MODELS AND ALGORITHMS FOR DYNAMIC REAL-TIME FREIGHT TRAIN RE-ROUTING

A Thesis Defense
Presented to
The Academic Faculty

by

Alborz Parcham-Kashani

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology
December 2018

MODELS AND ALGORITHMS FOR DYNAMIC REAL-TIME FREIGHT TRAIN RE-ROUTING

Approved by:

Professor Alan L. Erera, Advisor
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Professor David M. Goldsman
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Professor John H. Vande Vate
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Doctor David Ramcharan
General Director, Decision Technologies and Real-time Advanced Analytics
Creighton University

Professor Joel Sokol
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Approved: October 12, 2018

This dissertation is dedicated to my father.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Alan Erera, for his advice and guidance throughout my graduate studies. Dr. Erera sets a high bar for what it means to be a seasoned and self-sufficient researcher, and I have learned from his great example. I am grateful to my dissertation committee: Dr. David Goldsman, Dr. John Vande Vate, Dr. Joel Sokol, and Dr. David Ramcharan. Their constructive engagement with me over the course of completing this dissertation is appreciated, and I feel privileged to have worked with and learned from seasoned professors of their caliber.

My heart-felt thanks goes out to Douglas Fuller and Aly Megahed. Over the past few years they have believed in my abilities as a student, a professional, and a researcher. Their guidance has pushed me outside my comfort zone and enabled me to achieve what I set out to achieve.

To the fellow graduate students, I thank you for a fantastic learning environment and friendships I will keep with me forever. You made this chapter of my life a memorable one.

Chi Tran, I thank you for your continuous support when it mattered most. You understood how important this journey was for me, and supported me when I was in need of support.

Ashkan and Ardalan, thank you guys for always being the positive voice in the room. You have always believed in me, and it has given me confidence in myself.

Most importantly, I thank my parents from the bottom of my heart. They have always put my needs ahead of their own, and understood me better than I understand myself. They made this success possible, and I am forever grateful, and indebted.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This dissertation focuses on solving a train re-routing problem in a near real-time context for a freight train carrier operating over a large network. When rail terminal operations of a freight carrier are disrupted, one of the levers available to mitigate the disruption is changing the destination of a select subset of trains. Traditionally, the decision of whether and how to re-route the trains under these conditions has been solved in practice using high-level analysis and business judgement. This is partially due to the real-time nature of the problem, and the need to enact a solution quickly. The challenge with this approach is two-fold: (1) it does not take into account the holistic impact of the real-time decisions being made on the traffic patterns of the rail network; and (2) it does not account for the implications of each train re-routing decision on the cost-benefit dynamic of the remaining re-routing decisions being made.

Note that scientific methods have been developed to solve service network design problems such as train scheduling and railcar-to-block assignment. However, the challenge of applying scientific rigor to the real-time context of the problem explored in this dissertation is whether one can do so while still allowing decisions to be made with the desired speed. In this study we address the challenges discussed above:

1. We develop a solution evaluation framework that allows us to evaluate the holistic impact of any decision on the rail network.
2. We ensure train re-routing decisions are considered jointly.
3. We ensure the methodologies proposed are capable of providing solutions in near real-time.

Chapter *II* develops the holistic solution evaluation framework discussed above using an event-based time-space network model. Key contributions include (1) developing and using this framework to objectively determine the problem instances where the optimal joint decision can be made in real-time using systematic enumeration, and (2) determining the best answer that can be provided in real-time using systematic

enumeration methods. Computational results using data from Union Pacific Railroad, a class I carrier in the United States, are presented to demonstrate solution time and solution quality for a variety of problem instances. The computational results show at least 2000 railcar-hours of delay beyond expected customer arrival times eliminated if the decision must be made within 15 minutes, and at least 5000 railcar-hours of delay eliminated if the decision must be made within one hour.

In Chapter *III* we explore methods to find improved solutions for problem instances where systematic enumeration does not provide the optimal answer in real-time. Key contributions include the development and testing of two solution methodologies: an arc-based multi-commodity network design problem approach and a Tabu search heuristic approach. Both approaches use the time-space network developed in Chapter *II*. Computational results allow a comparison between the two approaches, as well as a comparison between each approach and the baseline approach presented in Chapter *II*. We show that the advantage of the Tabu search heuristic approach is the robustness of the solution time under scenarios where network traffic increases are pressure-tested. By contrast, we show the advantage of the optimization-based approach is relative improvement in solution objective as the number of trains to re-route increases.

In Chapter *IV* we modify the train re-routing problem from the previous two chapters by introducing an additional complexity that is often involved in the operations of a large-scale freight carrier: the impact of rail terminal traffic on rail terminal throughput. Key contributions include (1) the introduction of the problem variant and extending the solution evaluation framework from Chapter *II* to address the problem variant, (2) extending the two solution methodologies from Chapter *III* to solve the modified train re-routing problem, and (3) demonstrating that a linear relationship between rail terminal traffic and rail terminal throughput allows us to preserve the linearity of the optimization-based approach from Chapter *III*. Computational results show that (1) the extended optimization-based approach is solvable in practice, with the exception of a subset of problem instances where increases in network traffic is pressure-tested; and (2) the optimization-based heuristic approach solution objective

shows relative improvement compared to the Tabu search heuristic approach as the number of trains to re-route increases.

Chapter *I*: Introduction to Freight Rail Operations

Founded more than 185 years ago, the freight rail industry is one of the largest industries in the United States, generating over $65.8 billion in revenue in 2016 alone and employing approximately 170,000 employees. Railroads with a 2015 revenue of over $457.9 million are referred to as *Class I railroads*, seven of which exist in the United States as of 2016. Together Class I railroads account for 69% of freight rail mileage, the remainder of which is accounted for by *short-line railroads* and *regional railroads* [25].

The physical network supporting the freight rail industry consists of *rail terminals*, sometimes called *railyards* or *rail stations*. Generally each rail terminal consists of three key internal components [22] [30]:

1. *Arrival yard*, a collection of tracks where incoming trains are gathered and await processing.

2. *Classification section*, where the railcars on the arrival tracks are sorted according to the next train they are to join and sent to the departure tracks where departing trains are formed.

3. *Departure yard*, a collection of tracks where departing trains are formed and await departure.

The terminals are connected via *rail-lines* or *connections*. The railroad operates based on a train schedule, which specifies a set of trains to be run over a specified time period, generally a week. Each train is specified with the following attributes:

- *Train origin*, the rail terminal where the train originates.

- *Train destination*, the rail terminal for which the train is destined.

- *Capacity of the train*, the maximum number of railcars allowed on the train.

- The set of intermediate terminals the train passes through. Although the majority of the railcars on the train generally stay on the train, each intermediate terminal allows the train to *pick up* a number of additional railcars and *set out* a number of its railcars.
- The *departure time* of the train at each of the above terminals except the terminal for which the train is destined.
- The *arrival time* at each of the above terminals except the terminal from which the train originates.
- *Cut-off time* which is always prior to departure time, the point of time after which the train will stop accepting new railcars at train origin.

The three main types of trains operated by Class I railroads are *unit trains*, *intermodal trains*, and *manifest trains*.

Unit trains transport a single commodity. Examples in the United States include coal trains, which deliver 70% of coal delivered to power plants [25]. These trains carry one bulk commodity directly from a *shipper* to a *consignee*. They are through trains, which means they do not have pick-ups or set-outs, and are generally not altered as they travel through intermediate railyards on the way to their final destination.

Intermodal trains use flat railcars, onto which containers or truck trailers are loaded. The name intermodal reflects the fact that the loaded containers also travel via other modes of transportation such as trailer trucks or cargo ships before and after being transported in the rail network. A key advantage of using intermodal trains is the efficiency gained by removing the need to load/unload containers when modes of transportation are changed. Unlike unit trains, intermodals trains perform pick-ups and/or set-outs at consolidation terminals for a transport leg [25].

Manifest trains, the focus of this study, contain mixed freight. These trains contain cargo from multiple customers. Railcars on these trains are introduced into the service network as *shipments*, which in this study we refer to as *commodities*. Each commodity is

a group of railcars sharing the same origin and destination terminal, entering the system together and due at the destination at the same time. Typically each commodity belongs to one particular customer of the railroad. Note that the origin and destination terminals of a commodity do not need to match those of any one particular train, and this is generally the case. Instead, each commodity is associated with a *trip plan*, which is the (inclusive) sequence of rail terminals the commodity is to pass through between the *commodity origin terminal* and the *commodity destination terminal*. The trip plan is analogous to an itinerary in the passenger airline industry. Any railcar in a given commodity is said to have the same trip plan as the commodity.

The movement between each consecutive pair of rail terminals in a commodity's trip plan is referred to as a *leg* of the trip plan. The initial and final legs are typically handled by a local train possibly operated by a regional railroad. The remainder of the legs are usually handled by a Class I railroad, with two notable exceptions:

1. *Short-line shipments*, which can be entirely handled by a regional railroad.

2. The subset of *long-line shipments* whose path cannot be covered by the physical rail network of a single Class I railroad. For instance, commodities traveling from Los Angeles to New York need to pass through the physical rail network of at least two Class I railroads.

The two rail terminals in each leg of the trip plan for a given commodity k generally correspond to the origin and destination of a train in the train schedule. The exception to this rule is if a leg of the trip plan is to be completed via a pickup or setout, which would mean the leg is a subset of the path travelled by the train fulfilling the leg. We shall refer to such trip plan legs as *pick-up/set-out legs*. Note that in order for a trip plan to be feasible for k, the following condition must be true: for each leg of the trip plan, there must exist a train scheduled to travel between the origin and destination of that leg.

Once a train arrives at its final destination terminal, it stops in the arrival yard. The

train is then broken apart in the following manner: the railcars on the train are separated and sorted as they are moved through the classification section of the rail terminal and into the departure yard where departing trains are assembled. The next leg of each commodity's trip plan is a key attribute in determining how commodities are grouped together in the departure yard. Locally destined commodities are sorted into local trains.

The sheer volume of commodities serviced by a Class I railroad creates opportunities to create operational efficiencies by organizing commodities into *blocks*. This is typically done for commodities that share common attributes such as commodity type, commodity origin, and commodity destination. Furthermore commodities which have significant overlap in their trip plans may also be assigned to the same block for the corresponding portion of the trip plans [4].

Let us next discuss the relevant literature in the freight transportation and freight rail industry, followed by an overview of the focus of this thesis.

1.1 Literature Review

There are three broad planning levels in the freight transportation industry: strategic planning, tactical planning, and operational planning. Strategic planning problems have the highest impact on the overall direction of the carrier. These types of problems typically involve long-term decisions and are typically made at the firm level. Examples of strategic planning problems in the rail industry include determining the placement and size of railyards to be built, as well as one-time rolling stock purchases and rail track expansion routes to new regions. In other words, strategic planning problems typically involve the design of the physical network operated by the railroad [11].

An example of a strategic planning problem related to the freight rail industry is the intermodal terminal location problem, which can more generally be thought of as an industry-specific example of the facility location problem. Intermodal terminals are where intermodal trains pick up and drop off standard-size containers from or to a different mode of transporta-

tion, such as road or sea transportation [33]. The problem definition formalized in Arnold et al. [3] starts with a set of candidate intermodal terminals and a set of shipments. There is a fixed cost to opening each intermodal terminal, and the cost of sending each shipment through any pairs of terminals is known, as well as the cost of sending each shipment directly without switching modes of transportation. Arnold et al. [3] proposes a mixed integer programming formulation that minimizes the overall incurred cost, the decision variables of which dictate which facilities to open and how to route the shipments through the induced network.

By contrast, tactical planning problems involve medium-term decisions that define the operating rules of the *service network* [35]. The service network refers to the set of services provided by a carrier and their schedules. Service network design problems normally assume the strategic planning problems related to the design of the physical network have been solved; instead the focus is on building services using the limited fixed and mobile resources comprising the physical network. This class of problems is commonly studied for the rail, airline, and trucking industries, as well as for intermodal services. For instance, in the context of the airline industry service network design can involve determining the set of flights a carrier provides and the schedule for those flights. For Class I railroads the equivalent class of problems involves determining the train routes and the policy of how commodities are blocked together as they are moved through the physical network [20][11].

Finally, operational planning problems, including the one this thesis studies, are typically decided at the mid-management level (e.g. dispatch or railyard operations management teams). Solving this class of problems determines the day-to-day operations of the service network. Service network operational planning problems are thus short-term in scope and dynamic in nature, where time is generally modelled at a highly granular level. Examples of operational planning problems in freight rail include crew scheduling, train scheduling, deadhead scheduling, and scheduling of maintenance activities [11].

An example of a service network design problem in the freight rail industry is studied

in Zhu et al. [35]. The paper develops a model that addresses several aspects of designing the service network of a freight rail carrier including train selection and scheduling, railcar blocking selection and shipment routing. While the scope of the tactical planning problem in this paper is indeed ambitious, one challenge faced by freight rail carriers in implementing the solution is implementing the cross-organizational software system required to do so. However, the modeling framework and solution methodology used are interesting and relevant to the techniques used in this study, and they are investigated further in the literature review sections of subsequent chapters.

Another interesting application of network design in the rail industry is the blocking problem. Barnhart et al. [4] model the blocking problem using a network design framework with maximum degree and flow constraints on the graph nodes. The paper proposes a mixed integer programming formulation, which is solved using a Lagrangian relaxation heuristic. Bodin et al. [6] attempt to solve a railroad blocking problem using a non-linear mixed integer programming formulation. One significant simplification made in the paper is the absence of explicit modeling of time: it is simply assumed that the number of shipments with any given origin-destination pair is independent of time. The presented formulation views customer demand in terms of number of shipments demanded between any given ordered pair of rail terminals, and is structured as a multi-commodity network flow problem with additional side constraints.

It is worth noting that this dissertation involves a significant number of computational experiments in each of the three chapters, setting up the data for which utilizes solving the shortest path problem induced on the graph of a railroad's physical network. In particular, the shipment trip plan data is generated by finding the shortest path between the shipment origin and shipment destination. Finding a solution to the shortest path problem is treated as a trivial problem in this dissertation since:

1. Theoretically it can be solved in polynomial time, for instance with Dijkstra's algorithm [15].

2. Using efficient data structures such as Fibonacci heaps it can be solved very efficiently, with complexity $O[m + nlog(n)]$ or better [1].

1.2 Overview of Thesis

This dissertation offers a comprehensive study of a specific operational problem for the freight transportation industry. The problem of interest involves re-routing a subset of trains in the event of large-scale disruptions to the rail network in order to minimize the impact on network performance. In chapter *II* we discuss a framework for assessing the performance of a typical train network in this context. We do this by using a simulation-based methodology which explicitly models railcar movement through the network over time using rule-based railcar routing and handling logic. The methodology can be adapted to simulate any particular railroad's physical network, train schedule, and standard operating procedures for railcar handling at rail terminals. It allows us to explicitly account for the impact of joint decisions on railcar traffic patterns throughout the rail network. We will demonstrate that this framework enables us to solve certain problem instances to optimality through a systematic enumeration procedure.

In chapter *III* we extend the framework of chapter *II* to larger problem instances. First we show that the railcar movement model enables us to develop a linear, arc-based exact-optimization solution approach that uses an extended multi-commodity network design methodology and utilizes the unique structure of the problem to ensure tractability and reasonable solution time. Second we show that the framework from chapter *II* also enables us to develop a Tabu search heuristic solution approach that can generate reasonable solutions in a real-time context. The two solution approaches are then compared.

In chapter *IV* we consider the impact on the problem if rail terminals cannot be modeled as standard queues. We extend the arc-based optimization solution approach from chapter *III* to a more general solution approach capable of handling more flexible assumptions on potentially non-independent rail terminal processing capacity. We also extend the Tabu search

heuristic solution approach from chapter *III* to handle the flexible new set of assumptions. We then make a comparison of the two approaches under the new assumptions.

Chapter *II*: A Framework for Evaluating Dynamic Freight Train Re-Routing Decisions to Mitigate Large-Scale Disruption Scenarios

2.1 Introduction

The sheer scale of the $65.8 billion rail industry in the United States inevitably invites operational challenges of different kinds for Class I railroads. One such challenge is that of finding ways to mitigate the impact of unusual congestion and/or disruption at rail terminals. Disruptions at a rail terminal can be related to severe weather, workforce-related issues such as strikes, equipment or track issues at the rail terminal, etc. These events can disrupt the ability of the rail terminal to process railcars. Specifically, a disruption may temporarily reduce the processing capacity at the rail terminal, potentially down to zero. In order to mitigate the consequences of this impact, two approaches may be pursued by the railroad to remedy the problem, potentially in tandem. The first approach is to alter the operational schedule within the disrupted rail terminal to minimize the impact of the disruption. This problem has been studied in the literature, see Lusby et al. [24] and Caprara et al. [7] for examples.

The second approach, and the focus of our research, is to re-route a subset of trains to alternate rail terminals. When the disruption is imminent the operations management of the impacted rail terminal(s) would request that dispatch re-route one or more trains that will be impacted by the disruption — trains typically destined for the disrupted rail terminal. We shall refer to such trains in consideration as *candidate trains* and use L to represent the set of candidate trains. Dispatch would consider potentially altering the rail terminal to which a number of the candidate trains are destined. The expectation for making the re-route decision for candidate trains can be anywhere from a few hours to within an hour, with the latter scenario arising in cases where one or more candidate trains may have already been dispatched to the disrupted terminal. As such this problem needs to be solved in a

real-time context. Prior to this study the re-routing decisions were made as follows.

- **Step 1:** $\forall\ \ell \in L$, a score was determined for each potential destination $s \in S_\ell$, and the re-route options for ℓ were sorted from best to worst. The primary objective is reduce delays, and this heavily impacts the score. Other factors contributing to the score include the impact of each re-route option on crew and other resource requirements.
- **Step 2:** Based on the ℓ sorted lists a judgement call is made about how many trains to re-route and where to re-route each one of those trains.

The above approach is myopic in two ways since it ignores the following two considerations:

1. **Holistic Impact of Re-Routing on the Service Network:** Changing the destination of any given train has a direct impact on the congestion patterns at the original destination terminal as well as the new destination terminal. This impact is propagated throughout the network as other railcards are impacted since the re-routing decision creates new congestion patterns.
2. **Joint Re-Routing Decisions:** The optimal decision for one candidate train in the absence of other candidate trains is not necessarily part of the optimal overall decision when other candidate trains are present.

We address both of the above challenges by introducing a decision support methodology which provides dispatch with visibility into the holistic impact of joint re-routing decisions — thus allowing a non-myopic solution to the problem. We are interested in problem instances where thousands of railcars enter the network daily and will test our ideas using data from one of the largest railroad companies in the United States. In this chapter we limit the discussion to the subset of such problem instances where the solution space, $\prod_{\ell \in L} |S_\ell|$, is reasonably small in size. In subsequent chapters we remove this artificial constraint. Let us next discuss the related literature prior to introducing the details of our modeling approach to the problem.

2.2 Literature Review

There have been a number of studies that focus on improving train operations within a rail terminal. Caprara et al. [8] discuss train routing within a rail terminal by formulating an integer linear programming model which is solved using a branch-and-cut-and-price method. Caprara et al. [7] expand on this work by considering the impact of train delays and the resulting re-routing implications by developing a robust optimization model. Potential changes to the rail terminal operations in the event of a schedule disruption are studied by Lusby et al. [24], where a set-packing-inspired formulation is developed and solved using a branch-and-price based solution method. Petersen [28] models the operations at a rail terminal as a queueing system and makes the assumption that the arrival and departure processes are statistically independent Poisson processes. Using this model Petersen [28] calculates the probability distribution of put-through times and compares the model results with empirical data and finds that the exponential service time assumption works reasonably well for the prediction.

