Data Reconciliation of Freight Rail Dispatch Data

William Barbour [a,1], Shankara Kuppa [b], Daniel B. Work [a]
[a] Department of Civil and Environmental Engineering,
Institute for Software Integrated Systems
Vanderbilt University
[b] CSX Transportation
[1] E-mail: william.w.barbour@vanderbilt.edu

Abstract
In order to enable widespread use of data driven analysis and machine learning methods for rail operations problems, large volumes of operational data are needed. This data has the potential to contain erroneous or missing values, especially given its size and dimensionality. In this work a data reconciliation problem for rail dispatch data is proposed to identify and correct errors, as well as to impute missing data. The data reconciliation problem finds the least-perturbed modification of the historical data that satisfies operational constraints, such as feasibility of meet and overtake events, safety headway, siding allocation, and running time. It also imputes missing values with estimates that satisfy all operational constraints. The data reconciliation method is applied to a large historical dataset from freight rail territory in Tennessee, United States, containing over 3,000 train records over six months. The method identifies and corrects errors in the historical data, and is able to impute data on a synthetically decimated version of the historical data. The quality of the imputed data from data reconciliation is compared to imputed data using naive interpolation. The results show that data reconciliation reduces timing error of imputed points by up to 15% and increases the number of meet and overtake events estimated at the correct historical location from less than 40% to approximately 95%. These findings indicate that the data reconciliation method is a useful preprocessing step for analysis and modeling of railroad operations that are based on real-world physical dispatching data.

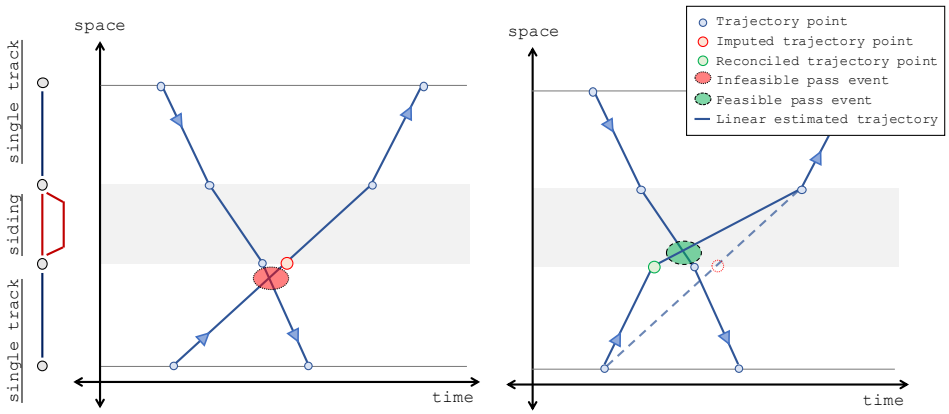
Keywords
data reconciliation, dispatching, modeling, optimization

1 Introduction

1.1 Motivation

Data-driven methods for railroad operations require abundant, high-quality sources of data for model building. Machine learning, and deep learning methods in particular, require large datasets for training. These methods will learn trends from input data, so if the data contains errors, then the errors may propagate into the trained model and the resulting analysis.

A common challenge in the emerging data science and data analytics fields is the amount of time spent on data cleaning and data preparation. Common tasks include standardizing and normalizing data, identifying faulty data and discarding or correcting it, and imputing values for missing entries. Especially when (reasonably) clean data from large systems



(a) Infeasible trajectories.

(b) Feasible trajectories.

Figure 1: A time-space plot example of two trains traveling in opposite directions. Train trajectories are denoted by the blue points and linearly estimated between points. It is obvious that these trains met at the siding track (shaded grey area). This example demonstrates how an erroneous trajectory point (red point) in (a) can result in an infeasible meet location (shaded red oval). This can be corrected by reconciliation of the timing to the green point in (b) to make the meet occur at a feasible location on the siding track (within the grey area).

that describe physical processes (e.g., freight rail flows) is needed, ad hoc and manual approaches to data preprocessing can easily be inefficient and can often be intractable.

To automate some aspects of data cleaning and data preparation, it is possible to use knowledge about the physical constraints of the system to identify and correct erroneous values. The process by which missing data is estimated and erroneous data is corrected using a model as a constraint is referred to as *data reconciliation* (Tjoa and Biegler, 1991). Given that railroad operations have obvious logical and physical constraints, useful data reconciliation problems can be posed and solved, which we demonstrate in this work.

1.2 Overview of Data Errors in Dispatch Data

Train trajectory data typically comes in the form of train arrival times at fixed locations on the rail network; in the United States these are often called *on-station points*, or *OS-points*. Most OS-points are located at the endpoints of passing sidings in single-track territory or at crossover points in multi-track territory. Track segments refer to the sections of track delineated by OS-points. The arrival times of a train at each OS-point between two rail yards or terminals on the network constitute a trajectory.

Data errors are identifiable as infeasible trajectories because they violate *meet* constraints (passing events between trains in opposing directions), *overtake* constraints (passing events between trains in the same direction), headway constraints (trains following, meeting, overtaking with insufficient time headway clearance), or other operational constraints. Data may also be missing, e.g., due to incomplete data fusion or sensor failures, which can further compound the difficulty of identifying and correcting errors.

Consider the data error in Figure 1a, a time-space plot, as an example. Three tracks

are shown on the spatial axis (y-axis): one siding track denoted by the grey shading and two single track segments. Trajectories of two trains traveling in opposing directions are denoted by the blue points, which represent known trajectory points, and the blue lines that are the linearly approximated trajectories between them. In Figure 1a, there is an imputed point shown in red along the trajectory. The timing value of this point results in a meet event (intersection between trajectories highlighted by the red shaded circle) between the two trains that occurs on a single track segment and is therefore infeasible. It is clear that the two trains must have passed each other on the siding and that the imputed trajectory point must be incorrect. Indeed, by relocating the erroneous imputed point to the location of the green point in Figure 1b, a feasible set of trajectories is found where the meet event occurs on the siding (highlighted by the green shaded circle). The amount by which the imputed point must be moved is dictated by the safety headway.

This simple example demonstrates one types of data error that can be encountered in rail operations data, noting that in real datasets the errors can be more complex, randomly distributed in the dataset, and can be compounded by missing values. As a consequence, ad hoc or manual approaches to diagnose and fix infeasible data are not viable on national-scale rail networks that move thousands of trains daily.

1.3 Problem Statement and Contribution

The main contribution of this work is the development of a method to perform optimization-based *data reconciliation* of railroad dispatching data. Given a set of train trajectories and as set of operational constraints, the data reconciliation problem simultaneously corrects any data that is infeasible, and also imputes any missing data.

A constraint set from a dispatch optimization problem that models single-track rail operations is used to perform data reconciliation and we note that the method may be generalized to other optimization-based dispatch tools and network topologies. To illustrate the performance of the method, the data reconciliation problem is implemented on a real freight rail dataset with synthetic omissions in the data. This is the first work to formalize data reconciliation for cleaning rail dispatch data, which can be a critical step for machine learning and data-driven rail operations and is a practical challenge in the transportation industry.

The remainder of the article is organized as follows. Section 2 reviews the related work on optimal dispatching, data driven rail operations, and data reconciliation problems posed in other domains. Section 3 provides a general forms of optimization-based dispatching and its relationship to the data reconciliation problem. Section 4 instantiates a specific data reconciliation problem used in the work on a real dataset from a US Class-1 railroad. In Section 5, we present and discuss results from applying the reconciliation to the historical dataset and to a synthetically incomplete dataset.

