The SBB Cargo network has 46 stations. We used the supply data for testing our implementation. After discarding the stations with zero supply (at these stations there are only demands), the remaining network has 38 stations. We generated smaller instances by taking only a subset of the nodes (in no particular order).

As a first set of tests, we compared instance sizes to running times. Given a set of nodes S to be served by K trains, we set the capacity of the trains to $C = 3.8\frac{\sum_{v\in S} d(v)}{K}$. The excess capacity of 280% is the smallest for which all these instances are feasible for the reasons explained above. The proportional increase makes all different size instances similarly tight from the point of view of the capacity constraints. We set four different length constraints of each route: 1, 1.1, 1.25 and 1.5 times the longest distance from a supply point to the depot, respectively. Note that these bounds become tighter as the instance becomes bigger (since the total traveling time increases). Figure A.1(a) plots instance size versus running time with zero couple time. Figure A.1(b) plots instance size versus running time with a couple time equivalent of 12km and a length constraint with factor 1.25.

The experiments show that already instances of size 38 are challenging to solve. Usually, simple VRP instances of the same size are relatively easy to solve. The added complexity stems from the additional length constraints and the additional flexibility to choose a starting point. We are currently considering a more involved preprocessing and the addition of more problem specific separation heuristics.

In the second experiment we choose a fixed network of size 33 and calculate the solution cost depending on the number of employed freight trains. As the total solution cost depends both on the total traveled distance and on the number of trains, we plot this cost for realistic assumptions on both train and route costs. For these instances we chose a constant maximum capacity such that for the minimum number of trains (6) we have 100% excess capacity. For the couple time we choose an equivalent of 12 km.

This experiment shows how to minimize the total operational routing costs. Due to the high capacity constraints a solution with only six trains has the minimum cost. Because of the high costs of operating a train, we expect that in general the feasible solution using the least trains has the minimum cost.

5 Conclusion and Outlook

In this paper, we describe different aspects of freight transportation and identify challenging combinatorial problems. Our solution approach for the routing problem seems promising, both in terms of formulating it as an ILP, and also in the choice of the software platform. Nevertheless, our current version needs to be improved in order to solve the real SBB Cargo network to optimality, but the real world instances are within the reach of the approach. To that aim, it is necessary to analyze the combinatorial structure of the problem further, for example to derive more effective separation heuristics. At this stage it is unclear, how our approach scales to significantly bigger instances.

Additionally, we provide new theoretical insight in some aspects of the shunting operation. These aspects only consider the sequence of the shunting operations with respect to the capacity of the shunting yard, but do not consider the detailed operation at the shunting yard.

So far, we separate the shunting problem from the routing problem. The combination of the shunting aspects may change the solution of the routing aspect considerably, and vice versa. Hence, the next step is to integrate both aspects into one approach.

6 Acknowledgment

We thank the team from SBB Cargo Basel for their great support in providing us with data and answering all conceivable questions about freight rail systems. Further, we thank Ted Ralphs for his kind support on SYMPHONY related issues.

References

[1] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, and D. Naddef. Computational results with a branch and cut code for the capacitated vehicle routing problem. Research Report 949-M, Université Joseph Fourier, Grenoble, 1995.

[2] U. Blasum, M. R. Bussieck, W. Hochstättler, C. Moll, H. H. Scheel, and T. Winter. Scheduling trams in the morning. *Mathematical Methods of Operations Research*, 49(1):137–148, 1999.

[3] U. Blasum and W. Hochstättler. Application of the branch and cut method to the vehicle routing problem. Technical Report zpr2000-386, Zentrum für angewandte Informatik, Köln, 2000.

[4] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for vehicle routing. *Mathematical Programming*, 20:255–282, 1981.

[5] G. Clarke and J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[6] E. Dahlhaus, P. Horák, M. Miller, and J. F. Ryan. The train marshalling problem. *Discrete Applied Mathematics*, 103(1-3):41–54, 2000.

[7] E. Dahlhaus, F. Manne, M. Miller, and J. Ryan. Algorithms for combinatorial problems related to train marshalling. In *Proceedings of AWOCA 2000, In Hunter Valley*, pages 7–16, July 2000.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, 1979.

[9] R. Hall. On the road to recovery. *OR/MS Today*, June 2004. available at `http://www.lionhrtpub.com/orms/orms-6-04/frsurvey.html`.

[10] W. Hiller. *Rangierbahnhöfe*. Transpress VEB Verlag für Verkehrswesen, 1983.