The research in our study aims to enable a decision support system for a freight train operator in the United States. Such systems have been implemented and studied in the transportation industry before. For example, de Matos and Powell [13] discuss requirements of such a system in the airline industry and address the issues of integrating such an automated support system into existing air traffic management systems. Rosenberger et al. [31] model the daily operations of an airline as a stochastic process. They implement the stochastic model as a simulation with modular components, each of which can be changed to reflect the specific policies of a given airline. The simulation implementation of the model allows the airline to evaluate crew scheduling plans and recovery policies when flights are delayed. Rosenberger et al. [32] build on this work by developing an optimization model to make decisions to re-route aircraft in order to minimize re-routing and cancellation costs. In order to keep computational time manageable, the paper develops an aircraft selection

heuristic which determines the subset of all aircraft to be considered for re-routing, while ensuring this subset remains a superset of all delayed aircraft. The impact of the re-routing decisions is evaluated using the comprehensive simulation model developed by Rosenberger et al. [31]. Argüello et al. [2] take a different approach, solving the aircraft re-routing problem with an overall greedy randomized adaptive search procedure (GRASP) heuristic solution, which solves in polynomial time.

The methodology we develop in this study includes using a time-space network. For a few decades, time-space networks have been used as a methodology for modeling transportation routing problems. Jarvis and Ratliff [19] demonstrate the ability to formulate three classic routing problem objectives using a time-space network model. They use a *discretized* time-space network, where time is discretized by creating nodes at pre-determined intervals which are generally regularly spaced. By contrast, Kliewer et al. [21] formulate an *event-based* time-space network, where nodes correspond to specific events at the various locations. The event-based time-space network is then applied to efficiently solve a multi-depot multi-vehicle type vehicle routing problem. The time-space network we develop in this paper is of the latter type.

The flexibility of the time-space network formulation is further demonstrated in the work of Zhu et al. [35]. They develop a three-dimensional time-space network consisting of three two-dimensional time-space sub-networks corresponding to trains, blocks, and individual railcars respectively. This framework for modeling the movement of railcars allows incorporating the train selection logic, blocking logic, and individual railcar routing to be incorporated as decision variables in a mixed integer programming math model. Since the mathematical model cannot be reasonably solved for realistic problem instance sizes, Zhu et al. [35] develop a metaheuristic solution approach that applies a dynamic block generation mechanism and a search intensification mechanism.

2.3 Problem Definition

As the candidate trains selected for re-routing are generally manifest trains we limit the focus of this dissertation to the manifest train network. The manifest train network excludes local train connections into and out of the rail terminals serviced by manifest and other non-local trains. Railcars are considered to enter the manifest train network when they are dropped off by a local train; similarly railcars are considered to leave the manifest train network when they arrive at the rail terminal from which they are picked up by a local train.

The railroad operates a physical network consisting of rail terminals and manifest train connections between terminals. Figure 1 shows an example of a simple manifest freight train network as such. A connection between rail terminals s_m and s_n means there exists a manifest train route from s_m to s_n. Note that the connections do not necessarily need to be bi-directional in the general case, though this is the case in this illustration.

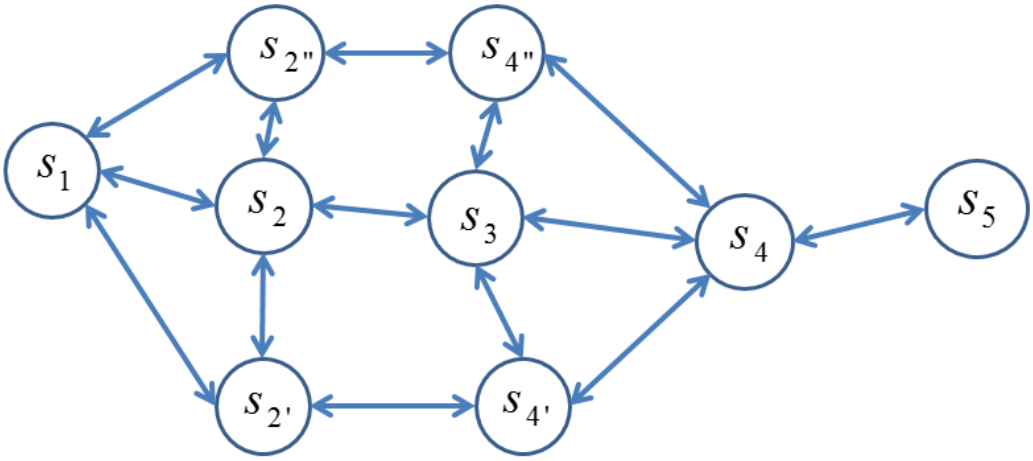


Figure 1: Small example of a manifest rail network

The manifest service network of the railroad uses this physical infrastructure and involves running manifest trains between terminals, for which an a priori train schedule is set. Table 1 illustrates part of a weekly train schedule summary for the physical network in Figure 1.

Each commodity has an a priori *trip plan*, which specifies the sequence of rail terminals it will traverse during its trip through the manifest network. Figure 2 depicts the trip plan of

Table 1: Weekly manifest train schedule sample

Train Origin	Train Destination	Departure - Arrival Times	Capacity
s_1	s_2	Mon 10:00 - 17:00	100
s_4	s_3	Wed 3:30 - 16:15	100
s_2	s_2''	Thu 13:30 - 16:30	100

one particular commodity in the manifest network illustrated by Figure 1. The commodity in question has origin terminal s_1 and destination terminal s_5. Note that a commodity's trip plan is, among other factors, a function of the origin and destination terminals for that commodity (though it is determined as such *a priori*). The trip plan assigned to the commodity in question is $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5$. Each arc in the trip plan is referred to as a *leg* of the trip plan. It is assumed that for each leg of the trip plan, there must exist a train in the schedule whose origin and destination match the origin and destination of that leg, respectively. The origin terminal of the commodity is the terminal in the manifest network into which the commodity is brought by a local train. Similarly the destination terminal of the commodity is the terminal in the manifest network out of which the commodity is taken with a local train.

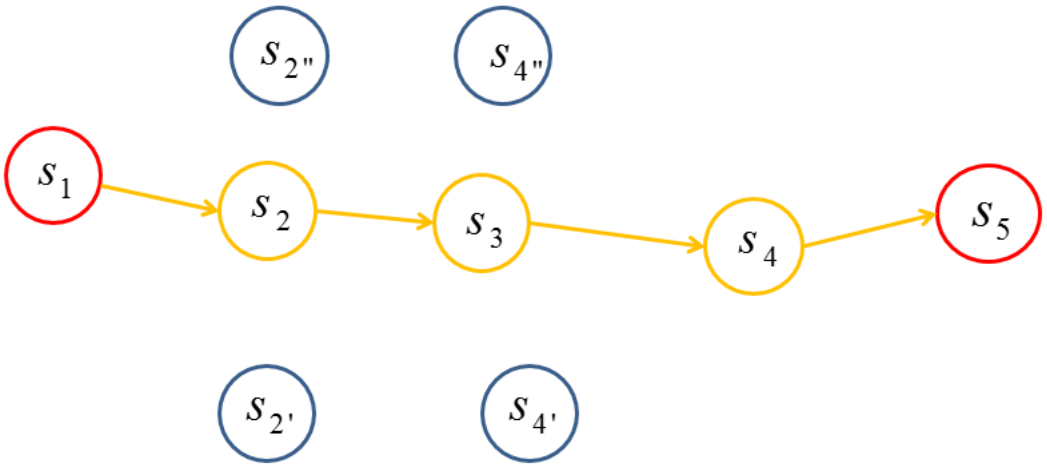


Figure 2: Illustration of one particular commodity's trip plan in the network

Any rail terminal in the manifest network s has a set of arrival tracks, a classification section which carries out the railcar processing, and a set of departure tracks, as shown in

Figure 3. Railcars that arrive at the terminal are positioned on the arrival tracks until they are processed at the classification section, at which point they are moved onto departure tracks and are ready to leave with the next applicable train. The processing rate of each rail terminal in the physical network is known a priori. We will let $\mu_s^{t_1,t_2}$ denote the average processing rate for railcars at rail terminal s during the time period starting at t_1 and ending at t_2 (i.e. processing capacity in terminal s between t_1 and t_2 will be $(t_2 - t_1)\mu_s^{t_1,t_2}$).

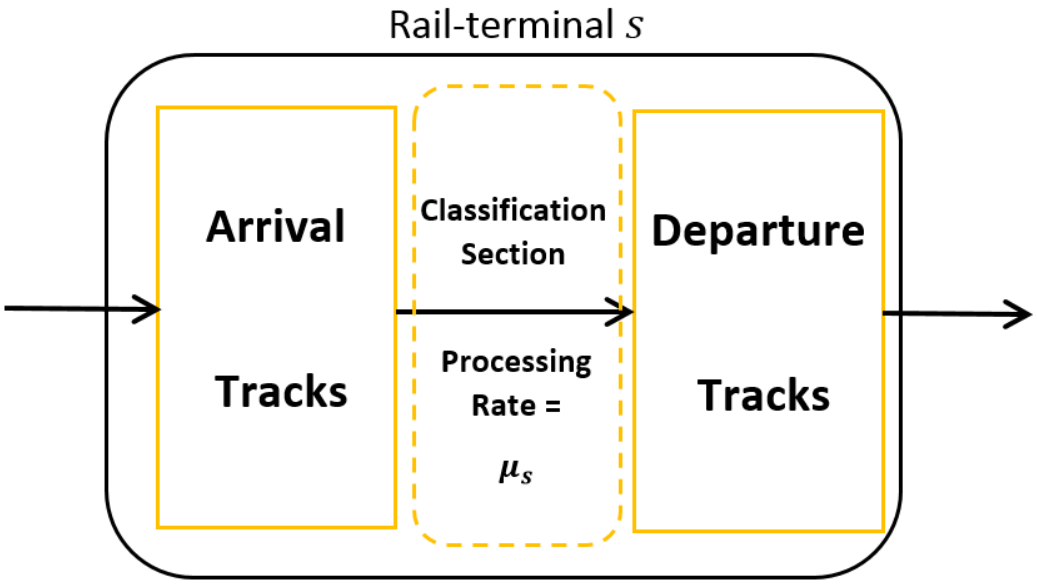


Figure 3: Rail terminal as it is modeled in this paper

Recall that the railroad is considering re-routing one or more candidate trains, regardless of whether already dispatched or not, due to unusual congestion at the original rail terminal for which the candidate train(s) are bound. In this section we define the details of this problem.

Consider a set of candidate trains, L, where each $\ell \in L$ starts from rail terminal o_ℓ and ends at rail terminal s_ℓ. The problem is to choose new destination rail terminals s'_ℓ from a feasible subset of rail terminals, S_ℓ. In the previous chapter we described the process of *picking up* and *setting out* railcars at intermediate stops between train origin and train destination. One simplifying assumption made in our problem definition is the absence of

pick-ups and set-outs: each train $\ell \in L$ carries a subset of commodities starting from o_ℓ and ending at s_ℓ.

Once ℓ reaches its destination the commodities that comprised ℓ are processed via the sort operation at s_ℓ, and each commodity k is sent to the appropriate departure tracks at s_ℓ for the next leg of the trip plan of k. If the destination of ℓ is changed to s'_ℓ, all commodities carried by ℓ will go through this process at s'_ℓ instead of at s_ℓ. Thus the decision to re-route train ℓ will alter the individual trip plan of every commodity k on ℓ. The new trip plan of commodity k describes how k will traverse the physical network from s'_ℓ to the final destination of k. Figures 4 and 5 illustrate this visually for one commodity.

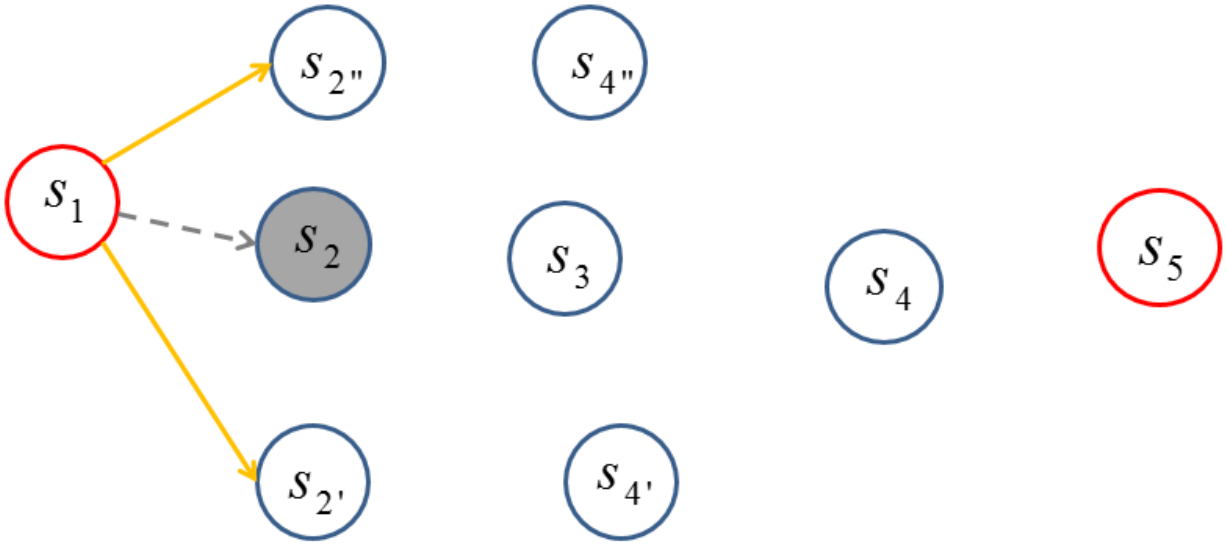


Figure 4: Train ℓ goes from s_1 to s_2, train destination s_2 is experiencing unusual congestion: $S_\ell = \{s'_2, s''_2\}$ is the set of re-route options for ℓ.

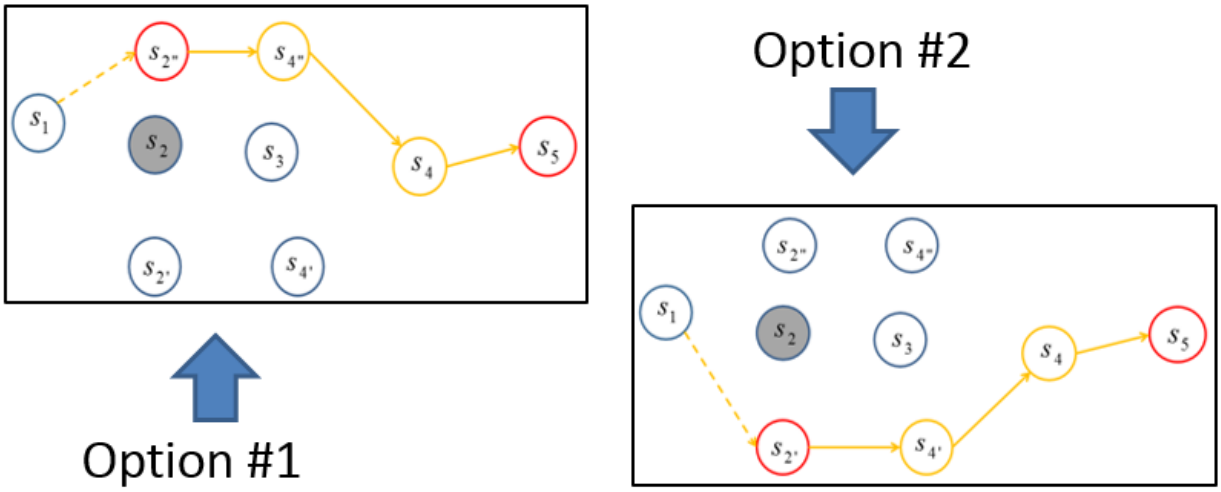


Figure 5: New trip plan for one specific commodity on ℓ for each re-route decision, commodity's final manifest destination is s_5.

The problem is formulated using the following data:

Train Data:

- Most recent train schedule prior to the re-route decision. This is generally (but not necessarily) a weekly schedule. The origin and destination terminals of all scheduled trains are known, as well as departure and arrival times.
- The capacity of each train, measured in terms of number of railcars.

Commodity Data: status and location of all manifest commodities currently in the network.

- Time of entry and manifest terminal of entry for all commodities entering the manifest network in the near future.
- The trip plan of each commodity in the manifest network (Note: the first and last rail terminals in the trip plan are the origin and destination terminals, respectively).
- The time at which each commodity is due at its destination terminal.

- The number of railcars in each commodity.

Rail Terminal Data:

- Processing capacity of each rail terminal in terms of number of railcars per unit of time which can be processed from the arrival tracks and sent to the departure tracks.

Disruption Data:

- The rail terminal(s) at which the disruption occurs.
- The duration of the disruption at each terminal.
- The reduced processing capacity of each terminal during the disruption, as a function of time.

Re-Route Options Data:

- The set of candidate trains, L.
- The set of available destination terminals, S_ℓ, $\forall\ \ell \in L$. S_ℓ may include the original destination of ℓ, choosing which would represent not altering the destination of ℓ.
- The new trip plan of any commodity k if candidate train ℓ carrying k is re-routed to s'_ℓ, $\forall\ \ell \in L, s'_\ell \in S_\ell$.

The above information defines the specific instance of the problem to be solved. The decision variables of the problem are $s_\ell, \forall \ell \in L$, as defined earlier. The railroad's primary objective in solving the re-routing problem is to improve network performance. In this study our focus is to improve network performance by reducing railcar delay. The delay associated with railcar c is defined as follows:

$$\delta_c = max(0, A_c - D_c) \tag{1}$$

, where A_c denotes the time of arrival of railcar c at its final destination rail terminal and D_c denotes the time railcar c is due at that rail terminal.

As such the objective statement of the problem is defined as:

$$min \sum_{c \in C_p \bigcup C_f} \delta_c \tag{2}$$

, where C_p represents the set of railcars in the rail network at the time the re-route decision needs to be made, and C_f represents the set of railcars coming into the rail network in the near future.

Let us discuss the constraints of the problem and how they impact the above objective. The constraints can be divided into two categories:

Solution Space Constraints: These are direct constraints on the decision variables.

1. The set of candidate trains, L is pre-specified.

2. Each candidate train has a pre-specified set of possible destinations: $S_\ell, \forall \ell \in L$.

The feasibility of all members of S_ℓ is guaranteed for train ℓ. This is realistic since the existing process for figuring out the re-routing destinations assumes the same. Specifically, a subset of the alternate destinations may require additional resources such as an additional crew shift if the train travels a significantly longer distance. It is assumed that this subset is included in S_ℓ only if the additional crew can be feasibly obtained.

Railcar Flow Constraints: These can be divided into four subcategories: *schedule constraints*, *capacity constraints*, *standard operating procedure constraints*, and *trip plan constraints*.

Trip Plan Constraints:

- A commodity must travel through the physical network according to its assigned trip plan, which ends with the railcar's final destination rail terminal.

- No commodity is cancelled: once it enters the service network, it will be taken to its final destination by traversing the terminals specified in its trip plan.
- It is known a priori how any feasible re-route decision will impact each re-routed railcar's trip plan.
- The decision to re-route train ℓ modifies the assigned trip plan of each commodity travelling on ℓ.

Train Schedule Constraints: A commodity can only travel through a leg of its trip $s_1 \rightarrow s_2$ plan at a time when there is a train available from s_1 to s_2.

Capacity Constraints:

- *Train Capacity:* Each train in the schedule has a maximum number of railcars it can carry. This is generally due to the power capacity of the locomotive(s) assigned to the train, although there may be other factors in some instances. This is specified as part of the a priori train schedule. This type of capacity is measured in terms of the number of railcars.
- *Rail Terminal Processing Capacity:* The classification section of each rail terminal has a finite processing capacity. This type of capacity is measured in terms of the number of railcars per unit time. If a terminal is congested over a given time period, this value is updated to reflect the reduced processing capacity for the specified period of time.

Standard Operating Procedure Constraints:

- The railroad operates based on rules that dictate how railcars are prioritized for processing at rail terminals. These rules must be followed.
- The three network resources discussed under *capacity constraints* operate below their effective physical capacity at any given time if and only if there is no additional railcars for them to feasibly process.

The four railcar flow constraints above dictate when and where railcars move through the physical network.

2.4 Model

The decision to re-route any train originally scheduled to go from $s \rightarrow s'$ to a new rail terminal, s'', modifies the traffic patterns in the network, increasing railcar volume at s'' and reducing railcar volume at s'. This has a potential impact on a much wider set of railcars than just the set on the re-routed train. This impact is then propagated through the network as railcars move based on their trip plans. To account for this inherent dynamic, in this section we develop a methodology to explicitly quantify the impact of re-routing decisions on the movement of each individual railcar in the rail network. We do this in two sequential steps:

1. Build a time-space network that codifies the schedule constraints and capacity constraints introduced in section 2.3.

2. Using the time-space network, build a railcar movement model which, given the trip plan constraints and standard operating procedure constraints defined in section 2.3, evaluates the network's performance given any re-routing decision for an individual train or joint re-routing decision for multiple trains at once.