2 Background

First a selection of prior work pertaining to posing and solving the rail dispatch problem with optimization models is summarized. These models define the basis of logical and physical constraints for rail operations that are used by the data reconciliation problem. We then explain the need that data-driven rail analysis techniques have for large volumes of clean historical data and provide examples of such work. Finally, we discuss prior work on data reconciliation from other domains of study.

Optimization-based Rail Dispatch

Optimization-based rail dispatch is a common tool in passenger and freight railways around the world. Many rail dispatch and control schemes still require humans in the loop, but actions and plans are often suggested by *computer aided dispatching systems* (CAD) (Petersen et al., 1986). These systems are given the physical and logical constraints of the network, which include network topology, speed limits, signalling, train passing logic, and train physics, alongside railroad operating practices and preferences, which include train schedules, train priority, and delay recovery (Wang and Goverde, 2016; Khoshniyat and Peterson, 2015). The routing problem considers these many factors and constraints and, along with the size of the rail network, results in a large *mixed integer linear program* (MILP) (Bollapragada et al., 2018; Higgins et al., 1996). These problems are applicable at multiple levels of operations, including tactical planning, daily operations, and re-scheduling, as outlined by Törnquist (2006). Ultimately, we show how to extend these exact types of problem formulations to the data reconciliation problem that ensures feasibility of operational data that is collected.

One of the first formal definitions of the CAD problem was by Petersen et al. (1986). Higgins et al. (1996) focused on a similar CAD model as a decision support tool for single-line railways. Murali et al. (2016) used an *integer programming* (IP) model for tactical planning on the Los Angeles rail network. The intractably large MILP problems created by timetabling over a large geographical area and long time horizon are addressed with an incremental heuristic by Gestrelius et al. (2017). Wang and Goverde (2016) took a detailed approach to trajectory optimization from the energy conservation perspective with consideration of train performance calculations. Robust train timetabling was addressed with variable time headways by Khoshniyat and Peterson (2015). Törnquist and Persson (2007) studied the effects of different optimization objectives using a heuristic technique for disturbance re-scheduling on a mixed passenger/freight traffic corridor in Sweden. Bollapragada et al. (2018) describe a modern optimization-based system that handles train dispatching and other ancillary activities used at Norfolk Southern Railway in the United States. Much of the dispatching, scheduling, and disruption management literature is well-summarized by Fang et al. (2015) at the strategic, tactical, operational, and rescheduling levels.

Data-driven Rail Analysis

As previously discussed, train routing and control problems are difficult and nuanced, but are increasingly the focus of railroads seeking to further optimize and automate operations. Less work has been done on the post-hoc analysis of dispatching and dispatcher performance and this line of inquiry could have implications in safety, sustainability, and automation. See Ghofrani et al. (2018) for a review of many of the applications in this field. All of these data-driven methods require large volumes of reasonably clean historical dispatch data due to the unique topology of the rail network and the complexity of operations (Wang and Work, 2015; Barbour et al., 2018b; Oneto et al., 2019; Ghofrani et al., 2018).

Oneto et al. (2019) analyzed train behavior on a network scale using both prior network knowledge and historical data. Kecman and Goverde (2015) estimated passenger train running and dwell times in the Netherlands in real time using numerous machine learning models. Chapuis (2017) used artificial neural networks to produce arrival time estimates for French passenger trains. Wang and Work (2015) used historical data on Amtrak trains in the United States, which run with far higher variability than their international counterparts, along with vector regression to estimate arrival times. Support vector regression, ensemble

decision trees, and deep learning were all used to estimate arrival times of freight trains in the United States, which operate with a low degree of scheduling and high variability (Barbour et al., 2018a,b).

Data Reconciliation

The number works focused on data reconciliation is rather limited. An optimization-based data reconciliation problem was introduced by Tjoa and Biegler (1991), where chemical process measurement and control data was studied with respect to noise reduction and gross error correction. Leibman et al. (1992) develop a new method for data reconciliation using nonlinear programming targeted at dynamic and nonlinear environments. Tong and Crowe (1995) introduced the use of principle component analysis for gross error detection in data reconciliation, as an alternative to some statistical tests. Soderstrom et al. (2001) performed gross error detection and data reconciliation simultaneously by formulating and solving a mixed integer linear program for process flows.

In the transportation field, Zhao et al. (1998) used data reconciliation techniques for processing traffic count data under flow conservation constraints. Claudel and Bayen (2011) perform data reconciliation for highway traffic data, posed as a convex program based on constraints derived from a partial differential equation describing conservation constraints.

3 Optimal Dispatch and Data Reconciliation

In this section we first explain the generalized problem formulation of optimization-based dispatching and the corresponding data reconciliation problem formulation.

3.1 Optimal Dispatch Problem

The optimal dispatch problem takes a set of trains traveling on a section of the network (e.g., between major yards or terminals) and finds feasible trajectories that are optimal with respect to minimization of a function of weighted train runtime and satisfy physical and operational constraints. Broadly, many dispatch problems can be posed in the general form:

$$\begin{aligned} \underset{x,z}{\text{minimize:}} \quad & f(x,z) \\ \text{subject to:} \quad & A_1 x + A_2 z \leq b, \end{aligned} \tag{1}$$

where the decision variables are $x \in \mathbb{R}_+^p$ and $z \in \mathbb{Z}^q$. In a common formulation, the decision variables x encode times at which trains reach various points on the network, while the integer decision variables z encode dispatching logic that indicates if and where meets and overtakes occur on the network. The objective function $f(\cdot,\cdot)$ is a performance measure that quantifies the desirability of the dispatch solution, for instance with respect to delay or priority weighted delay of trains. Integer variables may factor into the objective function if, for example, one wishes to minimize the total number of meets and overtakes. The physical and operational constraints, such as the permissible locations of meet and overtake events, headway constraints, and train travel times, are encoded in the inequality constraints $A_1 x + A_2 z \leq b$. For simplicity the constraints are assumed to be mixed integer linear, although more general dispatch problems can also be considered.

3.2 Data Reconciliation Problem

With a generic form of the optimal dispatch problem defined, it is now possible to define the corresponding data reconciliation problem. The constraint set from the train dispatch problem plays a critical role in the data reconciliation problem. Accurate data reconciliation assumes that the constraint set correctly describes the operations of the rail network. Consider a historical trajectory dataset denoted by $\tilde{x}$ and $\tilde{z}$, possibly containing missing entries. Let $\tilde{x}_\Omega$ and $\tilde{z}_\Omega$ denote the subset of the historical dataset for which entries are present. The data reconciliation problem is written as:

$$\begin{aligned} \underset{x,z}{\text{minimize:}} \quad & g(x_\Omega - \tilde{x}_\Omega, z_\Omega - \tilde{z}_\Omega) + h(x_\Psi, z_\Psi) \\ \text{subject to:} \quad & A_1 x + A_2 z \leq b, \end{aligned} \tag{2}$$

where $x \in \mathbb{R}^p_+, z \in \mathbb{Z}^q$. The variables x_Ω and z_Ω are the subset of the decision variables that correspond to the historical dataset for which entries are present and x_Ψ and z_Ψ are the subset of the decision variables that correspond to missing historical entries.The reconciliation problem finds feasible trajectories, x, z, that are feasible and minimally-perturbed from the historical data according to the performance measure $g(\cdot, \cdot)$. An additional term $h(\cdot, \cdot)$ can be added to the reconciliation problem to further regularize the missing data that must be imputed by the data reconciliation problem. Importantly, while the historical data $\tilde{x}, \tilde{z}$ may or may not be feasible, and may or may not contain missing entries, the reconciled data indicated by the decision variables at optimality, x^*, z^*, are feasible and complete provided the constraint set is not empty.