[11] G. Laporte, M. Desrochers, and Y. Nobert. Two exact algorithms for the distance-constrained vehicle routing problem. *Networks*, 14:161–172, 1984.

[12] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33:1050–1073, 1985.

[13] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. *The Travelling Salesman Problem*. Wiley, 1985.

[14] F. S. Makedon, C. H. Papadimitriou, and I. H. Sudborough. Topological bandwidth. *SIAM Journal on Algebraic and Discrete Methods*, 6(3):418–444, July 1985.

[15] B. Monien and I. H. Sudborough. Min cut is NP-complete for edge weighted trees. *Theor. Comput. Sci.*, 58(1-3):209–229, 1988.

[16] G. Potthoff. *Verkehrsströmungslehre, Betriebstechnik des Rangierens*, volume 2. Transpress VEB Verlag für Verkehrswesen, 1977.

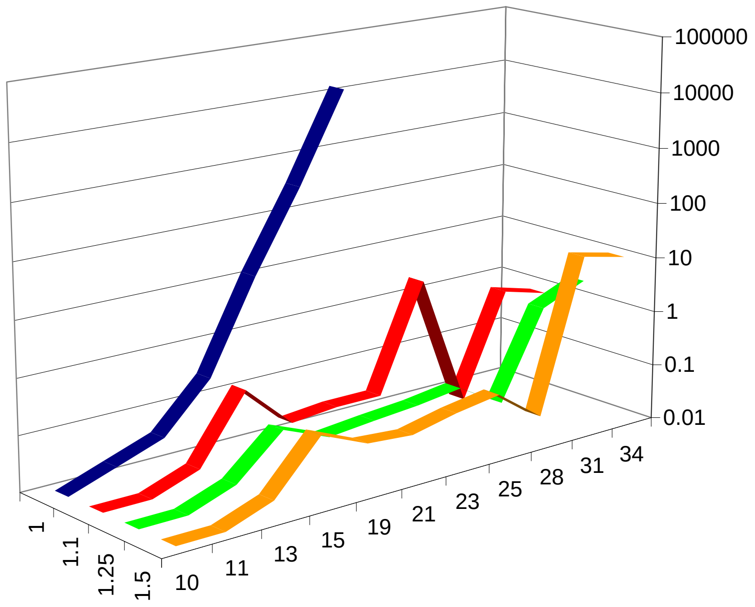
[17] T. K. Ralphs. Symphony 5.0. `http://www.branchandcut.org`.

[18] T. K. Ralphs, L. Kopman, W. R. Pulleyblank, and L. E. Trotter. On the capacitated vehicle routing problem. *Mathematical Programming*, 94(2–3):343–359, 2003.

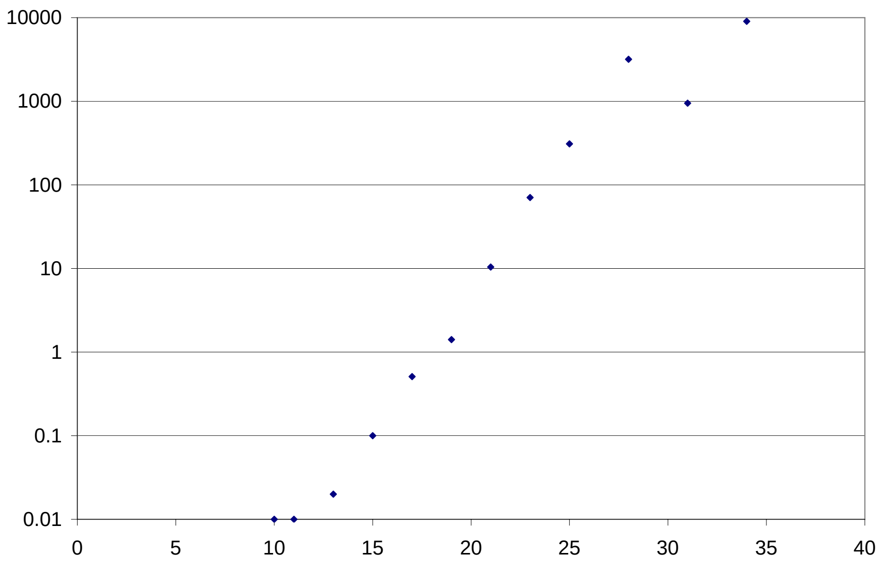
[19] SBB. Cargo express. `http://www.sbbcargo.com/en/index/ang_produkte/ang_produkte_express.htm`.

[20] P. Toth and D. Vigo. *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, 2002.