2.4.1 Event-based Time-Space Network Framework

In general a time-space network is a specific type of graph, where each node corresponds to: (1) a location, (2) a point in time. Figure 6 illustrates a generic small example. In this case the time-space network is *discretized* along the time dimension, since nodes are placed at pre-determined points in time for each location. Three locations and four *time periods* are modeled. The horizontal arcs are referred to as *waiting arcs*; any flow along a waiting arc models storing or staging goods at the corresponding location. The arcs connecting

nodes corresponding to different locations are referred to as *movement arcs*; any flow along a movement arc models the transportation of goods from one location to another.

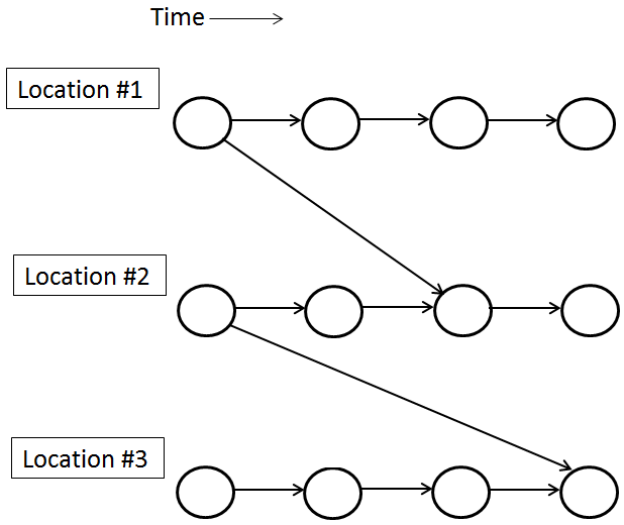


Figure 6: Generic discretized time-space network

For our modeling purposes we use an *event-based* time-space networking model instead of a discretized one. In an event-based time-space network the waiting arcs are not necessarily of uniform size. Instead, nodes are placed at key points in time for each location and connected to create the waiting arcs. We determine such key points in time using the train schedule as input. The nodes are created to correspond to train arrival and departure times: the movement arcs in the time-space network correspond to trains (henceforth known as *train arcs*) and the waiting arcs correspond to the storage of railcars at rail terminals (henceforth known as *terminal storage arcs*). The tail node of a train arc is referred to as the *train departure node*. The head node of a train arc is referred to as the *train arrival node*, which allows us to model when train cars have moved to a terminal's arrival tracks for processing. Figure 7 depicts an example, where we see a subset of the event-based time-space network which includes:

- An $s_1 \rightarrow s_2$ train with cutoff time 11:00 and arrival time 16:15.
- An $s_2 \rightarrow s_3$ train with cutoff time 10:30 and arrival time 17:10.
- A train which departs from s_1 with cutoff time 17:00.

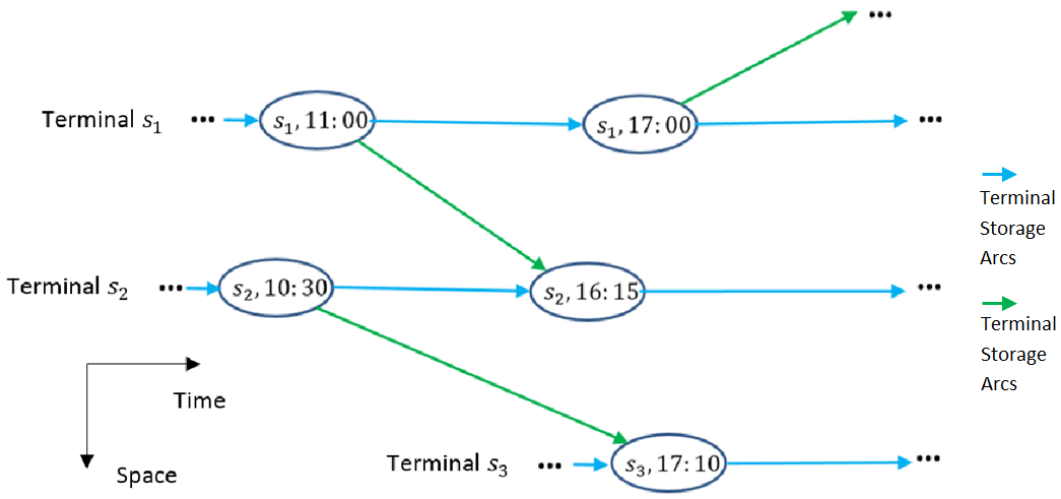


Figure 7: Time-space network set-up

This formulation allows us to represent potential train re-routing decisions with the introduction of additional train arcs in the time-space network, as shown in Figure 8. Thus, a non-candidate train induces one train arc in the time-space network, while a candidate train ℓ induces a set of train arcs equal in cardinality to the number of re-route options available to ℓ, plus one for the original schedule and route if it remains feasible.

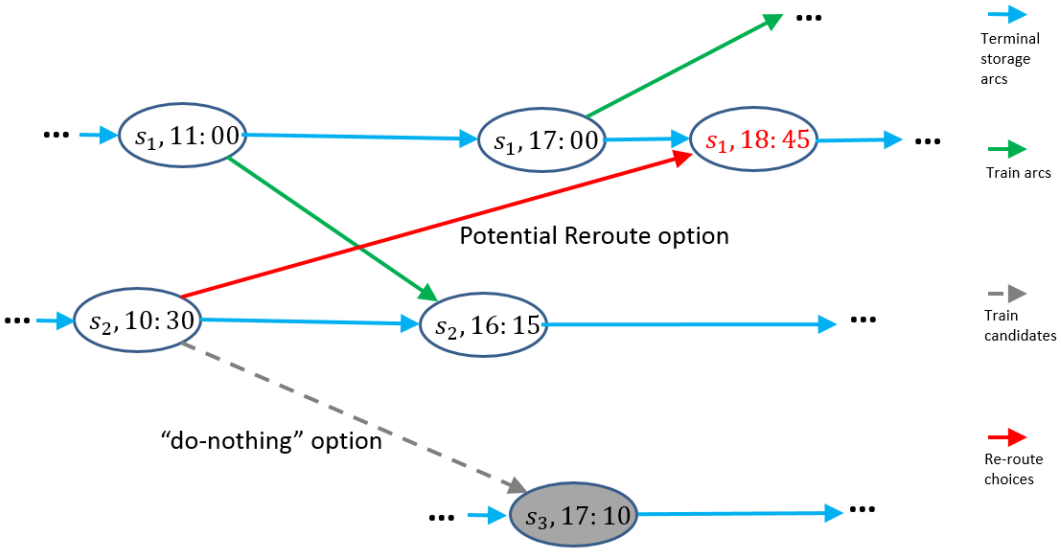


Figure 8: Time-space network with re-route option for train $s_2 \rightarrow s_3$ added.

Next we use the time-space network to explicitly model the processing of railcars at rail

terminals from the arrival tracks to the departure tracks. One can represent this process by including two separate sets of terminal storage arcs along the space dimension to account for the arrival and departure tracks at each rail terminal separately. This allows us to model the processing of railcars from the arrival tracks to the departure tracks by including *terminal processing arcs*, also referred to simply as processing arcs. For each train arc, two processing arcs along with four auxiliary nodes are added using the following two steps:

1. The first and second auxiliary nodes are added along the terminal storage arcs which correspond to the arrival and departure tracks of the train origin terminal, respectively, at ϵ time units prior to the train departure node. They are connected, forming the processing arc. Figure 9 shows this visually.

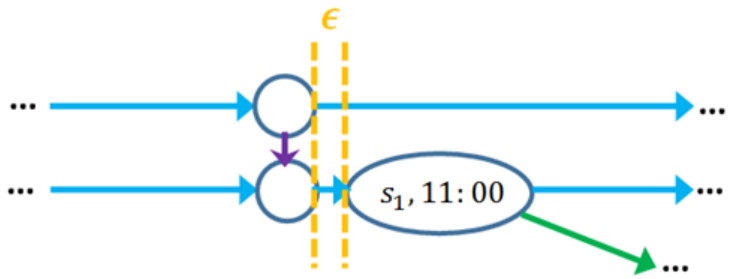


Figure 9: Processing arc added prior to train departure node

2. Along the terminal storage arcs which correspond to the arrival tracks of the train destination terminal the third auxiliary node is added at ϵ time units prior to the train arrival node. The fourth auxiliary node is added along the terminal storage arcs which correspond to the departure tracks of the train destination terminal at the same point in time as the third auxiliary node. Figure 10 depicts this.

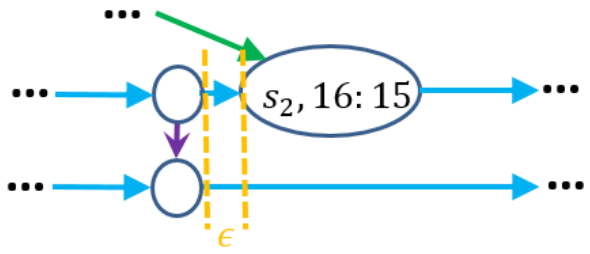


Figure 10: Processing arc added prior to train arrival node

Let $s(i)$ and $t(i)$ denote the rail terminal and point in time to which time-space node i corresponds, respectively. The resulting time-space network is particularly useful since it allows us to account for the processing rate of each terminal at all points in time by setting the capacity, U_{ij} of any processing arc (i, j) to the value:

$$U_{ij} = [t(i) - t(i')]\mu_{s(i)}^{t(i'),t(i)} \tag{3}$$

, where node i' denotes the predecessor of node i such that $s(i') = s(i)$. Figure 11 illustrates the procedure for setting the capacity of processing arcs. In the depicted example $t(i) - t(i') = 360$ minutes. Thus, if $\mu_{s_1}^{t(i),t(i')} = 10$ railcars per hour, the capacity of arc (i, j) in Figure 11 will be set to 3600 railcars.

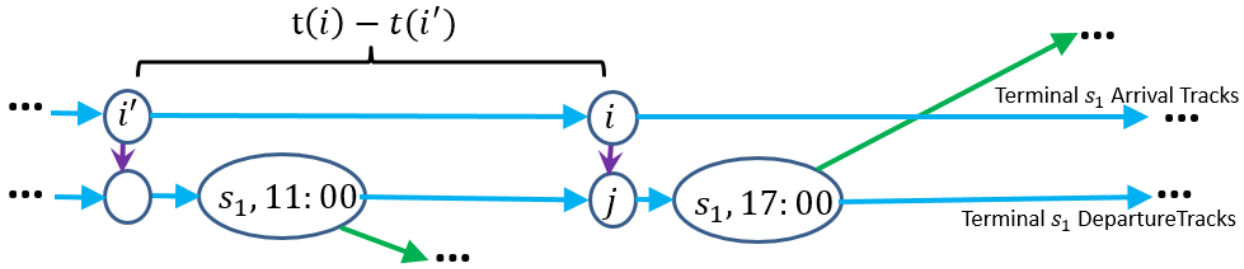


Figure 11: Example of processing arc's upper bound calculation.

Observe that, in a strictly literal sense, the proposed time-space framework assumes a batch of railcars are processed from the arrival yard to the departure yard simultaneously in one instance. For example, in Figure 11, a batch of U_{ij} railcars or less can be processed

instantaneously at time $t(i)$. This is of course not the case in reality, where railcars are processed continuously. However, given the assumption of constant processing rate of $\mu_{s_1}^{t(i),t(i')}$ between $t(i')$ and $t(i)$, we assert that the proposed time-space framework is sufficient for modeling this reality.

Consider a more literal representation of the processing of railcars between time-points $t(i')$ and $t(i)$, where there are U_{ij} processing arcs between these time-points, each with upper bound $U = 1$. Since the processing rate between time-points $t(i')$ and $t(i)$ is constant, the processing arcs will be placed after equal intervals, $\frac{1}{\mu_{s_1}^{t(i),t(i')}}$ apart. This is represented in the alternate time-space network formulation shown in Figure 12. Note that:

- Figure 12 is more expensive to formulate than Figure 11, since it uses more nodes and arcs to represent processing.
- Figure 11 allows correct modeling of physical storage capacity at the arrival and departure yards at the time of train arrival and train departure.

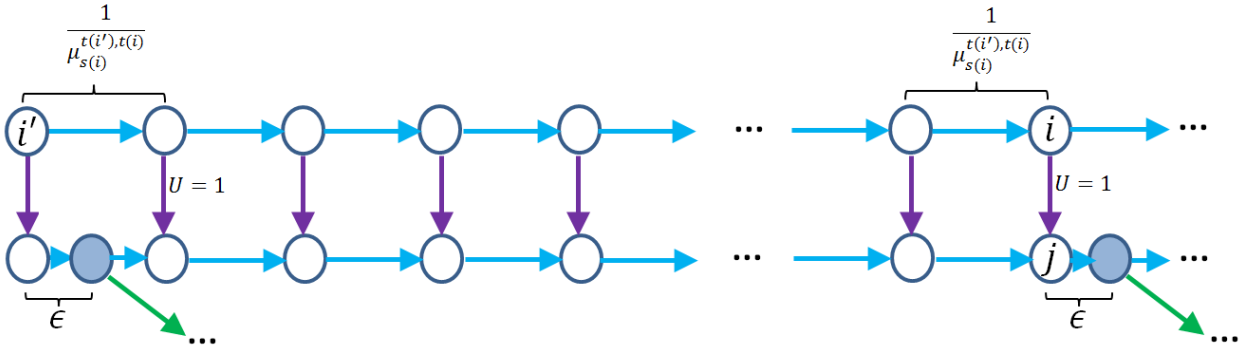


Figure 12: Continuous processing modeled in time-space paradigm

Hence we use the event-based time-space framework in Figure 11 instead of the discretized time-space framework in Figure 12. the addition of the resulting auxiliary nodes and processing arcs to the example time-space network of Figure 7 results in the time-space network shown in Figure 13.

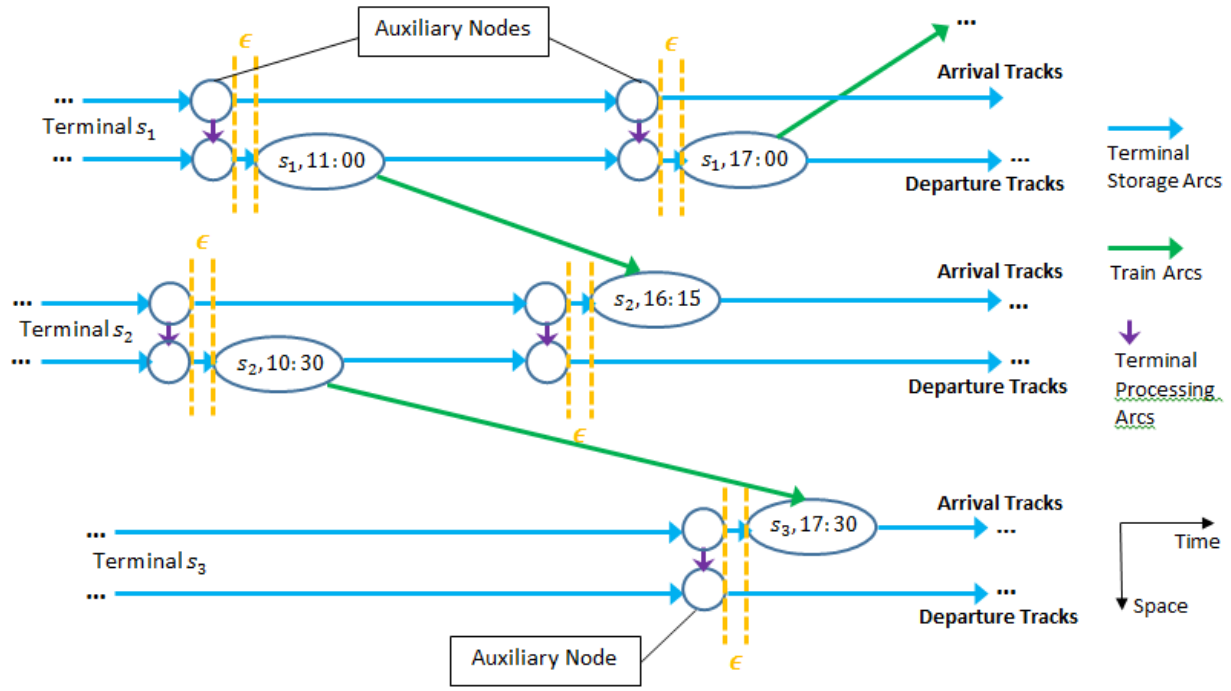


Figure 13: Time-space network including processing arcs

In addition to representing the movement, storage, and processing of railcars, we also need to model the exit of railcars from the network once they reach their final destination. To do this we add one sink node, i_{sink}, to the time-space network. This allows us to connect each train arrival node to the sink, creating *exit arcs*, which represent commodities leaving the system. This is shown in Figure 14. Furthermore, a time horizon must be selected beyond which no arcs are to be created. For the arrival tracks of each rail-yard, we connect the node i which has the latest $t(i)$ along the nodes at that arrival track to the sink node. An arc is similarly created for the departure tracks of each rail-yard. Let us refer to this class of arcs as *feasibility arcs*.

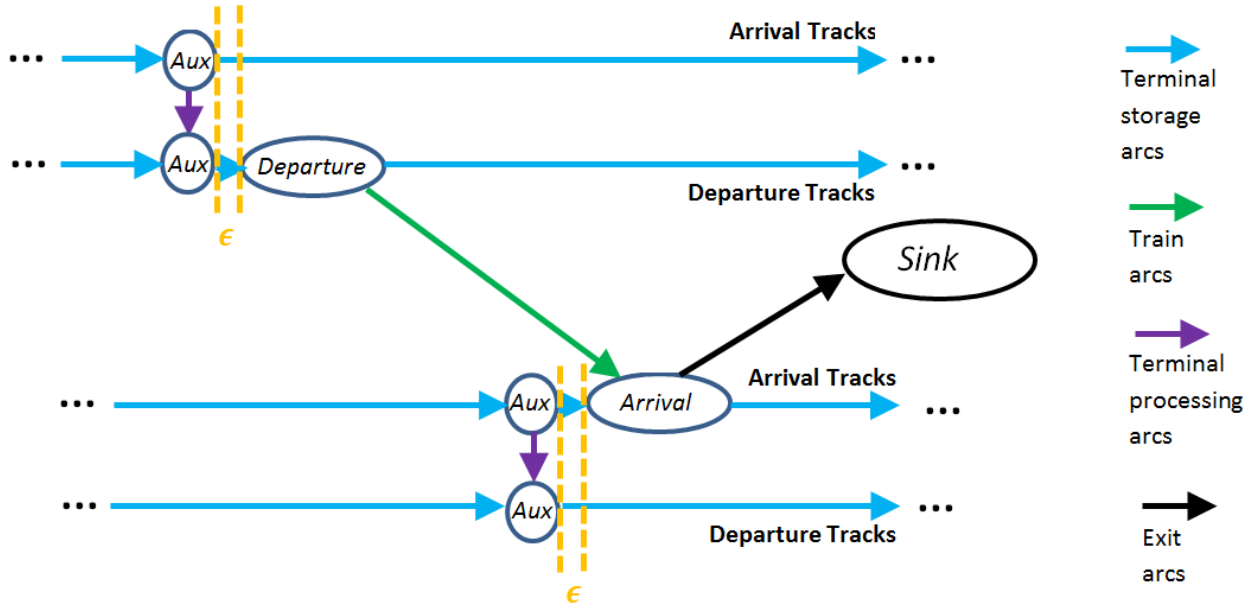


Figure 14: Illustration of sink node and an exit arc example in time-space network

Let $G = (V, A)$ represent the final time-space network formed as described in this section, where V is the set of nodes and A is the set of arcs in the graph. We note that any node $i \in V$ corresponds to: either the arrival yard of rail terminal $s(i)$ or the departure yard of rail terminal $s(i)$. The exception is the sink node i_{sink}, for which $s(i_{sink})$ and $t(i_{sink})$ are not defined.