A variety of possible performance measures can be designed for the data reconciliation problem. For example, a natural choice is an $\mathcal{L}_1$ penalty on the historical data:

$$g(x_\Omega - \tilde{x}_\Omega, z_\Omega - \tilde{z}_\Omega) = ||x_\Omega - \tilde{x}_\Omega||_1, \tag{3}$$

which promotes sparsity in the changes to the timing variables from the values in the historical data. Note an $\mathcal{L}_2$ penalty can also be considered, but it may return small changes to many of the entries rather than a few changes to a few entries. In (3), we do not consider a penalty on the integer variables z_Ω even though it is possible, because it requires more care to design and depends on the interpretation of the variables. For example, in the problems instantiated later in this work, the integer decision variables are uniquely determined once the continuous variables are fixed, and the primary objective is to match the timing data as much as possible.

In cases of missing historical data, the design of the regularization term influences the quality of the imputed values found when solving the data reconciliation problem. Supposing again that x denotes timing data, and x_Ψ denotes the vector of entries of x corresponding to the missing data in $\tilde{x}$, one can advance trains as quickly as possible with:

$$h(x_\Psi, z_\Psi) = ||x_\Psi||_1. \tag{4}$$

Letting w encode the priority of trains at the various timing points, one can advance the trains based on priority weights:

$$h(x_\Psi, z_\Psi) = w^T x_\Psi. \tag{5}$$

It is also possible to regularize based on desired timings x^{des} that allow for encoding desired segment speeds (e.g., average speeds) through the sections with missing data. This can be written as:

$$h(x_\Psi, z_\Psi) = ||x_\Psi - x^{\text{des}}||_1. \tag{6}$$

It is also possible to regularize based on the integer variables z_Ψ, to indicate a preference to avoid meets and overtakes, for example.

4 Instantiation of a Data Reconciliation Problem

This section provides an overview of the data reconciliation problem formulation including the parameters, decision variables, the objective function, and constraints.

We limit the discussion to terminology and constraints needed to understand the core functionality of the model. For clarity and brevity, in this abbreviated formulation we do not describe end of train clearance timing, trains entering and exiting in the middle of the network section, multi-track segments with crossing tracks, simultaneous meet and overtake events at sidings, and some features unique to this particular network section.

The dispatch optimization and data reconciliation problems share the same parameters, decision variables, and constraint set for a given network topology. Here a specific form of the MILP is used that is based primarily on the dispatch formulation of Petersen et al. (1986) and Higgins et al. (1996), but in principle the data reconciliation problem can be posed using constraints from other optimization-based dispatching problems.

4.1 Problem Setup

A track graph for the network section over which trains operate is delineated by OS-points that are located at the endpoints of multi-track segments or siding tracks (as discussed in Section 1.2). The set of all tracks segments is denoted M, with individual segments assigned integer labels beginning with track zero such that $M : 0, 1, 2, 3, ...$ Track segments containing a siding track or multiple tracks are included in the set S, where $S \subset M$. Each track segment $m \in M$ has length K_m. Trains travel in two directions on the network: direction 1 and direction 2. Define direction 1 to be the direction of increasing track integer labels and direction 2 to be the decreasing direction. Because track segments are denoted by integers, we can refer to the successor track in direction 1 relative to a segment $m \in M$ as segment $m + 1 \in M$. Likewise, the successor track in direction 2 relative to segment m is $m - 1$.

The set of trains traveling in direction 1 is denoted I and direction 2 trains are J. Individual trains are referred to as $i \in I$ or $j \in J$ and have unique identifiers such that $I \cap J = \emptyset$. Each train has a known length denoted L_i (or L_j).

An example network section is shown in Figure 2. This section has five track segments, $M : \{0, 1, 2, 3, 4\}$, two of which contain siding tracks, $S : \{1, 3\}$. The length of each track segment is labeled K_0, K_1, etc. Two trains are also shown: train $i \in I$ in direction 1 and train $j \in J$ in direction 2.

Additional parameters used in the objective function and in constraints must be provided. Historical data, as discussed in Section 3.2, is denoted $\tilde{x}$. Specifically, we define the historical completion time of each train i (and j) for each track segment m to be $\tilde{x}_{i,m}$ (and $\tilde{x}_{j,m}$). Note that completion time of a segment is relative to direction, so the values $\tilde{x}_{i,m}$ and $\tilde{x}_{j,m}$ for the same segment m refer to different endpoints of the track segment.

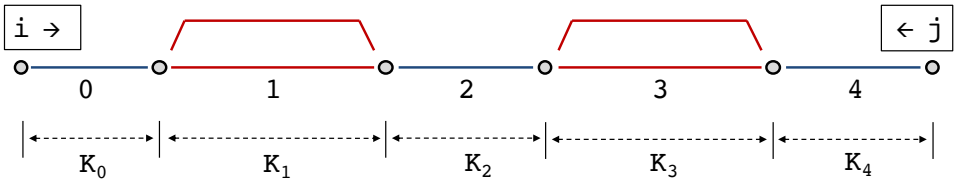


Figure 2: Depiction of notation used in data reconciliation problem for an example track graph with 5 segments. The set of all track segments in this example is $M : \{0, 1, 2, 3, 4\}$ and the set of siding segments is $S : \{1, 3\}$. The length of each track segment is denoted K_0, K_1, etc. The two trains in this example are labeled $i \in I$, which travels in direction 1, and $j \in J$, which travels in direction 2.

The free run (i.e., minimum) traversal time of each track segment is defined specific to each train. If train i takes the main line track on a segment m, its free run traversal time of the segment is $T_{i,m}$. If train i takes the siding track on a segment $s \in S$, its free run traversal time across the siding track is $U_{i,s}$. We assume that the siding free run time values are greater than the corresponding main line free run time (i.e., $U_{i,s} \geq T_{i,s}$). Trains $j \in J$ have corresponding parameters $T_{j,m}$ and $U_{j,s}$.

For meet or overtake events between pairs of trains, we define minimum clearance headways in terms of time (minutes). The minimum headway between trains traveling in the same direction is H_{i_1,i_2} (or H_{j_1,j_2}) for pairs of trains in $i_1, i_2 \in I$ (or $j_1, j_2 \in J$). For trains traveling in opposite directions, the headway time is $H_{i,j}$, where $i \in I$ and $j \in J$.

4.2 Decision Variables

The real-valued decision variables for the reconciliation problem are the reconciled trajectory timing values. The decision variables representing the reconciled data are denoted $x_{i,m}$ and $x_{j,m}$ for trains $i \in I$ and $j \in J$, respectively, corresponding to each track segment $m \in M$. These correspond to the historical data $\tilde{x}_{i,m}$ and $\tilde{x}_{j,m}$.