A Plots and Network of the Experiments



(a) Running times vs. number of trains for 380% capacity instances



(b) Running times vs. number of trains for 380% capacity and 12km couple distance

Figure A.1: Running Times

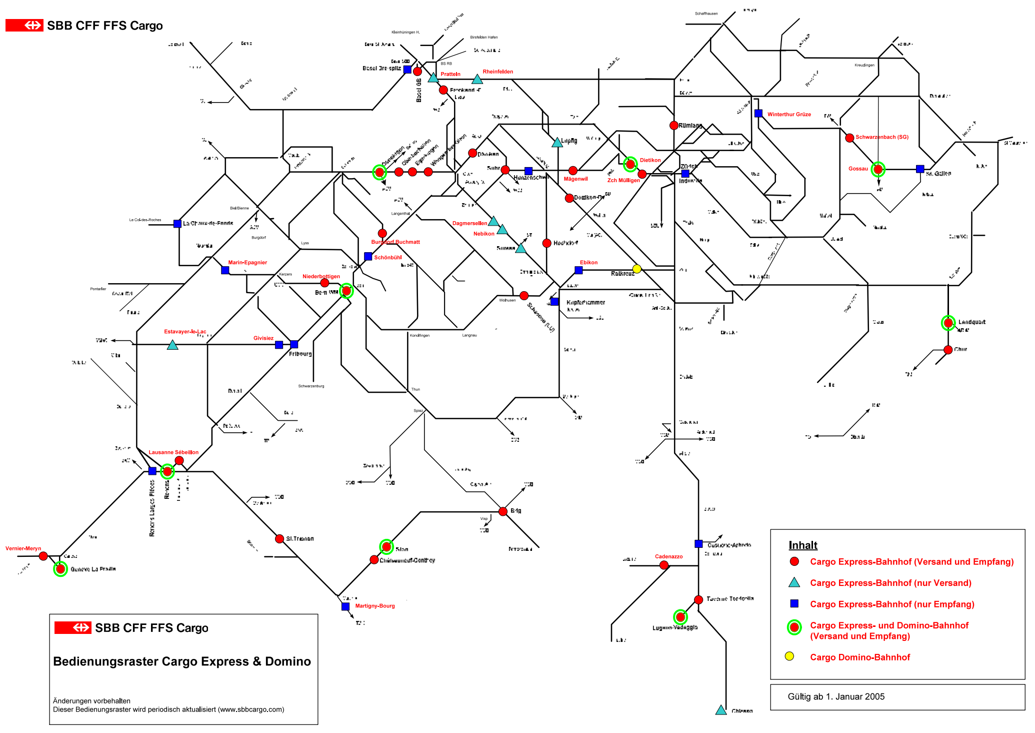


Figure A.2: The SBB Cargo Express network. All but one node (central light colored round node) are supply/demand points for Cargo Express.

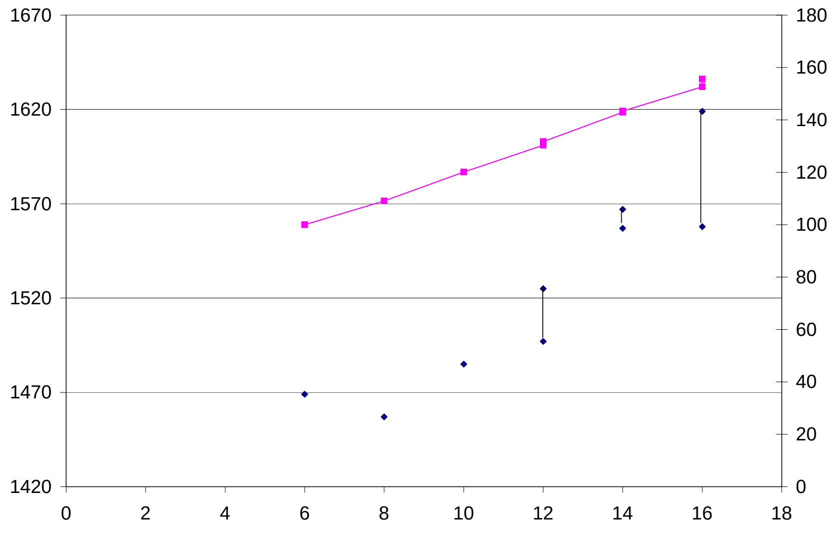


Figure A.3: Solution cost as a function of the number of trains, once as driven distance, and once as combined costs assuming an extra train is as expansive as 100km driving. The vertical bars show the gap between lower and upper bound found by our program.