In order to accurately represent the operations of the railroad, we set the arc capacities as described below. All arc capacities are measured in number of railcars.

- The capacity of any train arc is set to the capacity of the corresponding train in the train schedule.
- The capacity of any processing arc, $(i, j) \in A$, is set using Equation (3).
- The capacity of any terminal storage arc, exit arc, or feasibility arc is set to ∞.

In the next section we will assert that this directed, capacitated time-space network can be used as an accurate simulator of railcar movement through the physical network. We will show that this simulation capability allows us to solve the problem defined in section 2.3 for reasonably small problem instances by running a separate simulation of railcar movement for each potential solution.

2.4.2 Simulation of Railcar Movements

Evaluating any re-route decision requires estimating the resulting railcar movements in the network and consequent aggregate delay accrued by commodities en route to final destinations. Note that a given feasible flow on the time-space network specifies:

- How railcars move through the physical network.
- The timing of all such movements.

As such, in this study we characterize railcar flow solutions in the physical network using railcar flows on the time-space network. In this section we introduce an algorithm that creates a feasible flow on the time-space network that is a good simulator of actual railcar movements in the physical network. The algorithm requires the following five inputs:

1. $G = (V, A)$, The time-space network.

2. C_p, the current set of railcars in the network, including the time due, current location, and remaining trip plan for each railcar $c \in C_p$.

3. C_f, the set of railcars projected to arrive into the network in the near future, including the time due and trip plan for each railcar $c \in C_f$. The near future is defined as the time horizon covered by the time-space network.

4. $\tau(c)$, the time at which railcar c is introduced to the physical network. If $c \in C_f$, c is available for movement at the arrival tracks of $s_o(c)$ into the network at time $\tau(c)$, where $s_o(c)$ is the first rail terminal in the trip plan of c. If $c \in C_p$, c is available for movement at the section (arrival or departure) of the current rail terminal where it is currently stationed and $\tau(c)$ is assumed to be the start of the time horizon. Note that any railcars on a moving train at the current time can be included in C_f and assumed to arrive into the network at the train destination and at the time of train arrival with a truncated trip plan, without loss of generality.

5. Operating rules regarding the order of routing and processing of railcars in $C_p \cup C_f$ through the physical network. Operating rules reflect the standard operating procedure constraints for any specific railroad in question. As an example, we model the application of a *first-in-first-out (FIFO)* railcar processing rule at its rail terminals. Such a rule would state that railcars are to be processed at a rail terminal in the order in which they have arrived at the arrival tracks. It would further require that trains are to be loaded with applicable railcars at the departure tracks in the order the railcars have been received at the departure tracks[1].

An example of a FIFO violation is shown in Figure 15, where one railcar bypasses another at the classification section. Note that railcar #2, which takes the red path, bypasses railcar #1 at terminal s_1, violating the first-in-first-out rule.

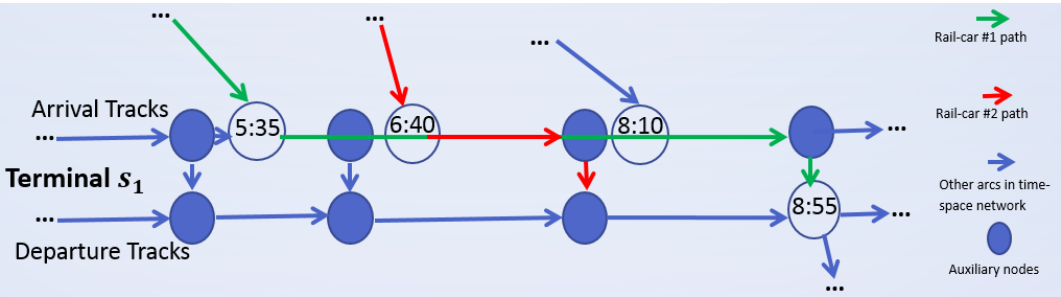


Figure 15: Example of FIFO violation by railcar with red path over railcar with green path.

In order to evaluate the performance of the network in terms of the objective function defined in section 2.3, Equation 2, one needs to create a feasible flow on the time-space network that is a good simulator of actual railcar movements in the physical network. Algorithm 1 accomplishes this using the assumptions regarding the railroad's standard operating procedures discussed above, and outputs the flow of railcars through the time-space network. We represent this output with the following variables:

[1]Note that the operating procedure modeled here is one example of such rule. For instance, a potential alternative operating rule may assume that each arriving track in the arrival yard is processed FIFO. It is furthermore possible to create a more complex time-space network with multiple sets of terminal storage arcs, each set representing one of the tracks in the arrival or departure yard.

$$x_{ij}^{c} = \begin{cases} 1 & \text{if railcar } c \in C_p \cup C_f \text{ is to flow on arc } (i,j) \in A; \\ 0 & \text{otherwise.} \end{cases}$$

The algorithm is described in detail below.

Algorithm 1 Railcar movement model applied to network with FIFO rule.

Require: $G = (V, A)$, C_p, C_f, $U_{ij} \forall (i,j) \in A$.

Require: $t(i), s(i) \quad \forall i \in V$.

Require: $\tau(c) \quad \forall c \in C_p \cup C_f$.

Step 0 - Initialization:

For each rail terminal s, initialize $Q_A(s)$ and $Q_D(s)$ as two sorted sequences or queues of railcars, each starting with no elements.

For each train arc (i,j), Initialize an empty sorted sequence $Q_{(i,j)}$ as the queue of railcars to traverse (i,j).

Set $x_{ij}^c = 0 \quad \forall (i,j) \in A, \forall c \in C_p \cup C_f$.

Step 1 - Add current railcars to queue

Step 1a: Sort the railcars in C_p into sequence $\overline{C}_p$ in non-decreasing order of $\tau(c)$.

Step 1b: Remove the first railcar c in $\overline{C}_p$ and note both its rail terminal, $\overline{s}$, and whether it is on the arrival or departure tracks.

Step 1c: If c is on the arrival tracks of $\overline{s}$:

Add c to the end of $Q_A(\overline{s})$.

If c is on the departure tracks of $\overline{s}$:

Add c to the end of $Q_D(\overline{s})$.

Step 1d: If $\overline{C}_p$ is empty:

Go to Step 2.

Else:

Go to Step 1b.

Step 2 - Create sequences for near-future railcars: For each rail terminal s:

Place the subset of railcars in C_f that come into the network at s into sequence $\overline{C}_f(s)$.

Sort cars in $\overline{C}_f(s)$ in non-decreasing order of $\tau(c)$.

Step 3 - Sort nodes by point in time: Sort the nodes in V into sequence V_{sorted} in order of non-decreasing $t(i)$. Break ties by placing nodes that correspond to arrival tracks prior to nodes that correspond to departure tracks.

Step 4 - Pick next node: If there are no remaining nodes in V_{sorted}:

Terminate the algorithm.

Else:

Remove the first node i from V_{sorted}.

If i corresponds to the arrival tracks of $s(i)$:

Go to Step 5.

If i corresponds to the departure tracks of $s(i)$:

Go to Step 6.

Continued on next page

Step 5 - Arrival node steps

Step 5a: If $\overline{C}_f[s(i)]$ is empty or the first railcar c in $\overline{C}_f[s(i)]$ has $\tau(c) > t(i)$, skip to Step 5b. Otherwise, remove c from $\overline{C}_f[s(i)]$, add it to the end of $Q_A[s(i)]$, and repeat Step 5a.

Step 5b: For all nodes $i' \in V$ such that $\exists$ a train arc $(i', i) \in A$: Add all railcars in $Q(i', i)$ to the end of $Q_A[s(i)]$.

Step 5c: If $Q_A[s(i)]$ is an empty sequence, go to Step 4.

Step 5d: If processing arc (i, j) departs i:

Set $n = max(U_{ij}, |Q_A[s(i)]|)$, where $|Q_A[s(i)]|$ is the number of cars in $Q_A[s(i)]$.

For each of the the first n cars from $Q_A[s(i)]$, denoted c:

Remove c from $Q_A[s(i)]$.

Set $x_{ij}^c = 1$.

Add c to the end of $Q_D[s(i)]$.

Step 5e: Let (i, j) denote the rail terminal storage arc or feasibility arc leaving node i. Set $x_{ij}^c = 1 \quad \forall c$ such that c is in sequence $Q_A[s(i)]$.

Step 6 - Departure node steps

Step 6a: If $Q_D[s(i)]$ is an empty sequence, go to Step 4.

Step 6b: If $\exists j \in V$ such that (i, j) is a train arc, let $Q_D^{(i,j)}$ represent the ordered sub-queue of railcars in $Q_D[s(i)]$ whose trip plan includes $s(i) \rightarrow s(j)$ as a leg.

Set $n = max(U_{ij}, |Q_D^{(i,j)}|)$.

Remove up to n railcars from $Q_D^{(i,j)}$. Remove the same set of railcars from $Q_D[s(i)]$.

For railcar c removed from $Q_D[s(i)]$:

Set $x_{ij}^c = 1$.

If $s(j)$ is the final rail terminal in the trip plan of c:

Set $x_{ji_{sink}} = 1$.

Else:

Add c to $Q_{(i,j)}$.

Step 6c: Let (i, j) now denote the rail terminal storage arc or feasibility arc leaving node i. Set $x_{ij}^c = 1 \quad \forall c$ where c is in sequence $Q_D[s(i)]$.

The algorithm eventually terminates in Step 4 when there are no nodes left in sequence V_{sorted} at the end of the time horizon.

Recall from section 2.3 the definition of railcar delay in Equation 1. If the problem is feasible, the delay incurred by each railcar at the time of arrival into its final destination

can be calculated based on the values of x_{ij}^c determined by Algorithm 1. To do so, for each railcar $c \in C_p \cup C_f$ we associate a delay cost, ρ_{ij}^c to each arc $(i,j) \in A$ as follows:

- If (i,j) is an exit arc, set $\rho_{ij}^c = max[t(i) - D_c, 0]$, where D_c is the due time of railcar c. Note that $t(i)$ and D_c correspond to two points in time. We define $t(i) - D_c$ as the amount of time, measured in minutes, between the two timepoints.

- If (i,j) is a feasibility arc, the delay cost must reflect two components:

  - *Future Delay:* The objective function needs to also account for the delay railcar c may incur in the future. We penalize future delay by setting a sufficiently large penalty term, P, and calculating a *penalty ratio* for railcar c as follows:

$$p_c = \frac{q_c^{s(i)}}{q_c} \tag{4}$$

    , where q_c is the number of legs in the trip plan of railcar c, and $q_c^{s(i)}$ is the remaining number of legs in the trip plan of c after rail terminal $s(i)$.

  - *Incurred Delay:* If D_c is an earlier time than $t(i)$, an additional delay of $t(i) - D_c$ is already incurred.

  To account for both components, we set $\rho_{ij}^c = max[t(i) - D_c, 0] + p_c\{P - [max(0, D_c - t(i)]\}$. Note that in this equation future delay is reduced if D_c is after $t(i)$, since that allows more time beyond the end of the horizon for railcar c to reach its destination without incurring a delay. Furthermore, the penalty ratio incentivizes railcar c to move towards its final destination through the legs of its trip plan, even if it may not reach the final destination by the end of the horizon.

- If (i,j) is neither an exit arc nor a feasibility arc: set $\rho_{ij}^c = 0$.

Based on the above data, the delay for railcar $c \in C_p \cup C_f$ is calculated below.

$$\delta_c = \sum_{(i,j)\in A} \rho_{ij}^c x_{ij}^c \tag{5}$$

As such, the objective function z can be calculated upon the completion of the algorithm as follows:

$$z = \sum_{c\in C_p} \sum_{(i,j)\in A} \rho_{ij}^c x_{ij}^c \tag{6}$$

2.4.3 Systematic Enumeration

In the previous section we developed an algorithm to simulate railcar flow in the time-space network. In this section we describe the final component of the overall solution approach of Chapter I: systematic enumeration of the potential solutions to the train re-routing problem.

Let Φ denote the sequence (ordered set) of all possible candidate train re-route solutions and note that $|\Phi| = \prod_{\ell\in L} |S_\ell|$. Any particular solution $\phi \in \Phi$ is characterized by specifying a destination terminal for each candidate train: $\phi = (\phi_1, \phi_2, ..., \phi_{|L|})$. Let us discuss how an instance of the railcar movement simulation can be run for each $\phi \in \Phi$.

We have shown that, by design, each scheduled train induces a train arc in the time-space network. Furthermore each potential re-route option for any candidate train (including the "do-nothing" option) also induces a train arc in the same network. Thus any candidate train $\ell \in L$ with corresponding set of re-route options S_ℓ induces $|S_\ell|$ train arcs in the time-space network. Let $A_\ell \subset A$ represent the set of such arcs. Recall from subsection 2.4.2 that the time-space network $G = (V, A)$ is one of the five required inputs for running the railcar movement simulation. The railcar movement simulation can be run for any particular $\phi \in \Phi$ by altering the capacities of the train arcs which correspond to candidate trains:

Algorithm 2 Running railcar movement simulation for scenario characterized by $\phi \in \Phi$

Require: $G = (V, A)$, $\phi = (\phi_1, \phi_2, ..., \phi_{|L|})$, $A_\ell \; \forall \ell \in L$.
Step 1: Modify $G = (V, A)$, the original time-space network.
For ℓ in $(1, 2, ..., |L|)$:
For each $(i, j) \in A_\ell$:
If $s(j) == \phi_\ell$:
Set U_{ij} to the capacity of train ℓ.
Else:
Set $U_{ij} = 0$.
Step 2: Run Algorithm 1 using the modified time-space network.

Furthermore, let us use $z(\phi)$ to denote the objective function value (i.e. aggregate railcar delay measured in railcar-minutes) obtained using Equation 6 after running Algorithm 1 for the scenario represented by ϕ.

We begin our attempt to solve the train re-routing problem by articulating the simplest way to find the solution with the smallest amount of resulting aggregate railcar delay: systematically enumerating each solution. Algorithm 3 accomplishes this using one possible method of sorting the solutions in the solution space for enumeration. The computational limitations of this method are studied in this chapter. More advanced methods are discussed in the next chapter.

Algorithm 3 Systematic Enumeration Algorithm

Initializations:
Initialize $\phi^0 = (\phi_1^0, \phi_2^0, ..., \phi_{|L|}^0)$, where ϕ_ℓ^0 is the original destination terminal $\forall\ \ell \in L$.
Set $\phi^* \rightarrow \phi^0$, $z^* \rightarrow z(\phi^0)$, $\hat{\Phi} \rightarrow \{\phi^0\}$, $\hat{\Phi}^{new} \rightarrow \{\phi^0\}$
Initialize $\ell = 1$.
Procedure:
While $\ell \leq |L|$:
  For each $\phi \in \hat{\Phi}$:
    For each $s \in S_\ell$:
      Set $\phi^{new} \rightarrow \phi$.
      Set $\phi_\ell^{new} \rightarrow s$.
      Add ϕ^{new} to the end of $\hat{\Phi}^{new}$.
      If time elapsed has passed time threshold:
        Exit all three loops.
      Compute $z(\phi)$ using Algorithm 2.
      If $z(\phi) < z^*$:
        Set $\phi^* = \phi$ and $z^* = z(\phi)$.
  Set $\hat{\Phi} \rightarrow \hat{\Phi}^{new}$.
  ℓ++.
Return ϕ^* and z^*.

2.5 Computational Experiments

In this section we test the optimization by systematic enumeration approach using computational experiments. The physical network and weekly train schedule data from Union Pacific Railroad, a Class I railroad in the United States, is used to set up the experiments. Shipment data is simulated to reflect realistic volume in the same network. Different examples of network disruptions are tested, as well as different candidate train sets, L, and candidate destinations, $S_\ell \forall \ell$. Variations in the number of candidate trains, $|L|$ are of particular significance since the size of the solution space increases exponentially as $|L|$ increases. We further motivate the use of the two metrics used to evaluate the effectiveness of systematic enumeration for different types of problem instances: the time required to exhaust the solution space and the quality of the solution retrieved in realistic time windows. This allows us to establish two things:

1. Which problem instances are reasonably small for systematic enumeration to suffice as a solution method.

2. The motivation behind the subsequent chapter, which tackles the problem instances that do not fall into the above category.

As a practical matter it is crucial for the railroad to understand under what circumstances a more complex solution methodology is required. This informs the prioritization decision of the investment in the organizational system required to execute the more complex methodology.

We begin by discussing the specific data inputs used to set up the computational experiments.

2.5.1 Physical Network

The physical network data obtained from the railroad indicates the presence of 152 rail terminals across a significant portion of the continental United States, and 2625 trains scheduled weekly between these rail terminals. This data is used to construct the relevant instance of the time-space network $G = (V, A)$ as described in section 2.4.1.

2.5.2 Simulation Horizon and Traffic Instances

Since most disruptions do not last beyond one or two days and the impact of a disruption is most acutely felt during and immediately after the fact, we consider a timeline of 14 days sufficient to evaluate the effectiveness of a given solution. In order to have comprehensive results, we generate 15 independent instances of manifest commodity traffic in the railroad's network. For each traffic instance a subset of commodities is generated as the manifest network traffic. Each such commodity is assigned the following attributes.

Number of Railcars: As discussed previously, a commodity may have one or more railcars — we refer to the number of railcars in a commodity as the *size* of the commodity. For

Table 2: Volume type and λ values for each traffic instance

λ	Volume Type		
	Regular	Heavy	Light
1.0	Traffic instance #1	Traffic instance #6	Traffic instance #11
1.2	Traffic instance #2	Traffic instance #7	Traffic instance #12
1.3	Traffic instance #3	Traffic instance #8	Traffic instance #13
1.4	Traffic instance #4	Traffic instance #9	Traffic instance #14
1.5	Traffic instance #5	Traffic instance #10	Traffic instance #15

each traffic instance we use a specific probability mass function $f(n)$ to determine the size of each commodity. Since the scope is limited to manifest trains, we assert it is reasonable to assume that commodity sizes are between 1 and 10 railcars, with *frequency of commodity size* inversely proportional to *commodity size*. Therefore we use the following exponentially decreasing probability mass function to generate the size of each commodity:

$$f(n) = \frac{\lambda^{10-n}}{\sum_{n'=1}^{10} \lambda^{10-n'}} \tag{7}$$

, where λ is an input parameter we vary across traffic instances. Table 2 summarizes the values of λ used for each specific traffic instance.

For the remaining commodity attributes, we define the same for each individual railcar c in the commodity.

Time of Entry to System (τ_c): Let us define two useful quantities below.

- R_0: the number of railcars present in the rail network at the beginning of the simulation. Note that $R_0 = |C_p|$.
- R_d: the average additional daily number of railcars coming into the rail network. Note that $R_d * 14 = |C_f|$.

Based on data provided in 2013 by one of the Class I railroads in the United States, we have determined reasonable estimates, $\hat{R}_d = 1850$ and $\hat{R}_0 = 6500$. For traffic instances 1 through 5 we generate times of entry into system for each commodity accordingly. Specifically:

Table 3: Volume of railcar traffic in different traffic instances

Volume Type	$\hat{R}_0$	$\hat{R}_d$
Regular	6500	1850
Heavy	7800	2220
Light	5200	1480

1. For the number of commodities equivalent to $\hat{R}_0$ railcars, set the time of entry into the system at the beginning of the horizon.
2. For the number of commodities equivalent to $\hat{R}_d$ railcars, set the time of entry into the system to a random time in day 1 of the horizon. Repeat for all 14 days in the horizon.

Note that since each traffic instance from #1 through #5 assumes a different λ the number of commodities equivalent to an a priori number of railcars varies across traffic instances.

For the remaining ten traffic instances we tweak $\hat{R}_d$ and $\hat{R}_0$ in order to avoid over-reliance on our estimates in the experiments. For traffic instances #6 through #10, we increase each estimator by 20%, setting $\hat{R}_d = 2220$ and $\hat{R}_0 = 7800$. For traffic instances #11 through #15, we decrease each estimator by 20%, setting $\hat{R}_d = 1480$ and $\hat{R}_0 = 5200$. The traffic instances thus simulate 5 regular, 5 heavy, and 5 light examples of network traffic. Table 3 summarizes this setup.