The integer-valued decision variables govern the interactions between trains. We use variables indicating train ordering (i.e., the order in which trains complete a track segment) to identify meet and overtake events. Let the set of track segments that are only single-track segments be denoted $M \setminus S$, which is the set M minus the set S. We define the ordering variables $\pi_{i,j,m}$ for all combinations of trains $i \in I$, trains $j \in J$, and track segments $m \in (M \setminus S)$, to be $\pi_{i,j,m} = 1$ if train i crosses segment m before train j, and $\pi_{i,j,m} = 0$ otherwise. For trains traveling in the same direction, we define $\phi_{i_1,i_2,m} = 1$ to indicate that train $i_1 \in I$ completed traversal of segment m before train $i_2 \in I$ (where $i_1 \neq i_2$), and $\phi_{i_1,i_2,m} = 0$ otherwise. Likewise, $\phi_{j_1,j_2,m} = 1$ if train $j_1 \in J$ completed traversal of m before train $j_2 \in J$, where $j_1 \neq j_2$.

The occurrences of meet events are indicated by binary values of $\mu_{i,j,s}$, which take the value $\mu_{i,j,s} = 1$ if a meet occurs between trains $i \in I$ and $j \in J$ along track segment $s \in S$, and $\mu_{i,j,s} = 0$ otherwise. The occurrence of overtake events for trains I in direction 1 are indicated by binary values of $\rho_{i_1,i_2,s}$, which takes the value $\rho_{i_1,i_2,s} = 1$ if a meet occurs between trains $i_1 \in I$ and $i_2 \in I$ (where $i_1 \neq i_2$) along track segment $s \in S$, and $\rho_{i_1,i_2,s} = 0$ otherwise. Values of $\rho_{j_1,j_2,s}$ encode overtakes for trains $j_1, j_2 \in J$ in direction

2.

When meet and overtake events occur, one of the trains in each event must take the siding track and one must take the main line track. Let $\sigma_{i,s} = 1$ if train $i \in I$ used a siding track at track segment $s \in S$, and $\sigma_{i,s} = 0$ if it did not. Likewise, let $\sigma_{j,s} = 1$ if train $j \in J$ used a siding track at $s \in S$, and $\sigma_{j,s} = 0$ if it did not.

4.3 Objective Function

The specific data reconciliation objective used in this work is as follows. We apply an $\mathcal{L}_1$ norm on the deviations from the historical data when the historical data is present, and regularize with an $\mathcal{L}_1$ penalty to a background term that encourages trains to travel at a constant speed in all sections for which data is missing. This is written as:

$$||x_\Omega - \tilde{x}_\Omega||_1 + ||x_\Psi - x^{\text{des}}||_1, \tag{7}$$

where x_Ω corresponds to a vector containing entries of $x_{i,m}$ and $x_{j,m}$ for all $i \in I$, $j \in J$, and $m \in M$ for which historical data is available. The vector x^{des} is arranged to have entries corresponding to the elements of x for which no historical data is available, and is set assuming trains in the historical dataset travel at constant speeds through sections with missing data, independent of other trains or physical constraints. Note that x^{des} may or may not be feasible, and is only used as a regularization term.

4.4 Constraints

Travel Time Constraints

Train timing at each OS-point is governed by prior OS-point timing data and minimum free run times. Precisely, the completion time $x_{i,m}$ for train i of segment m must be greater than or equal to the completion time $x_{i,m-1}$ of the preceding segment plus the minimum free run travel time $T_{i,m}$ specific to that train and segment. This is written as:

$$x_{i,m} \geq x_{i,m-1} + T_{i,m}, \tag{8}$$

where $i \in I$ and $m \in M$. For trains j traveling in direction 2, we have:

$$x_{j,m} \geq x_{j,m+1} + T_{j,m}, \tag{9}$$

where $j \in J$ and $m \in M$. Note that the segment preceding segment m is $m+1$ for direction 2, because the track segments labels are numbered in increasing order in direction 1.

When train i uses siding s (i.e., $\sigma_{i,s} = 1$), the completion time $x_{i,s}$ of the track segment s depends on the completion time of the previous segment $x_{i,s-1}$ and the minimum siding travel time $U_{i,s}$:

$$\texttt{IF } \sigma_{i,s} = 1, \texttt{ THEN } x_{i,s} \geq x_{i,s-1} + U_{i,s}, \tag{10}$$

where $i \in I$ and $s \in S \subset M$. Recall based on the numbering of the track segments, $s - 1 \in M$ refers to the track segment immediately before s and that the siding travel time $U_{i,s} \geq T_{i,s}$, indicating the minimum siding travel time is longer than the minimum main line travel time for each train at each segment.

A similar constraint on the completion time when trains $j \in J$ take the siding track handles trains in the opposite direction:

$$\texttt{IF } \sigma_{j,s} = 1, \texttt{ THEN } x_{j,s} \geq x_{j,s+1} + U_{j,s}. \tag{11}$$

Meet and Overtake Constraints

Meet and overtake events are constrained using logical properties of the binary ordering variables π and ϕ.

We constrain the arrival times of opposite direction trains at siding endpoints such that a train may not enter onto a single-track segment until the train in the opposite direction has cleared off the single-track segment, plus a safety headway. Recalling that $\pi_{i,j,m}$ indicates which train (i or j) first traverses a single-track segment $m \in (M \setminus S)$, and takes the value $\pi_{i,j,m} = 1$ if train i traverses first and 0 otherwise. Then the meet constraint is written as:

$$\texttt{IF } \pi_{i,j,m} = 1, \texttt{ THEN } x_{i,m} + H_{i,j} \leq x_{j,m+1}, \texttt{ ELSE } x_{j,m} + H_{i,j} \leq x_{i,m-1}, \tag{12}$$

where $m \in (M \setminus S)$, $i \in I$, and $j \in J$. Constraint (12) activates based on the value of $\pi_{i,j,m}$ and applies only to single track segments. If $\pi_{i,j,m} = 1$, then train i is arriving at the end of the single track segment before train j, and must have at least $H_{i,j}$ minutes of safety headway before train j proceeds onto the single-track segment. Note that because of directionality, the timing variable $x_{i,m}$ refers to the completion time of the single-track segment by train i and $x_{j,m+1}$ refers to the entry time of train j onto the same single-track segment. In the case that j traverses the single-track segment before train i ($\pi_{i,j,m} = 0$), then we require train j to finish the single-track segment, plus the safety headway, before train i may finish segment $m-1$ and enter onto the single-track segment m. Note that in the case that train j traverses m first, the constraint refers to the opposite end of the single-track segment where the completion time of train j is $x_{j,m}$ and the entry time of train i is $x_{i,m-1}$.

In the case of same-direction trains, we impose a following-headway to the completion times of each track segment depending on which train completes the segment first. Recall that $\phi_{i_1,i_2,m} = 1$ if train $i_1 \in I$ traverses segment $m \in M$ before train $i_2 \in I$, where $i_1 \neq i_2$ (i.e., train i_2 follows train i_1). In this case, the completion time $x_{i_2,m}$ of the segment for train i_2 must be at least H_{i_1,i_2} minutes (the safety headway) after the completion time $x_{i_1,m}$ of train i_1:

$$\texttt{IF } \phi_{i_1,i_2,m} = 1, \texttt{ THEN } x_{i_1,m} + H_{i_1,i_2} \leq x_{i_2,m} \tag{13}$$

A similar constraint handles the headway separation of trains traveling in direction 2:

$$\texttt{IF } \phi_{j_1,j_2,m} = 1, \texttt{ THEN } x_{j_1,m} + H_{j_1,j_2} \leq x_{j_2,m} \tag{14}$$

The next set of constraints allows overtakes only on siding segments, by forcing the order of same-direction trains to stay the same on single-track segments. For direction 1:

$$\phi_{i_1,i_2,m} = \phi_{i_1,i_2,m-1}, \tag{15}$$

where $m \in (M \setminus S)$, and direction 2:

$$\phi_{j_1,j_2,m} = \phi_{j_1,j_2,m+1}, \tag{16}$$

where $m \in (M \setminus S)$.

In a single-track network topology with a high volume of traffic, simultaneous meet and overtake events occurring at sidings with more than two parallel tracks do occur, albeit rarely. For example, if a train $i_1 \in I$ is overtaken by train $i_2 \in (I \setminus \{i_1\})$ and both i_1 and i_2 meet train $j \in J$, then three parallel tracks are required. To simplify the presentation, here we only describe the constraints that consider the case of two parallel tracks. Extensions to there or more parallel tracks result in additional meet and pass constraints that are tedious but also result in mixed integer constraints.

Recall that meet events are identified by $\mu_{i,j,s} = 1$ if a meet occurs between trains i and j at siding segment s, and $\mu_{i,j,s} = 0$ otherwise. Overtake events are identified by $\rho_{i_1,i_2,s} = 1$ if an overtake occurred between trains i_1 and i_2, and $\rho_{i_1,i_2,s} = 0$ otherwise. Consider train i_1 at track segment s. The total number of meet events train i_1 experiences with any opposite direction trains in J at s is $\sum_{j \in J} \mu_{i_1,j,s}$. Similarly, the total number of overtakes that train i_1 experiences with any same direction trains $i_2 \in (I \setminus \{i_1\})$ is $\sum_{i_2 \in (I \setminus \{i_1\})} \rho_{i_1,i_2,s}$. To avoid simultaneous meet and/or overtake events occurring on segment s, we would require:

$$\sum_{j \in J} \mu_{i_1,j,s} + \sum_{i_2 \in (I \setminus \{i_1\})} \rho_{i_1,i_2,s} \leq 1 \tag{17}$$

where $i_1 \in I$ and $s \in S$.

Likewise, for a train j_1 traveling in direction 2, we have an analogous constraint:

$$\sum_{i \in I} \mu_{i,j_1,s} + \sum_{j_2 \in (J \setminus \{j_1\})} \rho_{j_1,j_2,s} \leq 1 \tag{18}$$

with $j_1 \in J$ and $s \in S$.

Siding Assignment Constraints

For each meet event and overtake event that occurs, one of the trains must be assigned to take the siding track, which in turn imposes the minimum siding travel time constraint. These constraints are activated by the values of μ and ρ that indicate the occurrence of meets and overtakes, respectively.

Recall that the siding track indicator variable $\sigma_{i,s}$ takes the value $\sigma_{i,s} = 1$ if train i takes the siding track on segment s, and $\sigma_{i,s} = 0$ otherwise. The same is true for train j and the $\sigma_{j,s}$ variables. When $\mu_{i,j,s} = 1$, a meet occurs between trains i and j at siding segment s. As a result, one and only one of the siding indicator variables $\sigma_{i,s}, \sigma_{j,s}$ must be 1. This is written as:

$$\texttt{IF } \mu_{i,j,s} = 1, \texttt{ THEN } \sigma_{i,s} + \sigma_{j,s} = 1, \tag{19}$$

where $i \in I$, $j \in J$ and $s \in S$.

Likewise, for overtakes occurring in direction 1 between trains $i_1, i_2 \in I$ on siding $s \in S$ (indicated by the value $\rho_{i_1,i_2,s} = 1$), we have:

$$\texttt{IF } \rho_{i_1,i_2,s} = 1, \texttt{ THEN } \sigma_{i_1,s} + \sigma_{i_2,s} = 1. \tag{20}$$

A similar constraint holds for overtaking trains in direction 2:

$$\texttt{IF } \rho_{j_1,j_2,s} = 1, \texttt{ THEN } \sigma_{j_1,s} + \sigma_{j_2,s} = 1, \tag{21}$$

where $j_1, j_2 \in J$ and $s \in S$.

Finally, any trains using siding tracks must be short enough to physically fit on the available track length without interfering with switch points at the end of the siding track. If the length L_i of a train $i \in I$ is greater than the length K_s of a siding segment $s \in S$, then train i must not be assigned to take siding s (i.e., $\sigma_{i,s} = 0$). This is written as:

$$\texttt{IF } L_i > K_s, \texttt{ THEN } \sigma_{i,s} = 0, \tag{22}$$

with a similar constraint holding for trains $j \in J$:

$$\texttt{IF } L_j > K_s, \texttt{ THEN } \sigma_{j,s} = 0. \tag{23}$$

We note that variables identifying meets μ and overtakes ρ are set by additional constraints using logic derived from timing variables, which we do not enumerate here. Similar sets of constraints are also used to encode the IF/THEN/ELSE logic used to simplify the presentation of the constraints. The complete problem formulation results in a mixed integer optimization problem and does not require the use of a constraint programming solver.

5 Data Reconciliation Case Study on US Class-1 Freight Rail Data

In this section, the data reconciliation problem from Section 4 is run on data from a portion of a US class-1 freight railroad network. First, a description of the historical dataset and computational environment on which the data reconciliation problem is implemented are described. Two sets of experiments are run to assess the quality of the data reconciliation approach. In the first experiments, the data reconciliation problem is applied to a data which is complete but contains errors, for example due to upstream data cleaning steps to impute missing values. In the second set of experiments a synthetic dataset is created from the real original dataset by decimating entries of the complete dataset. Since the true entries are known, it allows assessment of the quality of the imputed solutions from the data reconciliation problem.

5.1 Description of Historical Dataset

The experiments described in this section use a real historical dispatch dataset from a section of the CSX Transportation rail network in the eastern United States between Nashville, TN, and Chattanooga, TN, described also in Barbour et al. (2018a,b). The time period used is six months between January 1, 2016, and June 30, 2016. The dataset contains 4368 hours of data and it includes more than 3,000 individual train trajectories. This section of the network is approximately 100 miles in length (160 km) and is highlighted by the yellow dashed box in the map in Figure 3. The test corridor is predominantly single track (blue sections on the map) with 11 passing sidings (red sections with dashed line delineations) of varying length. It is a highly congested area of the CSX network and trains must also contend with significant grade at multiple locations caused by mountains. The topology of the network combined with the high volume of traffic result in many meet and overtake events.