The remaining commodity attributes below are determined independently of the number of railcars and time of entry into system.

Origin and Destination: In order to generate reasonably realistic origin-destination pairs for each commodity, we use the train schedule to inform the probability of any given terminal s being a commodity origin or commodity destination in the following way. Let p_s^{org} represent the probability that the commodity's origin is at s, and p_s^{dest} represent the probability that the commodity's destination is at s. Also let n_s^i represent the weekly number of inbound trains scheduled to s and n_s^o represent the weekly number of outbound trains from s. Note that n_s^o and n_s^i are deterministic functions of the weekly train schedule.

We set $p_s^{org} = \frac{n_s^o}{\sum_{s'} n_{s'}^o} \forall s$ and $p_s^{dest} = \frac{n_s^i}{\sum_{s'} n_{s'}^i} \forall s$. We reason that the higher the number

of inbound trains to s, the more reasonable it is to expect s to be the destination of any given shipment; similar logic is applied to the number of outbound trains from s and the probability of s being the origin of any given shipment.

Trip Plan: We assert that given the origin-destination pair of a commodity and the physical network of the railroad, a reasonable trip plan can be determined by solving a shortest path problem from the origin to the destination using the physical network of the railroad, as follows.

- Let $G_0 = (V_0, A_0)$ represent the original manifest network, where:
  - V_0 is the set of all rail terminals in the manifest network.
  - $\exists$ a directed arc $(i, j) \in A_0$ iff $\exists$ a manifest train in the schedule with train origin i and train destination j.
- Consider any manifest commodity, k. Let $o_k \in V_0$ and $d_k \in V_0$ represent the origin and destination of k, respectively.

Set the trip plan of commodity k to the shortest directed path between o_k and d_k in the physical network, G_0. Recall from section 1.1 that solving this shortest-path problem is trivial.

Due Date: We assert that the due date of a shipment is dependent on the specific shipment's origin, destination, trip plan, and how long one can reasonably expect the trip to take in the rail network. Thus we generate a due date for each shipment as follows:

For each shipment, the simulation is run with that shipment only in the rail network. This provides the hypothetical minimum trip time. We then multiply the minimum trip time by a factor decided a priori; in this case we used a factor of three as a reasonable estimate of the leeway allowed for the shipment's due date. Hence

$$D_k = \tau_k + 3m_k \tag{8}$$

, where D_k is the due date for shipment k, τ_k is the time at which shipment k enters into the network via its origin terminal, m_k is the minimum time required for shipment k to

reach its final destination. Note that Equation 8 adds τ_k, a variable representing a timepoint with $3m_k$, a variable representing an amount of time. In this context addition is assumed to produce the timepoint m_k units of to the future of τ_k.

Each railcar c in shipment k is said to have due date $D_c \triangleq D_k$.

2.5.3 Disruption Instances

Let us now discuss the process used to generate a number of disruption instances, similar to the different traffic instances discussed in section 2.5.2. Our strategy is to generate the disruption instances independently of the 15 traffic instances discussed in section 2.5.2, and run the simulation for all combinations of traffic instances and disruption instances.

Disrupted Stations:

We consider three different stations to disrupt, one of which is disrupted in each disruption instance. In order to obtain comprehensive results, we select one station from each of the Southern, Northern, and Western regions of the rail network; the stations are outlined below.

1. Fort Worth yard, TX
2. Englewood yard, IL
3. Roseville yard, CA

Each station is disrupted for 48 hours toward the beginning of the time horizon, during which time the station processing rate is assumed to drop to zero. Let t_1 and t_2 represent the start and end of the disruption period. Recall that $G = (V, A)$ represents the graph of the time-space network and U_{ij} represents the upper bound of capacity of $(i, j) \in A$. We model the disruption by overriding the U_{ij} of all processing arcs which fall in the disruption period and for the disrupted station. Specifically, we set U_{ij} using Equation (9) $\forall (i, j) \in A$, where $t(i) = t(j)$, and $s(i)$ and $s(j)$ correspond to the disrupted terminal:

$$U_{ij} = \begin{cases} [t_1 - t(i')]\mu_{s(i)}^{t(i'),t_1}, & t(i') < t_1 \quad \& \quad t_1 \leq t(i) \leq t_2 \\ 0, & t_1 \leq t(i') \leq t_2, \quad \& \quad t_1 \leq t(i) \leq t_2 \\ [t(i) - t_2]\mu_{s(i)}^{t_2,t(i)}, & t_1 \leq t(i') \leq t_2 \quad \& \quad t_2 < t(i) \end{cases} \tag{9}$$

, where $i' \in V$ denotes the predecessor of $i \in V$ such that $s(i') = s(i)$.

Setting U_{ij} in this manner ensures the upper bound of capacity of processing arc $(i, j) \in A$ reflects the realistic processing capacity of the corresponding terminal prior to, during, and after the disruption.

Candidate Trains:

Recall that we use L to refer to the set of candidate trains. In reality, the number of trains to consider re-routing in the event of a disruption is typically decided by the operations manager on the ground. To avoid over-reliance of the computational experiments on one particular such decision, we alter the number of candidate trains, $|L|$, across disruption instances. Specifically, for each disrupted terminal we generate 12 disruption instances which vary only by $|L|$. We do this by considering 13 incoming trains, ℓ_1 through ℓ_{13}, which are due to arrive at the disrupted station s during the disruption period. Each of the 12 disruption instances corresponds to a value of i, where we set $L_i = \{\ell_j | 1 \leq j \leq i\} \ \ \forall\ 2 \leq i \leq 13$.

We repeat the above process for each of the three disrupted terminals, resulting in a total of 36 disruption instances. Considering all combinations of traffic instances and disruption instances, we thus produce $15 \times 36 = 540$ overall problem instances to simulate. Furthermore, let S_ℓ represent the set of stations to which train ℓ can be re-routed. Given any problem instance, we ensure $|S_\ell| = 9$ (one of these options is the original destination of ℓ), which we assert is a reasonable number of destinations to consider. In reality, the average number of pragmatic options per train is typically smaller than 9, which would result in a smaller problem instance set size.

2.5.4 Results

In this section, we determine the limits of systematic enumeration. In doing so we need to consider the pragmatic requirements of the original problem. In reality, the decision to re-route trains may need to be made quickly due to the inherent real-time nature of the decision. We assert that in some cases the decision may need to be made as quickly as 15 minutes to 1 hour; therefore in this section we characterize the limitation of systematic enumeration by defining the problem sizes which can be optimally solved within the 15-minute and 1-hour time frames. However, there may also be cases where there is more advance notice regarding significant rail network disruptions. Thus we further examine the impact of having up to 10 hours of time (picked arbitrarily) on the performance of systematic enumeration.

Specifically, we attempt the 540 problem instances defined in Sections 2.5.2 and 2.5.3 using systematic enumeration with 15-minute, 1-hour, and 10-hour computation time cut-offs. The experiments are run on machines with 12-core Xeon E5645 CPUs and 48 GBs of RAM. Given each problem instance, the solutions in the solution space are searched in the (arbitrarily selected) order defined by Algorithm 3.

As motivated above, the following two metrics are used for determining the limits of systematic enumeration:

1. **Time Required:** The range of problem sizes which can feasibly be solved in a reasonable amount of time.
2. **Solution Quality:** The improvement in the objective function value given within reasonable time limits.

Note that the order of solution evaluation specified in Algorithm 3 does not impact the time required, but could potentially impact the solution quality.

Time Required

Note that the average time it takes to solve the problem by systematic enumeration has a linear relationship with the number of solutions in the solution space. Further, the average run time of a single simulation instance can depend on the traffic volume. Recall that traffic

Table 4: Average simulation runtime by traffic volume

Volume	Traffic #	Average Simulation Runtime (seconds)
Regular	1 – 5	16.4
Heavy	6 – 10	17.1
Light	11 – 15	15.0

volume is defined by the number of railcars in the simulation in Table 3. Therefore we find the average simulation runtime for each of the three traffic volumes. Since 180 problem instances are run for each traffic volume, and solving each problem instance involves running $\prod_{\ell \in L} |S_\ell|$ simulations, we assert that the calculation for the average simulation runtime is stable enough for a conclusion about the general limitations of systematic enumeration. Table 4 illustrates the resulting average simulation runtimes for light, regular, and heavy traffic volumes. Based on the values in Table 4, we can infer the problem sizes which are solvable by systematic enumeration within the allowable time, which we illustrate in Figure 16.

Each table in Figure 16 illustrates the problem sizes for which the solution space is small enough for a particular traffic volume. Each cell corresponds to a particular problem size, defined by $|L|$ and $|S_\ell|$, assuming $|S_\ell|$ does not vary $\forall \ell \in L$. The cells in the table are color-coded based on the required time to fully enumerate the solution space. Note the similar color pattern in the three tables suggests that changes in traffic volume within the expected range of $\pm 20\%$ do not have a significant impact on the limitations of systematic enumeration. It is worth calling out that the runtimes in Figure 16 are estimates, calculated by multiplying the size of the solution space for each cell with the relevant average runtime from Table 4.

Solution Quality

Figure 17 visualizes the improvement, in railcar-hours, by traffic instance and $|L|$, averaged across the three different disrupted terminal instances. Note that even a 15-minute time-limit for the decision will result in at least an average of 2000 rail-car hours of reduced delay in the network, with all but one data point indicating over 4000 railcar-hours of reduced delay. While allowing up to an hour of time for the decision does show an improvement in

solution quality, the solution quality seems to stay within the same order of magnitude. This is also true of the results given a 10-hour cutoff, which we show in Figure 18.

Total Time (min) Required to Evaluate All Solutions in Solution Space - Light Traffic Volume

# of Choices per Candidate Train	2	3	4	5	6	7	8	9	10	11	12	13
10	25	250	2,500	25,000	250,000	2,500,000	25,000,000	250,000,000	2,500,000,000	25,000,000,000	250,000,000,000	2,500,000,000,000
9	20	182	1,640	14,762	132,860	1,195,742	10,761,680	96,855,122	871,696,100	7,845,264,902	70,607,384,120	635,466,457,082
8	16	128	1,024	8,192	65,536	524,288	4,194,304	33,554,432	268,435,456	2,147,483,648	17,179,869,184	137,438,953,472
7	12	86	600	4,202	29,412	205,886	1,441,200	10,088,402	70,618,812	494,331,686	3,460,321,800	24,222,252,602
6	9.0	54	324	1,944	11,664	69,984	419,904	2,519,424	15,116,544	90,699,264	544,195,584	3,265,173,504
5	6.3	31	156	781	3,906	19,531	97,656	488,281	2,441,406	12,207,031	61,035,156	305,175,781
4	4.0	16	64	256	1,024	4,096	16,384	65,536	262,144	1,048,576	4,194,304	16,777,216
3	2.3	6.8	20	61	182	547	1,640	4,921	14,762	44,287	132,860	398,581
2	1.0	2.0	4.0	8.0	16	32	64	128	256	512	1,024	2,048
1	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
|L|: # of Candidate Trains												

Total Runtime <= 15 mins
15 mins < Total Runtime <= 1 hour
1 hour < Total Runtime <= 10 hours
Total Runtime > 10 hours

Total Time (min) Required to Evaluate All Solutions in Solution Space - Regular Traffic Volume

# of Choices per Candidate Train	2	3	4	5	6	7	8	9	10	11	12	13
10	27	273	2,733	27,333	273,333	2,733,333	27,333,333	273,333,333	2,733,333,333	27,333,333,333	273,333,333,333	2,733,333,333,333
9	22	199	1,793	16,140	145,261	1,307,345	11,766,104	105,894,934	953,054,403	8,577,489,626	77,197,406,638	694,776,659,743
8	17	140	1,120	8,957	71,653	573,222	4,585,772	36,686,179	293,489,432	2,347,915,455	18,783,323,641	150,266,589,129
7	13	94	656	4,594	32,157	225,102	1,575,712	11,029,986	77,209,901	540,469,310	3,783,285,168	26,482,996,178
6	9.8	59	354	2,125	12,753	76,516	459,095	2,754,570	16,527,421	99,164,529	594,987,172	3,569,923,031
5	6.8	34	171	854	4,271	21,354	106,771	533,854	2,669,271	13,346,354	66,731,771	333,658,854
4	4.4	17	70	280	1,120	4,478	17,913	71,653	286,611	1,146,443	4,585,772	18,343,089
3	2.5	7.4	22	66	199	598	1,793	5,380	16,140	48,420	145,261	435,782
2	1.1	2.2	4.4	8.7	17	35	70	140	280	560	1,120	2,239
1	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.27
|L|: # of Candidate Trains												

Total Runtime <= 15 mins
15 mins < Total Runtime <= 1 hour
1 hour < Total Runtime <= 10 hours
Total Runtime > 10 hours

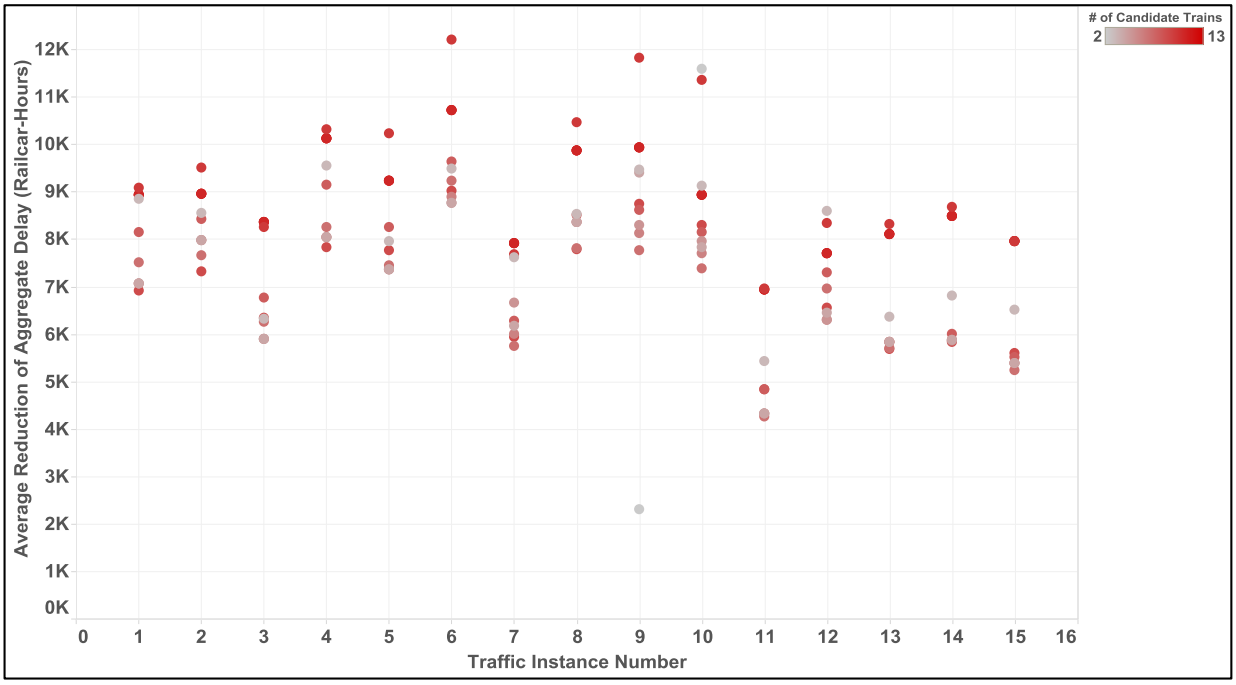
Total Time (min) Required to Evaluate All Solutions in Solution Space - Heavy Traffic Volume

# of Choices per Candidate Train	2	3	4	5	6	7	8	9	10	11	12	13
10	29	285	2,850	28,500	285,000	2,850,000	28,500,000	285,000,000	2,850,000,000	28,500,000,000	285,000,000,000	2,850,000,000,000
9	23	208	1,870	16,829	151,461	1,363,146	12,268,315	110,414,839	993,733,554	8,943,601,989	80,492,417,897	724,431,761,074
8	18	146	1,167	9,339	74,711	597,688	4,781,507	38,252,052	306,016,420	2,448,131,359	19,585,050,870	156,680,406,958
7	14	98	684	4,790	33,530	234,710	1,642,968	11,500,778	80,505,446	563,538,122	3,944,766,852	27,613,367,966
6	10	62	369	2,216	13,297	79,782	478,691	2,872,143	17,232,860	103,397,161	620,382,966	3,722,297,795
5	7.1	36	178	891	4,453	22,266	111,328	556,641	2,783,203	13,916,016	69,580,078	347,900,391
4	4.6	18	73	292	1,167	4,669	18,678	74,711	298,844	1,195,377	4,781,507	19,126,026
3	2.6	7.7	23	69	208	623	1,870	5,610	16,829	50,487	151,461	454,382
2	1.1	2.3	4.6	9.1	18	36	73	146	292	584	1,167	2,335
1	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29
|L|: # of Candidate Trains												

Total Runtime <= 15 mins
15 mins < Total Runtime <= 1 hour
1 hour < Total Runtime <= 10 hours
Total Runtime > 10 hours

Figure 16: Time limitation of systematic enumeration by problem instance size

Improvement of Systematic Enumeration with 15-Minute Cutoff Over "Do-Nothing" Solution



Improvement of Systematic Enumeration with 1-Hour Cutoff Over "Do-Nothing" Solution

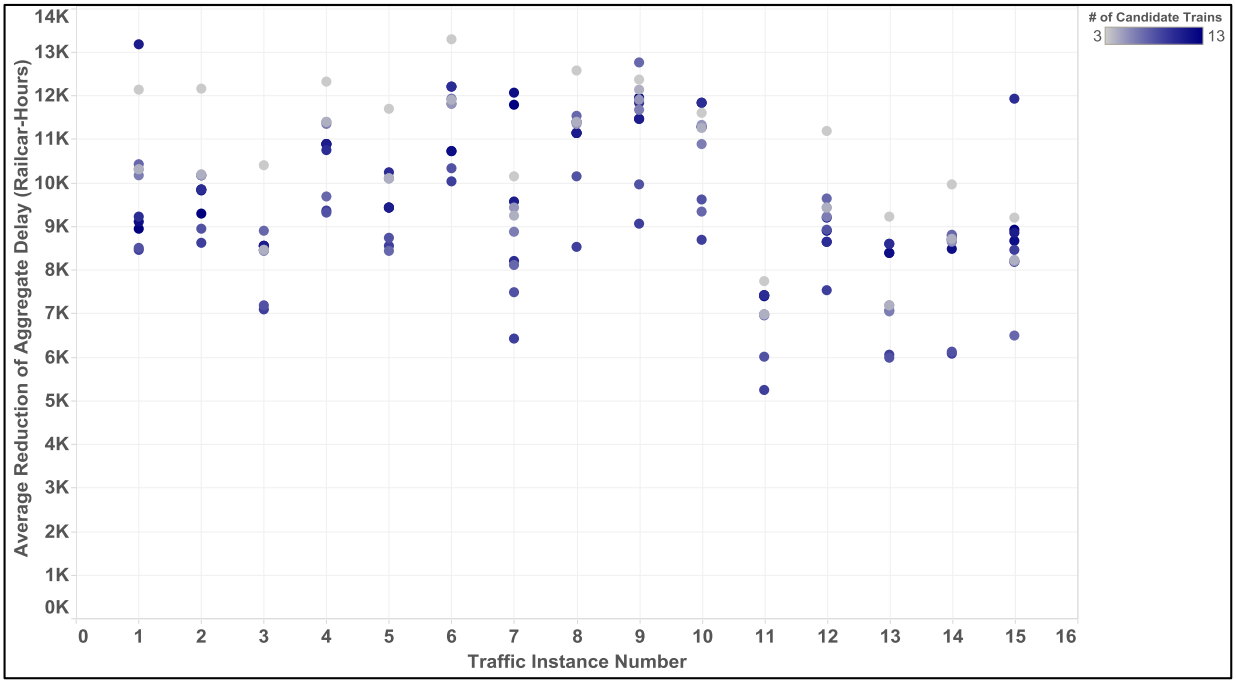


Figure 17: Improvement in objective value of best systematic enumeration solution compared to the original option of re-routing no candidate trains