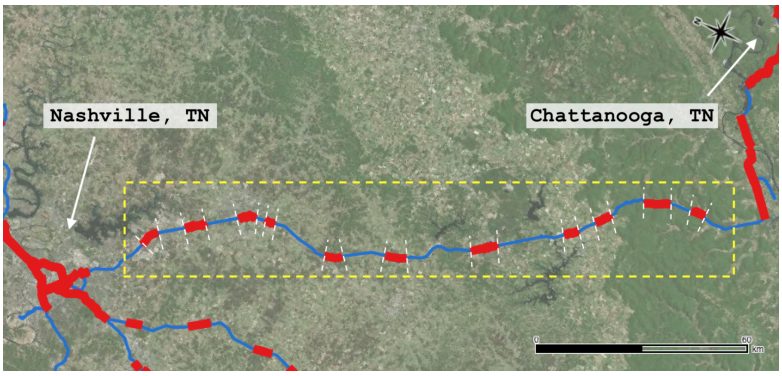


Figure 3: Map of the rail network territory is shown in the yellow dashed box between Nashville, TN, to Chattanooga, TN, United States. Multi-track sections are shown in red and single-track sections are shown in blue. The scale bar represents 60 kilometers.

5.2 Computational Environment

The data reconciliation problem is written in the AMPL mathematical programming language and solved using CPLEX 12, a commercial MILP solver. The model is connected to Python code that loads and transforms data, extracts results, and analyzes the output.

In order to maintain a reasonable size of MILP for the reconciliation problem, the data reconciliation problem is solved for datasets in a sliding window with a length between 8 and 24 hours (exact values explained in Section 5.4). A single 24 hour dataset containing approximately 20 trains yields a MILP of approximately 5,000 variables and 20,000 constraints, of which approximately 4,000 variables are binary and approximately 15,000 constraints encode logical constraints between the binary variables.

5.3 Experiment 1: Reconciliation of a Complete but Erroneous Historical Dataset

The first set of experiments are conducted on the six-month long historical dataset, which is complete, but contains errors. Any missing data points are imputed in upstream data cleaning steps which may or may not result in feasible trajectories. Using this dataset we apply the data reconciliation problem to identify and automatically correct erroneous data that do not satisfy operational constraints. The complete dataset is analyzed in a 12-hour shifting window until all data has been reconciled.

The results are as follows. On average, each 12-hour window of raw historical data contains approximately three errors that are corrected by the data reconciliation problem. Due to the proprietary and sensitive nature of the historical dataset, detailed descriptions and analysis of the errors (e.g., statistics on the types and the frequency at which they occur) specific to this dataset are not discussed in depth here. To qualitatively assess the quality of the reconciled data, after application of the data reconciliation problem, one week of the historical and reconciled data is manually inspected. The manual inspection verified that the reconciled data only deviates from the historical data in places where the historical data led to constraint violation. Common errors based on the manual inspection include infeasible meets and passes and headway constraint violations due to errors in the timing data.

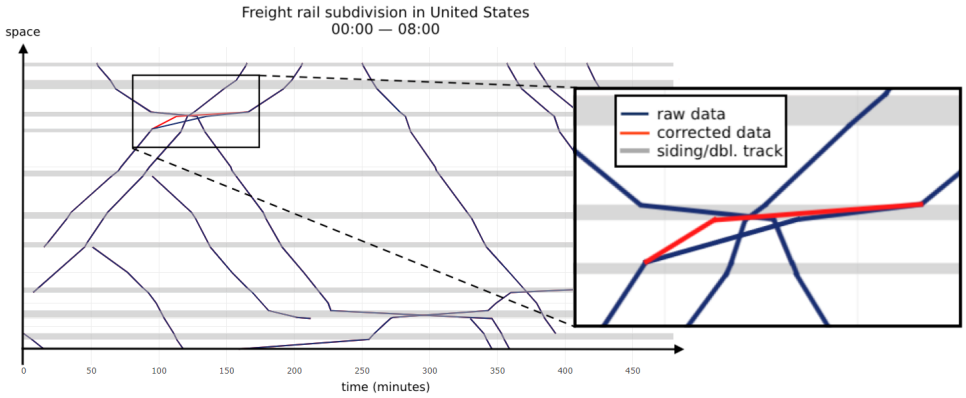


Figure 4: Stringline diagram of historical and reconciled data. Sidings and multi-track segments are shown as grey shaded areas. Raw train trajectory data is shown as blue lines. The raw data indicates that two meet and overtake events, magnified in the figure inset, occur on a single-track segment, which is infeasible. The red line is the reconciled data that results in feasible trajectories.

To give an insight into the type of errors that are automatically corrected, a representative example of an error in the historical data is shown in Figure 4 (The first eight hours of the 12 hour window are shown). Sidings and multi-track segments, where trains may pass each other, are denoted as grey shaded areas, with the white areas denoting single-track segments. The historical train trajectories (blue lines) have impermissible meet/overtake events that are magnified in the figure inset. The errors are evident in the stringline diagram because the expected meet and overtake events (i.e., the intersection point between trajectories) occur on a single track segment. In contrast, the data reconciliation problem produces the same trajectories as the historical dataset everywhere except in the neighborhood of the infeasible meet/overtake events. In that area, the reconciled data is indicated by the red line, and it results in a set feasible trajectories for all trains. There are three tracks at this passing siding, allowing both a meet and an overtake event to occur simultaneously. Note in Figure 4 that trajectories that do not cover the entire space correspond to local trains that complete routes between small intermediate destinations on this section of the network.

5.4 Experiment 2: Reconciliation of a Synthetically Decimated Historical Dataset

Next we quantitatively assess the performance of the data reconciliation problem when imputing missing data with feasible values. We begin with the historical data and create a dataset with missing entries by decimating (removing) a subset of the data entries. This is done to allow comparison between the imputed values produced by the data reconciliation problem with the true historical values that are known (but decimated in the data given to the data reconciliation problem).

To aid in interpretability of the results, the data is decimated only in areas far from any infeasible portions of the historical data, i.e., the historical data that is decimated is feasible. We clarify this is not a limitation of the method (i.e., it can be applied to a dataset containing both missing and erroneous data), but that it is not trivial to assess if differences between

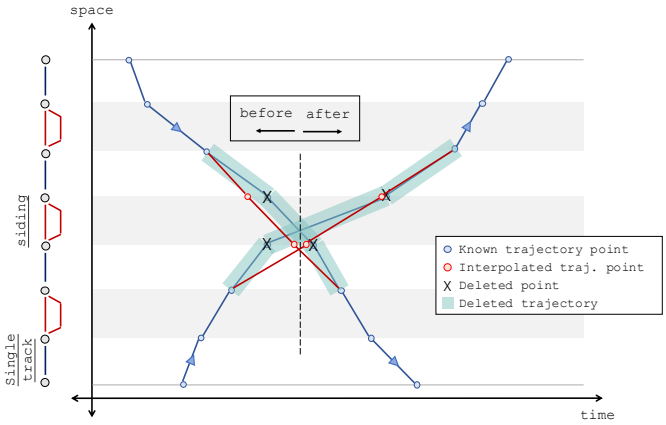


Figure 5: Four known trajectory points are selectively removed around meet/overtake events that are identified in historical data. One point immediately before the event and one point immediately after the event are removed for each train. These deleted points are shown as black 'X' markers and the missing trajectory segments are highlighted in light blue. A linear interpolation to impute the missing data (red points and lines) can result in infeasible meets.

the imputed and historical data are due to infeasibility of the historical data, or due to a poor imputed result from the data reconciliation problem. In the experiments conducted next, the synthetically decimated data is feasible so the ambiguity is avoided.

Generation of Synthetically Decimated Historical Datasets

The synthetically decimated historical dataset is created by removing known trajectory points around meet and overtake events in the reconciled historical data. At each of these events, a particular number of data points (per train) immediately before and immediately after the meet or overtake event are removed. This results in missing data centered around known meet and overtake events. Figure 5 shows an illustration of this removal process for a meet event between two trains. One point before the meet event in each trajectory and one point after the event in each trajectory are removed.