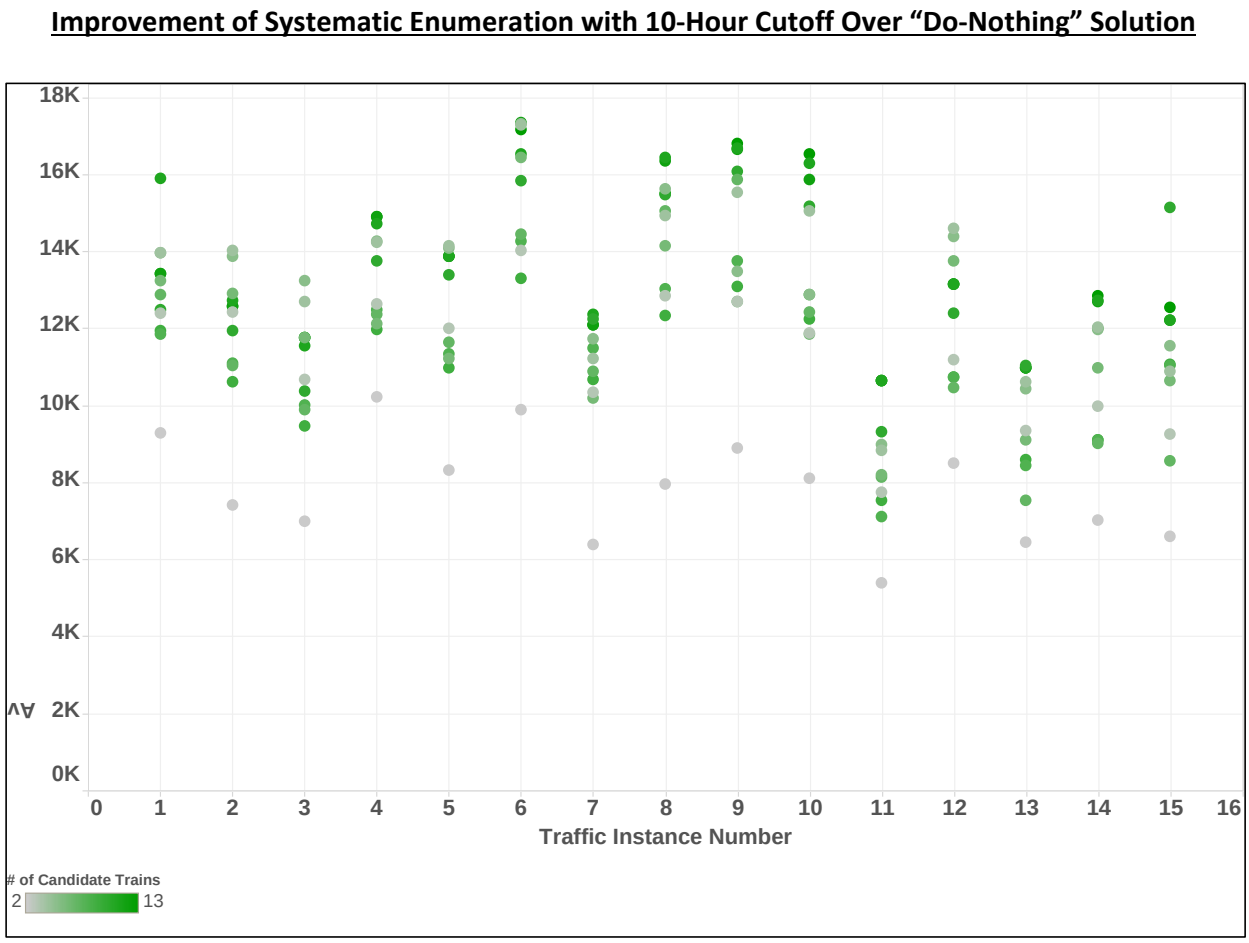


Figure 18: Improvement in objective value of best systematic enumeration solution compared with the original option of re-routing no candidate trains

2.6 Conclusion

We have introduced a time-space network modeling approach and a railcar movement simulation framework as a way to mathematically evaluate the operational policies of a commercial railroad. This has allowed us to quantify the limitations of using systematic enumeration to solve the problem of dynamically re-routing freight trains in the event of a significant disruption to the rail network. The limitations we have quantified motivate the next chapter, which explores novel approaches using the methodologies introduced in this chapter as a building block.

2.7 Future Work

The systematic enumeration procedure described in Algorithm 3 uses a specific search order. Exploring variations in the search order would be an interesting area of further computational experimentation. Possible search strategies to attempt can include randomized sampling or techniques such as latin hypercube sampling, one variance of which is explored by Florian [16].

Chapter *III*: Quantitative Methods for Modeling Freight Train Re-Routing Problems with Large Solution Space

3.1 Introduction

Recall the train re-routing problem from Chapter *II*: Given a set of candidate trains L, we are to determine a potentially new destination rail terminal for each $\ell \in L$. In Chapter *II*, we demonstrated the ability to solve problem instances with a reasonably small solution space using the developed framework. In this chapter we shift our focus to real-world problem instances, where there may be several candidate trains in L and several re-route options for each candidate train. Note that the solution space increases exponentially in size with the number of candidate trains: If S_ℓ is the set of potential re-route rail terminals for train ℓ, the solution space contains $\prod_{\ell \in L} |S_\ell|$ potential joint decisions. This warrants an extended solution methodology that remains tractable given the potential problem instances of interest. In this chapter we extend the work of the previous chapter in two ways:

1. We develop an exact-optimization approach that uses the time-space network developed in Chapter *II*. The approach we develop exploits the a priori nature of the commodity trip plans in order to maintain tractability.

2. We develop a search-based heuristic approach that uses the railcar movement simulation framework developed in Chapter *II*.

3.2 Literature Review

A significant part of our work in this chapter involves formulating a special minimum-cost multi-commodity network flow problem that we combine with a network design problem methodology. Both problems have previously received attention in the existing literature. The minimum-cost multi-commodity network flow problem has been studied for decades. Tomlin [34] formulates the general problem as a large integer program whose special structure

permits efficient solution computation. Foulds [17] studies a minimum-cost multi-commodity network design problem where any set of arcs can be removed from the network. He presents a solution procedure based on a branch-and-bound enumeration methodology with which he is able to determine a set of arcs to remove from the network in order to ensure all feasible flows are minimal.

A network design formulation problem can be thought of as a multi-commodity network flow problem with additional binary variables introduced to represent design decisions: Given a graph $G = (V, A)$, where V is the set of nodes and A is the set of arcs, a set of commodities K, and b_i^k units supply of commodity $k \in K$ entering at node $i \in V$:

$$
\begin{aligned}
\min \quad & \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij} \\
\text{s.t.} \quad & \sum_{j \in V} x_{ij}^k - \sum_{j \in V} x_{ji}^k = b_i^k \qquad \forall\ i \in V, k \in K \\
& \sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij} \qquad \forall\ (i,j) \in A \\
& y_{ij} \in 0, 1 \qquad \forall\ (i,j) \in A \\
& x_{ij}^k \geq 0 \qquad \forall\ (i,j) \in A, k \in K
\end{aligned}
$$

, where c_{ij}^k is the cost of flowing one unit of commodity $k \in K$ along arc $(i,j) \in A$, f_{ij} is the fixed cost of opening arc $(i,j) \in A$ for flow of commodities.

Crainic discusses an extension of the above formulation where there is a finite budget for the aggregate fixed costs of opening arcs with the following additional constraint:

$$\sum_{(i,j) \in A} f_{ij} y_{ij} \leq B.$$

He furthermore discusses the generalization of the problem where $y_{ij} \in N$, where each non-zero value represents a different level of service offered at arc $(i,j) \in A$ [10].

In addition, in this chapter we explore an alternate solution methodology using a variant of the combinatorial local search called *Tabu search*. In the most canonical form, combinatorial local search begins by selecting an initial feasible solution to the problem $x \in X$, where X represents the feasible region. The algorithm then iteratively defines a *neighborhood* for the selected solution, $N(x)$, and looks for a new solution to select in the neighborhood [23] – as described in algorithm 4.

Algorithm 4 Simple local search procedure

Requires: Neighborhood definition N.

Step 1: Generate and select $x \in X$.

Step 2: Select some $x' \in N(x)$ such that $z(x') < z(x)$, where $z(x)$ represents the objective function value corresponding to x.

If $\nexists$ such x', stop and report x as the local optimum.

Step 3: Set $x \leftarrow x'$ and go to Step 2.

In Chapter I we discussed the intermodal terminal location problem formulated by Arnold et al. [3]. Sorensen et al. [33] uses local search to propose two metaheuristics for solving the mixed-integer programming model of Arnold et al. [3]. Each metaheuristic first constructs a set of initial solutions, which are subsequently perturbed in search of a better solution using local search. The first metaheuristic constructs the initial solutions using greedy randomized adaptive search procedure (GRASP), while the second metaheuristic does the same using an attribute-based hill climber method.

Glover [18] studies the Tabu search variant of local search in detail. The primary motivation behind the technique is continuing the search beyond a local optimum and avoiding coming back to that optimum in subsequent iterations. The stopping criteria is typically set to one or both of the following:

1. A fixed number of iterations is completed.

2. No improvement compared to the best solution so far has been found after a fixed number of iterations.

The simple Tabu search algorithm is thus summarized in algorithm 5.

Algorithm 5 Simple Tabu search procedure

Requires:

Neighborhood definition N

Maximum iterations allowed M

Step 1: Generate and select $x \in X$.

Step 2: Initialize the following.

The best solution found $x^* \leftarrow x$.

Iteration counter $i \leftarrow 0$.

The Tabu set $T \leftarrow \{\}$.

Step 3: If $N(x) - T = \{\}$ or $i = M$:

Stop and report x^* as the best solution found.

Else:

$i \leftarrow i + 1$.

Select x' such that $z(x') <= z(y)\ \forall\ y \in N(x) - T$.

Update T according to the Tabu rule.

If $z(x') < z(x^*)$, then set $x^* \leftarrow x'$.

Go back to the beginning of Step 3.

Glover [18] introduces an example of Tabu rule where a constant t is selected and the Tabu set T is chosen such that the changes in the solution made in the previous t iterations cannot to be reversed.

There have been examples in the literature of both network design optimization and Tabu search being applied to service network design problems. For instance, Pedersen et al. [27] use a discretized time-space network formulation to model a service network design problem as an arc-based formulation of a multi-commodity network design problem with additional design balance constraints of the form:

$$\sum_{j \in V} y_{ij} - \sum_{j \in V} y_{ji} = 0 \qquad \forall\ i \in V$$

They then solve the problem using Tabu search. The initial solution is obtained by solving the linear relaxation of the multi-commodity network design problem and rounding up the

design variables, thus removing the guarantee that the initial solution will be feasible. In contrast to Glover [18], which sets the maximum number of iterations as the stopping criteria, Pedersen et al. [27] set a time limit for that purpose.

Crainic et al. [12] develops a simplex-based Tabu search method for solving the path-based formulation of the capacitated multi-commodity network design problem. The study notes that the set of feasible path flows forms a path-based minimum-cost network flow polygon, one of the extreme points of which is the optimal solution. Noting the difficulty of solving large or moderate problem sizes, the paper introduces a Tabu search method where the neighbors of an extreme point are defined as the set of adjacent extreme points reachable by one simplex pivot. The algorithm also performs an a priori number of diversification moves, and retries the Tabu search algorithm after each diversification move.

3.3 Methods: Optimization-based Solution Approach

Recall the time-space network developed in Chapter *II*. In this section we introduce an optimization-based solution method that solves a relaxation of the train re-routing problem using a service network design optimization framework. The re-routing solution proposed by the optimization framework is then evaluated using the railcar movement simulation methodology introduced in Chapter *II* (Algorithm 1); the evaluation step ensures the objective function of the proposed re-routing solution reflects realistic railcar flows following the re-routing decision. The following steps describe the approach:

1. **Solving a Relaxation Problem:** Using the time-space network allows we frame an arc-based, linear multi-commodity network flow problem with a network design extension to solve for the re-routing decisions as well as the optimal railcar flow in the time-space network. We refer to this framework as a multi-commodity network design problem. Using the structure of the re-routing problem we can ensure the multi-commodity network design problem is tractable for realistic problem instances.

2. **Solving the Train Re-Routing Problem for the Proposed Solution:** Since optimizing the rules of how railcars are processed is outside the scope of this problem, we evaluate the re-routing decisions recommended by the previous step using the railcar movement simulation, which calculates the resulting aggregate delay of the proposed re-routing solution under the set of rules used by the railroad.

Subsections 3.3.1, 3.3.2, and 3.3.3 describe the multi-commodity network design formulation which is used to solve the relaxation problem. Subsection 3.3.4 describes how the relaxation solution is evaluated for the original train re-routing problem.

3.3.1 Definitions

In order to describe a multi-commodity network design problem on the time-space network, we will use the following:

G: the directed time-space network, as defined in Chapter *II*.
V: the set of nodes in G
A: the set of arcs in G
K: the set of all commodities in the network during the selected time horizon
b_i^k: supply, measured in number of railcars, of commodity k at node $i \in V$
$\delta^{out}(i)$: set of directed arcs leaving $i \in V$
$\delta^{in}(i)$: set of directed arcs entering $i \in V$
L_p: set of dispatched trains to be re-routed
L_f: set of trains not yet dispatched to be re-routed
S_ℓ: set of terminals to which train ℓ can be re-routed
o^k: time-space node where commodity k originates in the train network
$i(\ell)$: time-space train departure node for train ℓ
$j(\ell, s)$: time-space train arrival node for train ℓ given it is re-routed to rail terminal $s \in S_l$
A^k: the subset of arcs in G "relevant" to commodity $k \in K$, determined by Algorithm 6.
$K_{(i,j)}$: the subset of commodities "relevant" to arc $(i, j) \in A$
c_{ij}^k: cost of flowing one railcar of commodity $k \in K$ along arc $(i, j) \in A^k$
$t(i)$: the point of time that corresponds to time-space node $i \in V$
$\tau(k)$: the time at which commodity k is introduced to the physical network

The decision variables for the integer programming model are as follows.

x_{ij}^k: number of railcars of commodity k flowing along time-space arc $(i,j) \in A^k$
$y_{\ell s}$: binary variable that indicates whether train ℓ is to be re-routed to rail terminal s

$$y_{\ell s} = \begin{cases} 1 & \text{if train } \ell \text{ is re-routed to rail terminal } s; \\ 0 & \text{otherwise.} \end{cases}$$

Note that each $y_{\ell s}$ corresponds to a unique train arc.

3.3.2 Use of Subsets

To manage the problem size we use $A^k \subset A$ or the relevant subset of time-space arcs for commodity k. This helps ensure tractability by creating the x_{ij}^k variables only for $(i,j) \in A^k \quad \forall k \in K$. For each $k \in K$, we determine A^k by utilizing the a priori nature of the trip plan of k, detailed in the following algorithm.

Algorithm 6 Determining the relevant time-space arc set for commodity k.

Require: Set of time-space arcs A.

Require: Trip plan of commodity k.

Require: If k is on a candidate train, potential alternate trip plans of k for each re-route option.

Step 0: Define Φ_k as the union set of trip plan legs for commodity k, including legs from potential alternate trip plans.

Step 1: Set $A^k = A$. Mark all $(i,j) \in A^k$ as unchecked.

Step 2: Pick an unchecked arc $(i,j) \in A^k$. If there is none, terminate the algorithm and output A^k.

Step 3: If (i,j) is not a train arc skip this step. If (i,j) is a train arc and $[s(i) \to s(j)] \in \Phi_k$, mark (i,j) as checked. Otherwise remove (i,j) from A^k. Go to Step 2.

Step 4: If (i,j) is not an exit arc skip this step. If (i,j) is an exit arc and $s(i)$ is the final rail terminal in the trip plan of k, mark (i,j) as checked. Otherwise remove (i,j) from A^k. Go to Step 2.

Step 5: If $s(i)$ is a rail terminal in any leg in Φ_k, mark (i,j) as checked. Otherwise remove (i,j) from A^k. Go to Step 2.

Furthermore, for each $(i,j) \in A$, we set $K_{(i,j)}$ to the set of commodities for which the relevant arc-set involves (i,j). Using A^k and $K_{(i,j)}$ instead of A and K as appropriate in

the multi-commodity network design problem significantly reduces the number of integer variables in the optimization model.

3.3.3 Data and Optimization Formulation

Prior to formulating the integer program, we will discuss the data it uses.

Let C_k denote the set of cars in commodity k. Set b_i^k as follows:

$$b_i^k = \begin{cases} |C_k| & \text{if original rail terminal in trip plan of } k \text{ is } s(i),\ \tau(k) \leq t(i), \text{ and } \tau(k) > t(i'); \\ 0 & \text{otherwise,} \end{cases}$$

i' is the predecessor node of i that corresponds to the same physical location.
Furthermore, for each time-space arc (i, j), set c_{ij}^k as follows:

- If (i, j) is an exit arc, set $c_{ij}^k = max[0, t(i) - D_k]$, where D_k is the time at which commodity k is due at its final destination rail terminal.
- If (i, j) is a feasibility arc, the delay cost must reflect two components:
  - *Future Delay:* A reasonable approximation is needed for the delay commodity k can be expected to incur in the future. Let P represent a reasonable penalty for the average commodity time through the network. We approximate future delay by calculating a *penalty ratio* for railcar c as follows:

$$p_c = \frac{q_{s(i)}}{q} \tag{10}$$

    , where q is the number of legs in the trip plan of commodity k, and $q_{s(i)}$ is the number of legs in the trip plan of k up to rail terminal $s(i)$.
  - *Incurred Delay:* If D_k is an earlier time than $t(i)$, an additional delay of $t(i) - D_k$ is already incurred.

To account for both components, we set $\rho_{ij}^k = max[t(i) - D_k, 0] + p_k P$.

- If (i, j) is a terminal storage arc, train arc, or processing arc, set $c_{ij}^k = 0$.

We now formulate the mixed integer program (MIP) as follows:

$$\max \quad \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij}^k x_{ij}^k \tag{11}$$

$$\text{s.t.} \quad \sum_{s \in S_\ell} y_{\ell s} = 1 \qquad \forall\ \ell \in\ L_f \cup L_p \tag{12}$$

$$x_{i(\ell)j(\ell,s)}^k \leq b_{o^k}^k y_{\ell s} \qquad \forall l \in L_f, \forall s \in S_\ell, \forall k \in K \tag{13}$$

$$x_{i(\ell)j(\ell,s)}^k = b_{o^k}^k y_{\ell s} \qquad \forall l \in L_p, \forall s \in S_\ell, \forall k \in K \tag{14}$$

$$\sum_{k \in K_{(i,j)}} x_{ij}^k \leq U_{ij} \qquad \forall (i,j) \in A \tag{15}$$

$$\sum_{j \in \delta^{out}(i)} x_{ij}^k - \sum_{j \in \delta^{in}(i)} x_{ij}^k = b_i^k \qquad \forall k \in K, \forall i \in V \tag{16}$$

$$x_{ij}^k \geq 0 \qquad \forall k \in K, \forall (i,j) \in A^k \tag{17}$$

$$y_{\ell s} \in \{0, 1\} \qquad \forall \ell \in L, \forall s \in S_\ell \tag{18}$$

(1) is the objective function, which minimizes aggregate railcar delay. (2) ensures exactly one re-route option is selected for each train ℓ. (3) and (4) ensure the flow variables x_{ij}^k are only allowed to be non-zero for a re-route option which is selected; (3) does this for candidate trains that are not yet dispatched, while (4) forces this for candidate trains that are already dispatched. The difference is that if a train is already dispatched, the flow of the commodities on the train given a re-route decision is forced with an equality constraint; however, if a train is to be dispatched in the future we do not force any particular commodity on the train to allow the IP to decide which commodities to load onto the train for optimal network performance. (5) outlines the bundling constraints. (6) outlines the flow-balance constraints.

3.3.4 Re-Route Solution Evaluation

Once a re-route solution is determined using the multi-commodity network design relaxation problem, it is to be evaluated by running the railcar movement simulation methodology introduced in Chapter *II*. In this subsection we describe how the simulation parameters are set to achieve this goal. Recall the binary variables in the multi-commodity network design formulation which specify the selected re-route options:

$$y_{\ell s} = \begin{cases} 1 & \text{if train } \ell \text{ is re-routed to rail terminal } s; \\ 0 & \text{otherwise.} \end{cases}$$

Solving the relaxation problem obtains solution values for these re-route variables. The next step is to evaluate the re-route solution by solving the corresponding railcar movement simulation introduced in Chapter *II*. We do this by running Algorithm 2 from the previous chapter, which, given a specified solution to the train re-routing problem, $\phi = (\phi_1, \phi_2, ..., \phi_{|L|})$, sets the corresponding arc capacities in the time-space network and executes the corresponding simulation.

Algorithm 7 Setting ϕ based on the output of the optimization model

Require: $y_{\ell,s}\ \forall \ell \in L, \forall s \in S_\ell$.
Step 1: Set ϕ.
For each ℓ in $(1, 2, ..., |L|)$:
Set $\phi_\ell = s$ if $y_{\ell,s} = 1$.
Step 2: Run Algorithm 2 using the constructed ϕ as input.

Algorithm 2, which is called in the second step sets the upper bounds of the train re-routing arcs in the time-space network to correspond to the solution prescribed by the y output variables, and subsequently calls the simulation algorithm.

3.4 Methods: Search-Based Heuristic Approach

In this section we introduce an alternative methodology to solve the large-scale re-routing problem introduced in this chapter. Recall that L is the set of candidate trains and S_ℓ denotes the set of terminals to which train $\ell \in L$ can be re-routed. Let Φ denote the set of all possible solutions and note that $|\Phi| = \prod_{\ell \in L} |S_\ell|$. Recall from subsection 3.3.4 that any particular solution $\phi \in \Phi$ is characterized by specifying a destination terminal for each candidate train: $\phi = (\phi_1, \phi_2, ..., \phi_{|L|})$. We also define the *distance* between two solutions, $\nu(\phi^1, \phi^2)$ to represent the number of trains that are routed to different rail terminals under ϕ^1 and ϕ^2. Note that $\nu(\phi, \phi) = 0$, $\forall \phi \in \Phi$.

We now discuss a heuristic solution procedure that applies a Tabu search algorithm using the railcar simulation framework introduced in Chapter *II*.

3.4.1 Neighborhood Definition and Tabu Rule

Given solution $\phi \in \Phi$, define the neighborhood of ϕ as follows:

$$N(\phi) = \{\widetilde{\phi} | \nu(\widetilde{\phi}, \phi) = 1\}. \tag{19}$$

Using this neighborhood definition, we apply the following local search algorithm with a Tabu rule added:

Algorithm 8 Tabu Local Search Algorithm

Step 1: Set $\phi^0 = (\phi_1^0, \phi_2^0, ..\phi_{|L|}^0)$, where ϕ_ℓ^0 is the original destination terminal for train ℓ. Set $T = \emptyset$, $\phi_{local}^* = \phi^0$, $\phi_{global}^* = \phi^0$, and $z_{global}^* = z(\phi^0)$.
Step 2: Set $z_{local}^* = \infty$ and $\phi_{prev} = \phi_{local}^*$. For each $\phi \in N(\phi_{local}^*) \backslash T$:

- Compute $z(\phi)$ using Algorithm 2.
- If $z(\phi) < z(\phi_{local}^*)$, set $\phi_{local}^* = \phi$ and $z_{local}^* = z(\phi)$.
- If $z(\phi) < z(\phi_{global}^*)$, set $\phi_{global}^* = \phi$ and $z_{global}^* = z(\phi)$.

Step 3: Set $T = \{\phi | \nu(\phi, \phi_{prev}) \leq 1\} \cap \{\phi | \nu(\phi, \phi_{local}^*) = 1\}$
Step 4: If time elapsed has passed time threshold, stop. Otherwise, go to step 2.

Note that setting $z^*_{local} = \infty$ in Step 2 ensures the search heuristic continues even if the best solution found in the neighborhood results in a worse objective function value.

3.5 Computational Results

So far in this chapter we have introduced two methods for solving the problem originally introduced in Chapter *II*. In this section we examine the efficacy of each approach. In order to maintain consistent evaluation criteria, the 540 problem instances introduced in Sections 2.5.2 and 2.5.3 of Chapter *II* and solved in Section 2.5.4 are used again in this section. Furthermore, recall that the results shown in Section 2.5.4 were obtained by running the experiments on machines with 12-core Xeon E5645 CPUs and 48 GBs of RAM. The same machines are used to obtain the results in this section.

3.5.1 Results for Optimization-Based Approach

In Chapter *II* we used the following two metrics to evaluate the efficacy of the proposed approach: (1) time required, and (2) solution quality. We will evaluate the optimization-based approach using the same criteria.

Time Required

Note that, unlike systematic enumeration in Chapter *II*, the solution time of the optimization formulation is highly sensitive to the number of commodities in the network over the time horizon of interest. This is because the optimization formulation assigns a flow decision variable x^k_{ij} per $(i, j) \in A$ and $k \in K$. The variation of problem size across the 15 traffic instances in Figure 19 illustrates the sensitivity of problem size to the number of commodities in the network.

	Traffic Instance Number														
|L|	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	11.95	16.37	18.66	21.52	24.19	15.01	19.68	22.69	25.55	28.53	10.40	13.60	15.51	16.98	19.43
3	12.04	16.46	18.76	21.62	24.29	15.11	19.78	22.79	25.65	28.63	10.50	13.70	15.61	17.07	19.53
4	12.15	16.57	18.86	21.73	24.39	15.21	19.88	22.90	25.76	28.74	10.60	13.80	15.71	17.18	19.63
5	12.25	16.67	18.97	21.83	24.50	15.32	19.99	23.00	25.86	28.85	10.70	13.90	15.82	17.28	19.74
6	12.36	16.78	19.08	21.94	24.61	15.43	20.10	23.11	25.97	28.96	10.81	14.01	15.93	17.39	19.85
7	12.46	16.89	19.18	22.05	24.71	15.53	20.20	23.22	26.08	29.06	10.91	14.11	16.03	17.49	19.95
8	12.57	16.99	19.29	22.16	24.82	15.64	20.31	23.33	26.19	29.17	11.02	14.22	16.14	17.60	20.06
9	12.68	17.11	19.40	22.27	24.94	15.75	20.42	23.44	26.30	29.29	11.13	14.33	16.25	17.72	20.17
10	12.79	17.22	19.51	22.38	25.05	15.86	20.54	23.56	26.42	29.40	11.24	14.44	16.36	17.83	20.29
11	12.89	17.32	19.62	22.49	25.16	15.97	20.64	23.66	26.53	29.51	11.34	14.55	16.47	17.93	20.39
12	13.01	17.44	19.74	22.61	25.28	16.08	20.76	23.78	26.64	29.63	11.46	14.66	16.58	18.05	20.51
13	13.13	17.56	19.86	22.73	25.40	16.20	20.88	23.90	26.77	29.75	11.58	14.78	16.70	18.17	20.63

Figure 19: Average number of variables in optimization formulation (millions)

The variability observed in problem size in turn creates variability in the required time to solve the problem using the optimization-based approach. Figure 20 reports the solve time for each traffic instance and $|L|$. The results indicate that despite the relatively large problem sizes at hand, the problem is solvable well within an hour, while the majority of the problem instances are solvable within 30 minutes.

	Traffic Instance Number														
|L|	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	5.44	15.82	18.41	23.30	23.40	13.86	22.03	22.99	24.64	27.04	4.37	6.18	7.04	16.39	19.24
3	5.82	15.89	18.59	20.25	22.87	14.15	19.33	21.89	24.38	27.18	4.77	6.98	7.02	16.47	19.17
4	5.91	15.90	18.79	20.70	22.51	14.30	19.46	22.28	25.25	27.20	5.04	7.08	7.31	16.70	19.24
5	6.29	16.03	18.99	21.03	23.04	14.23	19.83	22.29	24.55	27.07	5.17	7.17	7.35	16.64	19.78
6	6.90	16.31	18.22	21.41	23.24	14.78	19.19	22.13	26.07	27.20	5.47	6.83	7.99	16.84	19.47
7	6.71	16.31	18.13	20.73	23.60	14.89	19.39	21.83	26.03	26.56	5.20	7.15	9.20	17.05	19.78
8	6.90	16.39	18.26	21.87	24.09	14.65	24.38	21.76	26.50	28.65	5.72	7.15	9.20	17.22	19.93
9	7.01	16.25	18.56	21.45	24.47	14.85	23.42	21.75	26.44	38.53	5.41	7.62	8.57	17.46	19.83
10	6.96	16.52	18.96	21.76	25.13	14.89	24.01	22.30	27.01	35.67	6.62	7.51	8.94	16.82	19.96
11	7.11	16.35	19.03	21.84	25.26	15.12	23.24	21.48	27.08	44.98	6.01	7.65	10.19	17.00	20.01
12	7.11	16.10	18.85	22.04	25.54	14.82	22.72	21.47	26.27	44.15	6.48	7.79	10.10	17.17	19.42
13	7.80	16.16	18.33	21.60	24.82	15.23	24.19	22.63	27.73	45.71	6.79	8.08	9.70	16.79	18.77

Figure 20: Optimization problem solve time (minutes) heatmap

Note that since the developed IP assigns flow variables to commodities instead of indi-

vidual railcars, the number of commodities (and not necessarily the number of railcars) has a direct impact on solution time. Given a fixed number of railcars, the IP size is bigger if the average commodity size is smaller, with 1 being the worst-case scenario. The impact on solution time seems to be particularly great if average commodity size is set to 1 for commodities on candidate trains. Also note that the solution time given this set-up increases significantly when increasing solution space size from 10^3 to 10^4. However, it drops significantly when the "worst-case scenario" assumption of average commodity size 1 on candidate trains is removed.

Solution Quality

Recall that for problem instances where $|L| \leq 3$, the systematic enumeration algorithm described in Chapter *II* completes prior to the 10-hour cutoff time. Thus, from Chapter *II* results we know the optimal solution for such problem instances and can assess the optimality gap for any other proposed solution. Figure 21 illustrates the optimality gap associated with the solution obtained from the optimization-based approach where $|L| \leq 3$.

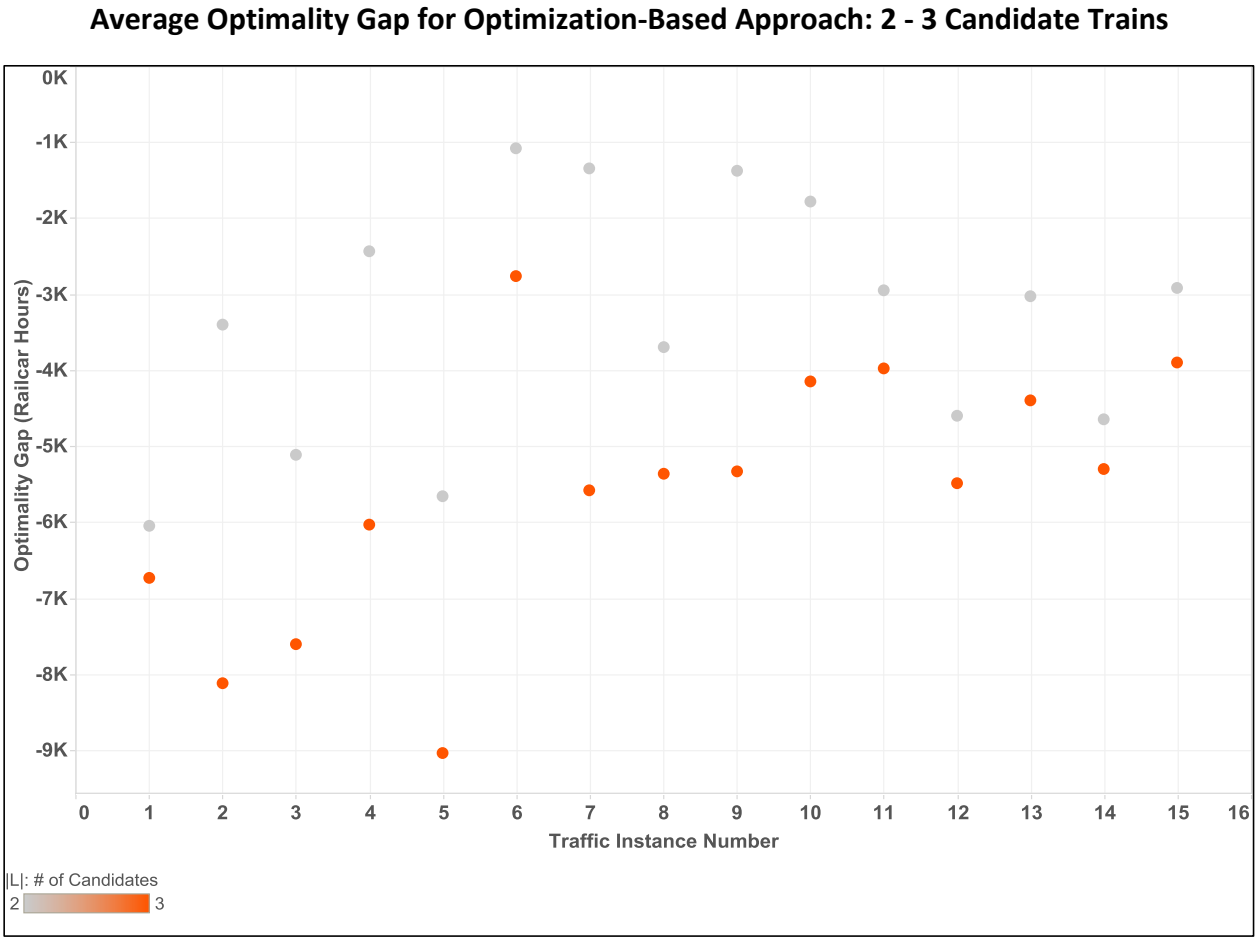


Figure 21: Difference between solution of optimization-based approach and optimal solution: $|L| \leq 3$

In addition to comparing to the optimal solution, let us compare the results of the optimization-based approach with those of systematic enumeration. Since the optimization solution time is within an hour we assert that the systematic enumeration solution obtained at the 1-hour cutoff should be used for the comparison. Figure 22 provides this comparison. Notice that the optimization-based approach provides the better-quality solution for problem instances with higher $|L|$. This is reasonable to see, since as $|L|$ increases systematic enumeration is able to search an exponentially smaller subset of the solution space.

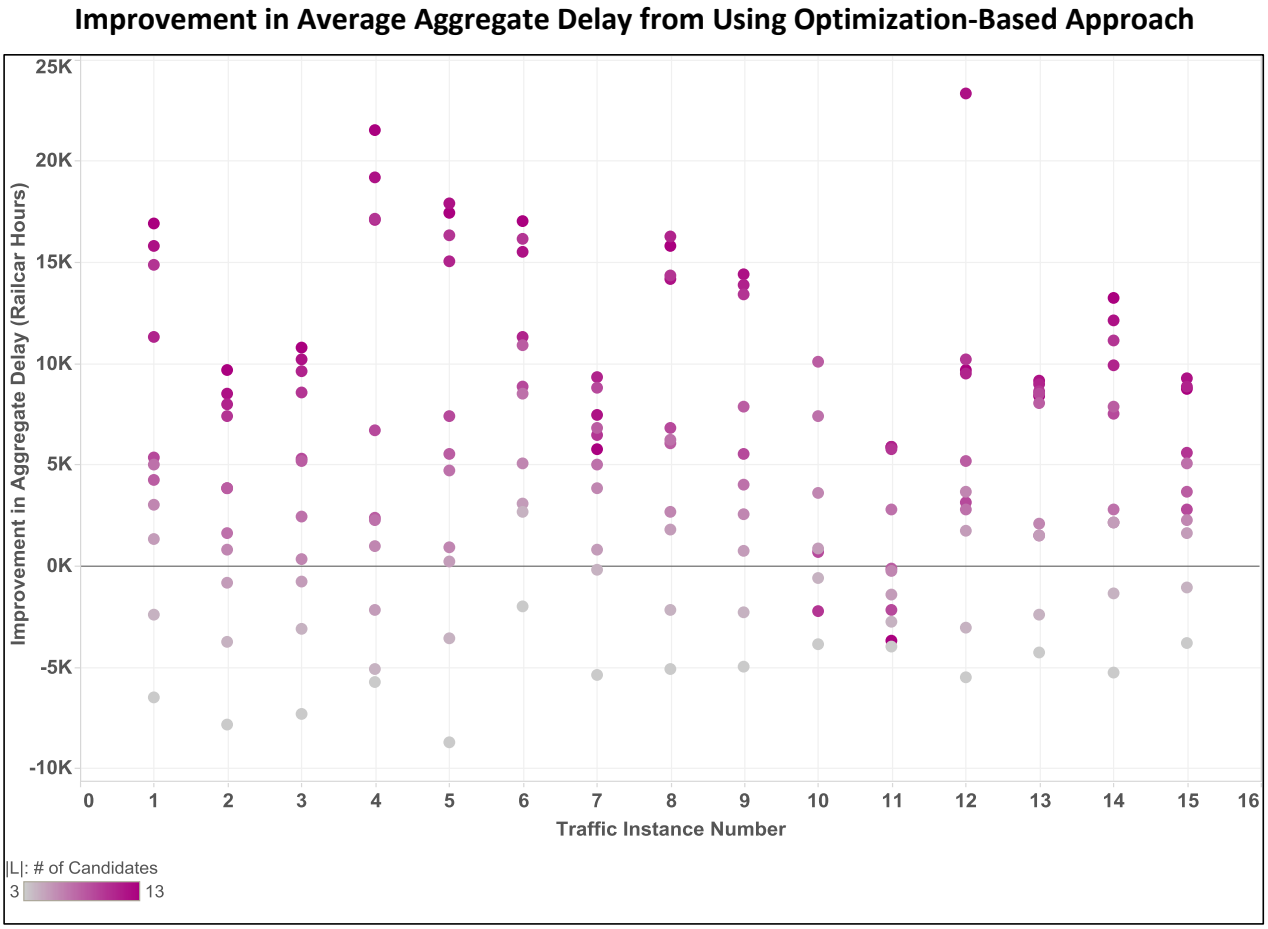
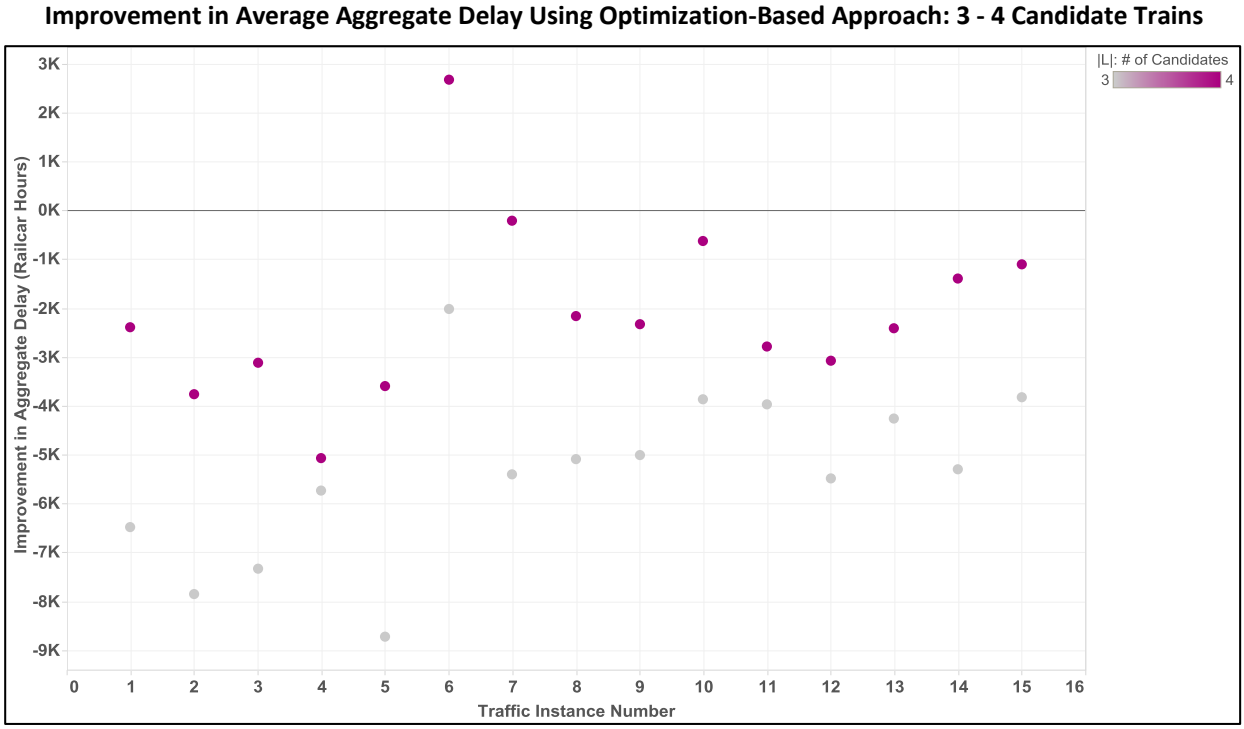


Figure 22: Objective function improvement of optimization-based approach, with systematic enumeration solution after 1 hour as baseline

Figure 23 further illustrates the increasingly superior performance of the optimization-based approach as $|L|$ increases. In addition, we notice that $|L| \geq 5$ seems to be the "breaking point" where systematic enumeration is no longer the recommended method for solving the problem.