We assess and compare the quality of the imputed data from data reconciliation with imputed data from a naive linear interpolation approach. In Figure 5, the red lines and points represent the values imputed via linear interpolation. The interpolation uses the nearest known trajectory points to calculate the average speed across the missing trajectory section, from which the missing points are interpolated. There are many methods more complex than linear interpolation to which data reconciliation could be compared – speed-regularized interpolation, delay minimization, and energy conservation, to name a few – but linear interpolation is used here as a straightforward baseline method.

The quality of the imputed trajectory points is assessed by *i*) evaluating the location at which the recovered trajectories estimate meet and overtake events to occur and *ii*) calculating the time difference between each imputed value and the known trajectory value.

The location of each meet or overtake event found in the reconciled data is *feasible* if and only if it is on a siding or multi-track segment and does not violate other constraints. This location is *correct* if it matches the true location of the event indicated by the known data.

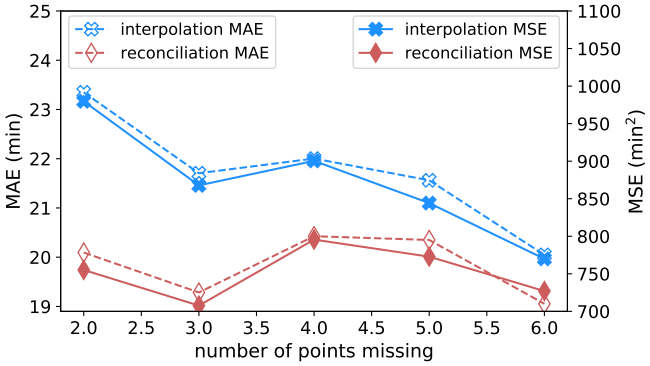


Figure 6: Mean absolute error and mean squared error of timing values for each missing point imputed by interpolated and reconciliation. MAE and MSE are averaged across trials, grouped by the total number of missing points around each meet or overtake event.

Note it is possible for linear interpolation to produce feasible or infeasible, and correct or incorrect imputed values. In contrast, data reconciliation always produces feasible imputed values which may or may not be at the correct location.

The quality of the timing data is assessed via the *mean absolute error* (MAE) and *mean squared error* (MSE) of the imputed values compared to the historical values that are decimated. Letting x^*_Ψ denote the vector of reconciled values, and with a slight abuse of notation, let $\tilde{x}_\Psi$ denote the historical data which is known but synthetically decimated in the experiment (i.e., the assumed ground truth). The quality of the imputed values are:

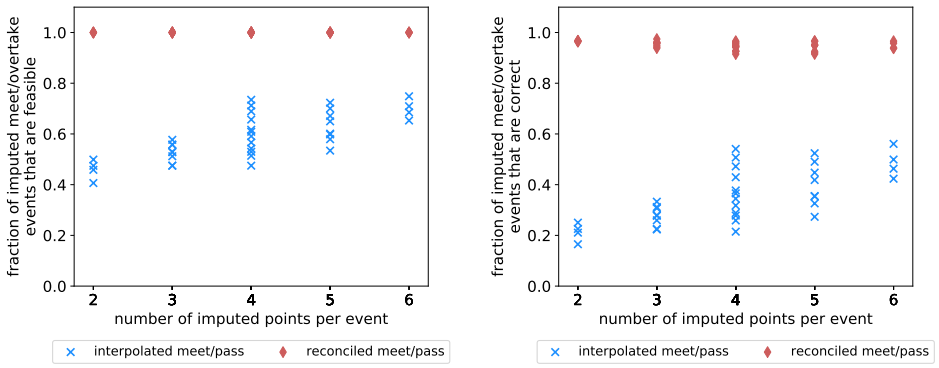
$$\text{MSE} = \frac{1}{|\tilde{x}_\Psi|}||\tilde{x}_\Psi - x^*_\Psi||_2^2, \qquad \text{MAE} = \frac{1}{|\tilde{x}_\Psi|}||\tilde{x}_\Psi - x^*_\Psi||_1, \tag{24}$$

where $|\tilde{x}_\Psi|$ denotes the number of imputed values.

Results on Synthetically Decimated Datasets

The results of the data reconciliation experiments on the synthetic, incomplete dataset are presented next. A total of 45 data reconciliation experiments are conducted on the six month dataset. Each experiment is defined by *i*) the number of points per train that are removed immediately before a meet or overtake event, *ii*) the number of points per train that are removed immediately after a meet or overtake event, and *iii*) the length of the sliding window. For example, the first experiment removes a single point per train before and a single point per train after each meet/overtake event, and the data reconciliation problem is solved on a sliding eight hour window through the six month dataset. The remaining experiments are defined by considering: *i*) the number of missing points per train immediately before a meet/overtake event (1, 2, or 3 points), *ii*) number of missing points per train after an event (1, 2, or 3 points), and *iii*) the sliding window length (8, 12, 16, 20, or 24 hours).

The MAE and MSE for trajectory points imputed by data reconciliation and linear interpolation are shown in Figure 6. The results are grouped by the number of total missing points around each meet or overtake event (i.e., the total points immediately before and after each event, resulting in between two to six missing points). Data reconciliation results in a 5-15% reduction in both MAE and MSE compared to linear interpolation.



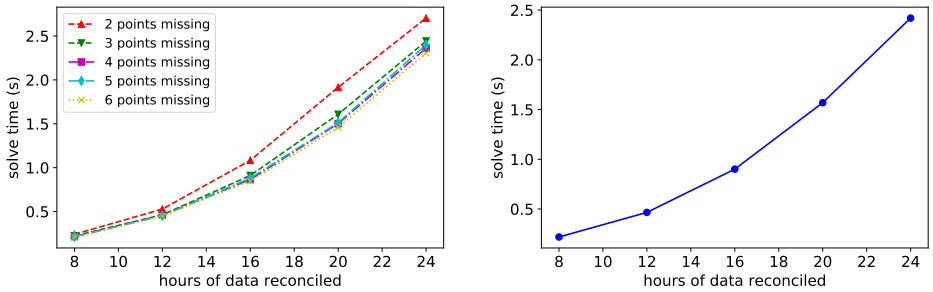
(a) (b)

Figure 7: Fraction of meet/overtake events found by data interpolation and reconciliation that are (a) found to occur at a feasible location, and (b) at the correct location.

The fraction of meet and overtake events that are found to occur at a feasible location when imputed by data reconciliation and by linear interpolation are shown in Figure 7a. The trial results are grouped by total number of points missing (i.e. points immediately before and after an event). The multiple values for a given number of points correspond to the various experiments run with differing points missing before or after each event but resulting in the same number of total missing points per event. Because the data reconciliation problem uses the physical constraints when interpolating the points, 100% of the imputed meet/overtake events are feasible. In contrast, linear interpolation results in feasible meet and overtake locations in only 40-70% of cases and exhibits variability across the different experiments for the same number of total missing points.

Figure 7b shows the fraction of meet and overtake events that are estimated to occur at the correct location as indicated by the known data, grouped by the number of missing points around each event. Reconciliation recovers the correct location for meet and overtake events in approximately 95% of cases, while linear interpolation recovers only 20-50%. Additionally, reconciliation preforms consistently across trials, with interpolation demonstrating higher variability in performance.

The reconciliation problem executes very quickly, even on large amounts of data. Solve time increases non-linearly as a function of the number of hours used for the shifting window, as seen in Figure 8b. The number of missing points per overtake event does not have a substantive effect on the solve time, as shown in Figure 8a. The solve times for two and three missing points per meet/overtake event are slightly longer than trials with larger numbers of missing points, but follow a similar trend to the larger numbers of missing points. Solve time of the reconciliation model is low due to the fact that the majority of constraints are already satisfied and the number of corrections required between historical and reconciled data is low. Based on the solve time for the reconciliation model, a year of data from a large rail network (e.g., track networks of freight railroad companies in the United States) could be reconciled in about 20 hours of total CPU time with a 24 hour sliding window.



(a) Average solve time by amount of missing data at each meet and pass event.

(b) Average solve time across all trials and amounts of missing data.

Figure 8: Solve time of data reconciliation model by length of shifting data window.

6 Conclusion

Given the growing emphasis on data driven analysis and algorithms to improve operational efficiency, tools are needed to automate the cumbersome data cleaning process. This work introduced the data reconciliation problem as a tool to correct errors and impute missing values in operational rail datasets. The data reconciliation problem leverages operational constraints that are commonly used in dispatch optimization in a new context that enables efficient reconciliation of infeasible historical data. To demonstrate the viability of the method, the data reconciliation problem is instantiated and applied to a real six-month dataset containing several thousand trains on a complex portion of a US Class-1 rail network. The data reconciliation problem is found to identify and correct erroneous data, as well as impute missing data in a way that is always feasible and often correct.

Numerous extensions to the data reconciliation problem are possible. For example, a detailed design and comparison of different performance measures in the data reconciliation problem objective function might lead to improved accuracy of the reconciled data. It will also be interesting to investigate the sensitivity of the data reconciliation problem to different constraint formulations. In addition to the optimization model discussed in this work, we also intend to test the data reconciliation model on an optimization-based dispatching formulation for multi-track network topologies. Finally, we note that the data reconciliation problem posed here does not identify inefficient but operationally feasible errors. Extensions to identify these errors would be a valuable addition to the rail data cleaning toolbox.

Acknowledgments

The authors acknowledge support by the Roadway Safety Institute, the University Transportation Center for USDOT Region 5, which Includes Minnesota, Illinois, Indiana, Michigan, Ohio, and Wisconsin. Financial support was provided by the United States Department of Transportation's Office of the Assistant Secretary for Research and Technology (OST-R) and by the Federal Highway Administration's Office of Innovative Program Delivery (OIPD), via the Dwight David Eisenhower Transportation Fellowship Program.

References

Barbour, W., Mori, J. C. M., Kuppa, S., and Work, D. B. (2018a). Prediction of arrival times of freight traffic on us railroads using support vector regression. *Transportation Research Part C: Emerging Technologies*, 93:211–227.

Barbour, W., Samal, C., Kuppa, S., Dubey, A., and Work, D. B. (2018b). On the data-driven prediction of arrival times for freight trains on us railroads. In *Proceedings of the IEEE 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2289–2296.

Bollapragada, S., Markley, R., Morgan, H., Telatar, E., Wills, S., Samuels, M., Bieringer, J., Garbiras, M., Orrigo, G., Ehlers, F., Turnipseed, C., and Brantley, J. (2018). A novel movement planner system for dispatching trains. *Interfaces*, 48(1):57–69.

Chapuis, X. (2017). Arrival time prediction using neural networks. In *Proceedings of RailLille2017: 7th International Conference on Railway Operations Modelling and Analysis*, pages 1500–1510. International Association of Railway Operations Research (IAROR).

Claudel, C. G. and Bayen, A. M. (2011). Convex formulations of data assimilation problems for a class of Hamilton–Jacobi equations. *SIAM Journal on Control and Optimization*, 49(2):383–402.

Fang, W., Yang, S., and Yao, X. (2015). A survey on problem models and solution approaches to rescheduling in railway networks. *Transactions on Intelligent Transportation Systems*, 16(6):2997–3016.

Gestrelius, S., Aronsson, M., and Peterson, A. (2017). A MILP-based heuristic for a commercial train timetabling problem. *Transportation Research Procedia*, 27:569–576.

Ghofrani, F., He, Q., Goverde, R. M., and Liu, X. (2018). Recent applications of big data analytics in railway transportation systems: A survey. *Transportation Research Part C: Emerging Technologies*, 90:226–246.

Higgins, A., Kozan, E., and Ferreira, L. (1996). Optimal scheduling of trains on a single line track. *Transportation Research Part B: Methodological*, 30(2):147–161.

Kecman, P. and Goverde, R. M. (2015). Online data-driven adaptive prediction of train event times. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):465–474.

Khoshniyat, F. and Peterson, A. (2015). Robustness improvements in a train timetable with travel time dependent minimum headways. In *Proceedings of the 6th International Conference on Railway Operations Modelling and Analysis (RailTokyo2015)*.

Leibman, M., Edgar, T., and Lasdon, L. (1992). Efficient data reconciliation and estimation for dynamic processes using nonlinear programming techniques. *Computers & Chemical Engineering*, 16(10-11):963–986.

Murali, P., Ordóñez, F., and Dessouky, M. M. (2016). Modeling strategies for effectively routing freight trains through complex networks. *Transportation Research Part C: Emerging Technologies*, 70:197–213.

Oneto, L., Buselli, I., Lulli, A., Canepa, R., Petralli, S., and Anguita, D. (2019). A dynamic, interpretable, and robust hybrid data analytics system for train movements in large-scale railway networks. *International Journal of Data Science and Analytics*, pages 1–17.

Petersen, E., Taylor, A., and Martland, C. (1986). An introduction to computer-assisted train dispatch. *Journal of Advanced Transportation*, 20(1):63–72.

Soderstrom, T. A., Himmelblau, D. M., and Edgar, T. F. (2001). A mixed integer optimization approach for simultaneous data reconciliation and identification of measurement bias. *Control Engineering Practice*, 9(8):869–876.

Tjoa, I.-B. and Biegler, L. (1991). Simultaneous strategies for data reconciliation and gross error detection of nonlinear systems. *Computers & Chemical Engineering*, 15(10):679–690.

Tong, H. and Crowe, C. M. (1995). Detection of gross erros in data reconciliation by principal component analysis. *AIChE Journal*, 41(7):1712–1722.

Törnquist, J. (2006). Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms. In *Proceedings of the 5th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS'05)*, volume 2.

Törnquist, J. and Persson, J. A. (2007). N-tracked railway traffic re-scheduling during disturbances. *Transportation Research Part B: Methodological*, 41(3):342–362.

Wang, P. and Goverde, R. M. (2016). Train trajectory optimization of opposite trains on single-track railway lines. In *Proceedings of the International Conference on Intelligent Rail Transportation (ICIRT)*, pages 23–31.

Wang, R. and Work, D. B. (2015). Data driven approaches for passenger train delay estimation. In *Proceedings of the IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*, pages 535–540.

Zhao, M., Garrick, N., and Achenie, L. (1998). Data reconciliation—based traffic count analysis system. *Transportation Research Record: Journal of the Transportation Research Board*, 1625:12–17.