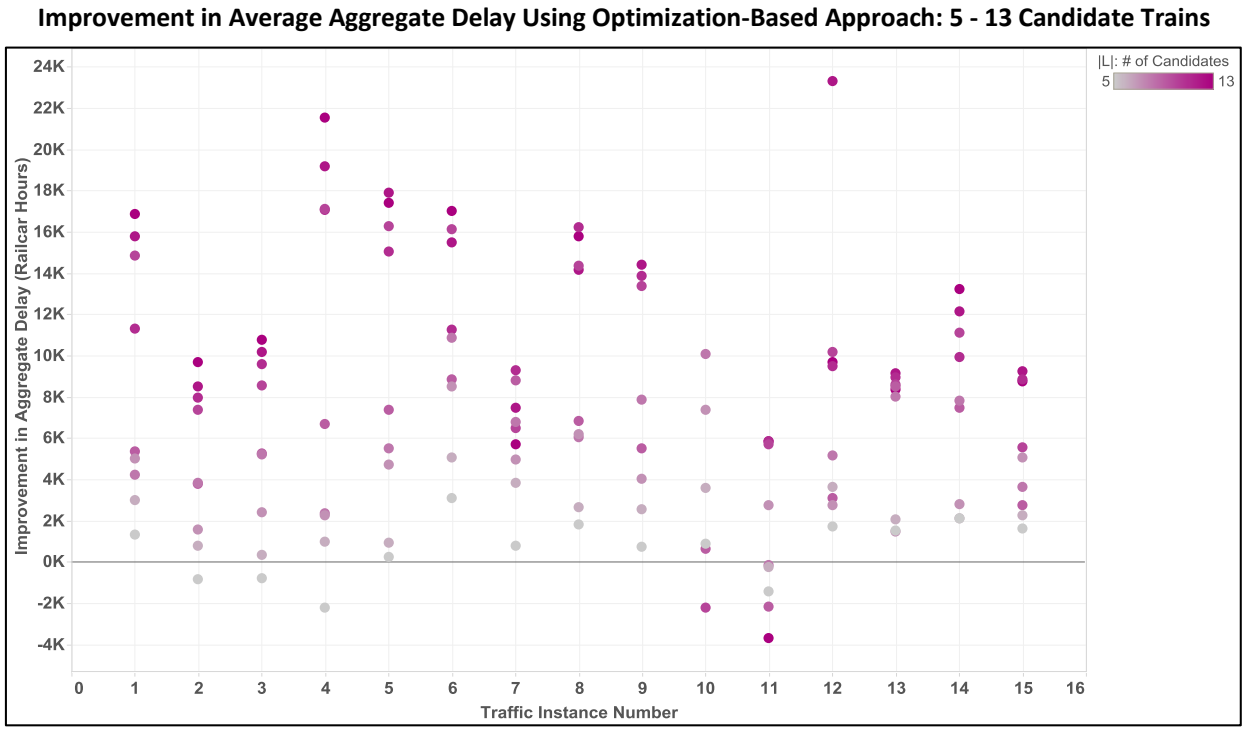


Figure 23: Impact of $|L|$ on improvement of optimization-based approach compared to systematic enumeration after 1 hour

Note that since the developed IP assigns flow variables to commodities instead of individual railcars, the number of commodities (and not necessarily the number of railcars) has direct impact on solution time. Given a fixed number of railcars, the IP size is bigger if the average commodity size is smaller, with 1 being the worst-case scenario.

3.5.2 Results for Tabu Search

Note the following similarities between the Tabu search method proposed in Algorithm 8 and systematic enumeration proposed in Algorithm 3 of Chapter *II*:

1. Both methods rely primarily on the railcar movement simulation described in Algorithm 1.
2. Both methods are run with an a priori maximum time threshold.

This allows us to compare the performance of Tabu search against that of systematic enumeration for identical time cutoffs. In Chapter *II* we asserted the usefulness of the 15-minute, 1-hour, and 10-hour cutoffs, which we will also use in this section. Specifically, Figures 24 and 25 depict the improvement in the objective function value obtained by using Tabu search instead of systematic enumeration for the problem instances described in detail in Section 2.5. Figure 24 reveals a significant improvement in the solution quality attributed to the use of Tabu search.

Figure 24: Objective function improvement of Tabu search solution, with systematic enumeration solution as baseline: 10-hour cuttoff (above), 1-hour cutoff (below)

Figure 25: Objective function improvement of Tabu search solution, with systematic enumeration solution as baseline (15-minute cutoff): $|L| \leq 6$ above, $|L| \geq 7$ below.

Note from Figure 24 that if a 10-hour cutoff is allowed, the improvement introduced by Tabu search becomes particularly significant as $|L|$ increases. This does not seem to be the case for the 1-hour cutoff. In the case of a 15-minute cutoff illustrated in Figure 25 the relative performance of Tabu search seems to depend on the number of candidate trains: It may or may not be beneficial to perform the Tabu search as an alternative to systematic enumeration. Specifically the results indicate $|L| = 7$ is the breaking point, since it is generally preferable to use systematic enumeration for $|L| \geq 7$.

To understand these phenomena note that based on Algorithm 8 the size of the neighborhood in the first iteration of the algorithm will be $\sum_{\ell \in L}(|S_\ell| - 1)$. Consider again the average simulation runtimes detailed in Table 4 from Chapter *II*. Based on this information we construct Table 5, which details the approximate time required to complete the first iteration of Tabu search. Observe that for $|L| \geq 7$ the Tabu search algorithm is generally not expected to complete the first iteration. Thus for problem instances with $|L| \geq 7$, running Algorithm 8 with a 15-minute time cutoff is effectively equivalent to running a systematic enumeration algorithm, though with a different order of searching the solution space than we apply in Chapter *II*. The effective search order here can be thought of as "broad" or "shallow" search, where all solutions ϕ with distance $\nu(\phi, \phi^0) = 1$ are evaluated first. This is in contrast to the systematic enumeration algorithm in Chapter *II* where a "deep" search is used which allows further deviations from ϕ^0 earlier on. As such, the empirical advantage of systematic enumeration under a 15-minute time cutoff when $|L| \geq 7$ suggests the potential superiority of deep search over broad search in extremely time-constrained real-time applications. In the future work section we discuss how this can be used to motivate future research.

Furthermore, notice that as $|L| \to 13$, the time to complete the first iteration approaches the 1-hour mark. This explains the relative degradation in the performance of Tabu search in comparison with systematic enumeration when comparing the two graphs of Figure 24.

Table 5: Time Required to Complete First Iteration of Tabu Search (Minutes)

$\|L\|$	Size of Neighborhood - Iteration #1	Time (Light Traffic)	Time (Regular Traffic)	Time (Heavy Traffic)
2	16	4	4.4	4.6
3	24	6	6.6	6.8
4	32	8	8.7	9.1
5	40	10	10.9	11.4
6	48	12	13.1	13.7
7	56	14	15.3	16
8	64	16	17.5	18.2
9	72	18	19.7	20.5
10	80	20	21.9	22.8
11	88	22	24.1	25.1
12	96	24	26.2	27.4
13	104	26	28.4	29.6

So far we have compared the performance of Tabu search to systematic enumeration from Chapter *II*. Let us also explicitly visualize and compare the best objective value found by Tabu across the various problem instances and the impact of having additional time on solution quality. Figure 26 illustrates the objective function value for the best Tabu search solution for the 10-hour and 1-hour time cutoffs. Recall from Chapter *II* that, by design, if two of the 540 problem instances at hand vary only by $|L|$, the optimal solution for the problem instance with larger $|L|$ dominates that of the problem instance with the smaller $|L|$. With a 10-hour time cutoff, the best solution found follows this pattern as $|L|$ increases; however, the increase seems to be more significant for smaller values of $|L|$. Also note that the advantage of having more candidate trains does not necessarily translate to better Tabu outcomes when the decision must be made within an hour. Furthermore, the trend observed in the 10-hour graph is reversed in Figure 27, which allows for a 15-minute time cutoff. In general, as we noted previously, if only one iteration of search is feasible given the time constraint at hand, the "deep search" of systematic enumeration is the preferred method to solve the problem, which occurs for $|L| \geq 7$ in this particular set of experiments.

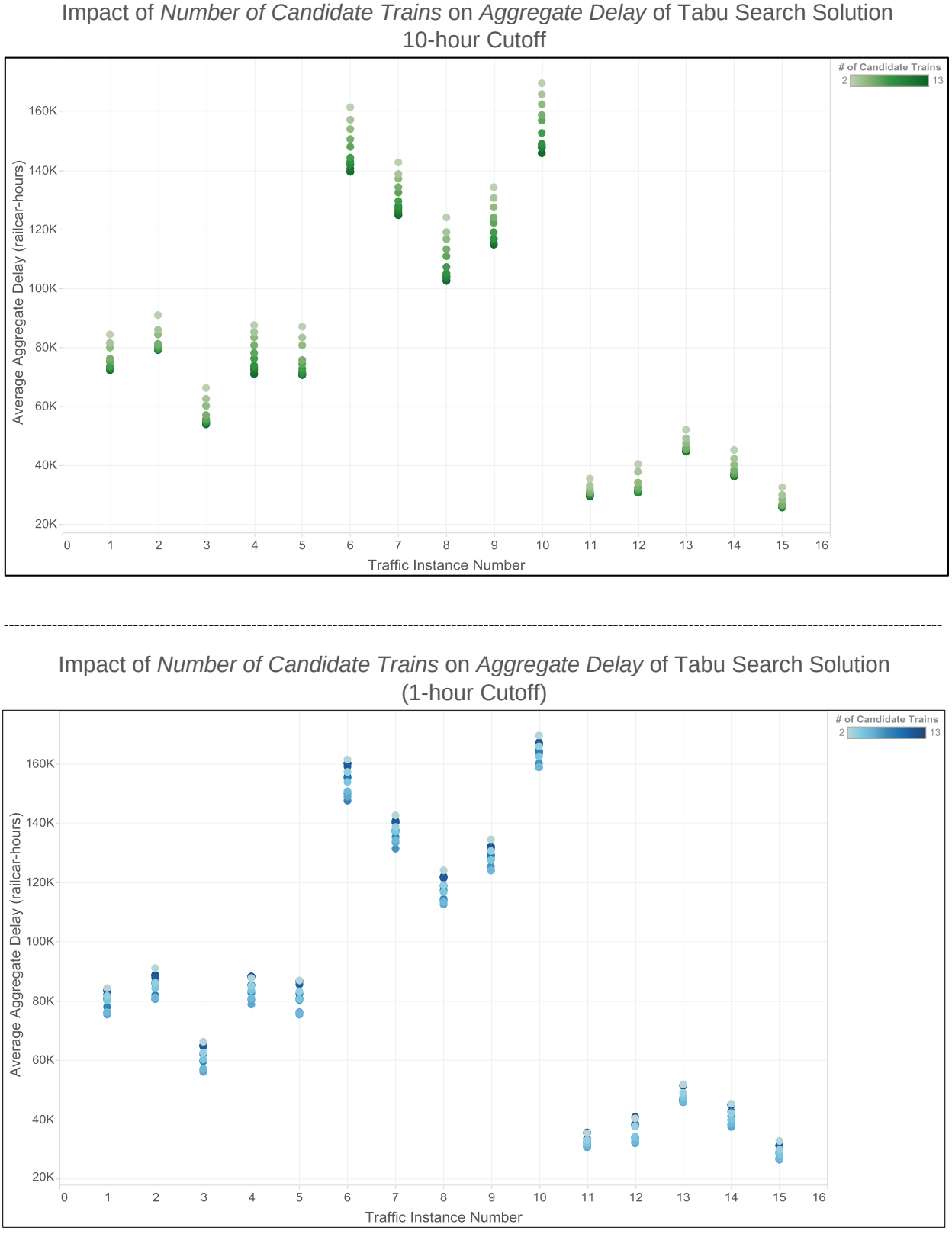


Figure 26: Impact of number of candidate trains on aggregate delay of Tabu search solution with 10-hour cutoff (above) and 1-hour cutoff (below)

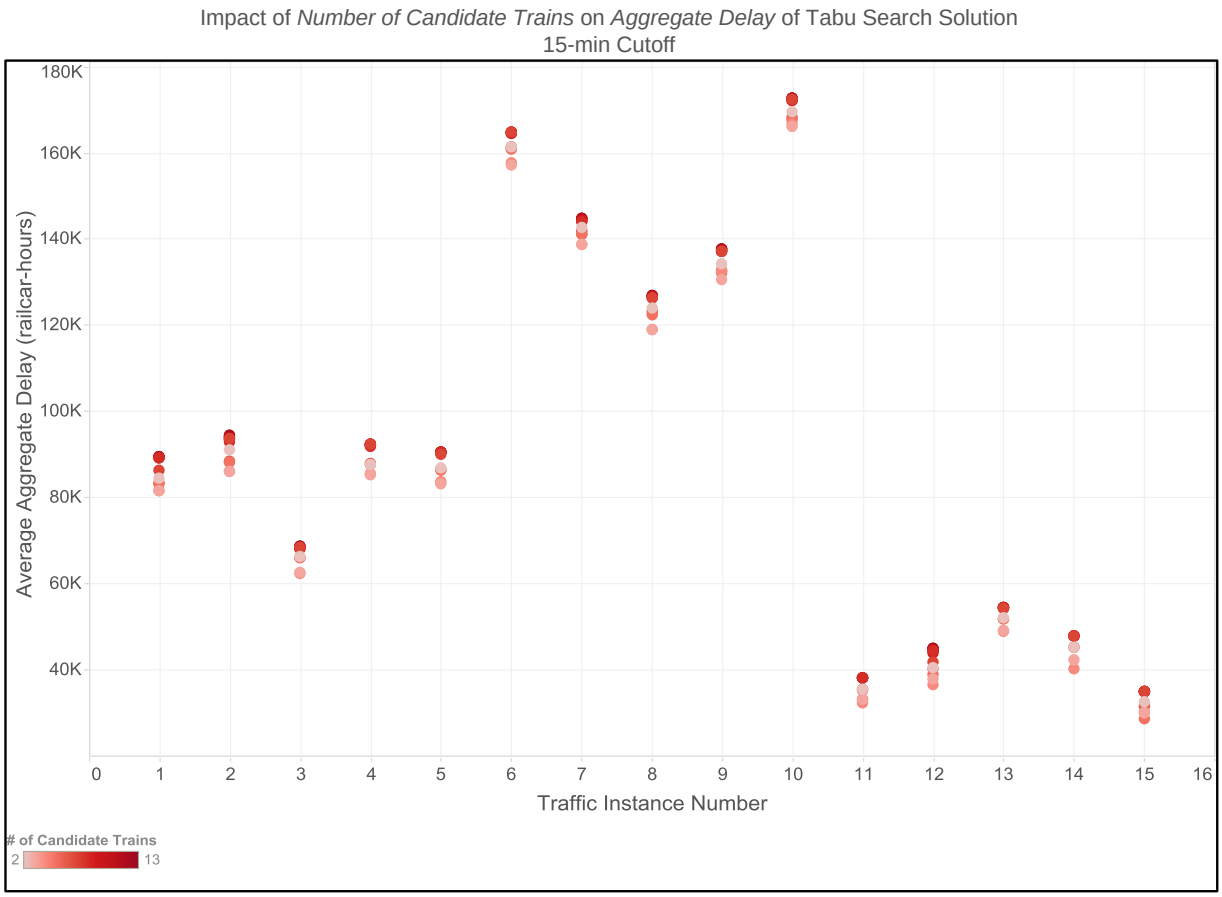


Figure 27: Impact of number of candidate trains on aggregate delay of Tabu search solution with 15-minute cutoff

Figure 28 explicitly illustrates the impact of allowing more time on the quality of the Tabu search solution. In particular, the results show the significance of having more time as $|L|$ increases on the solution quality.

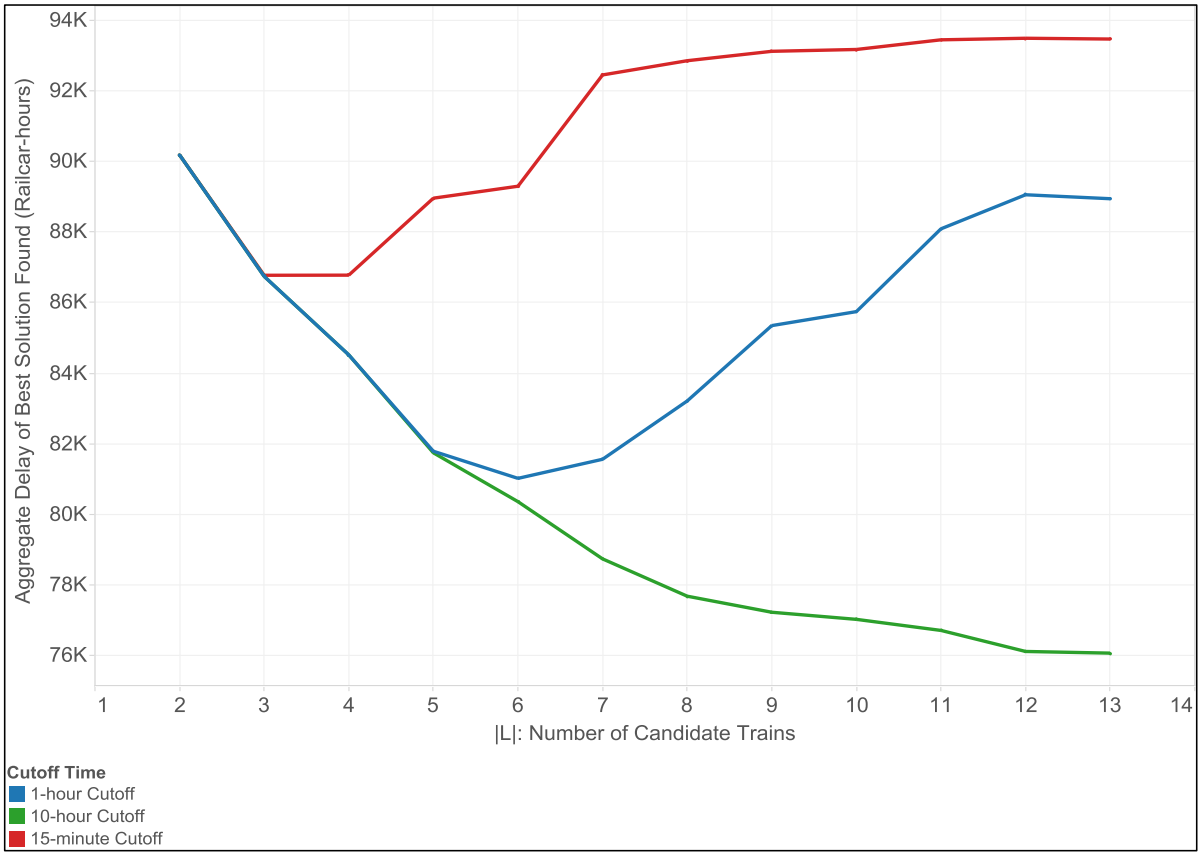


Figure 28: Tabu search results by cutoff time

3.5.3 Direct Comparison of the Two Approaches

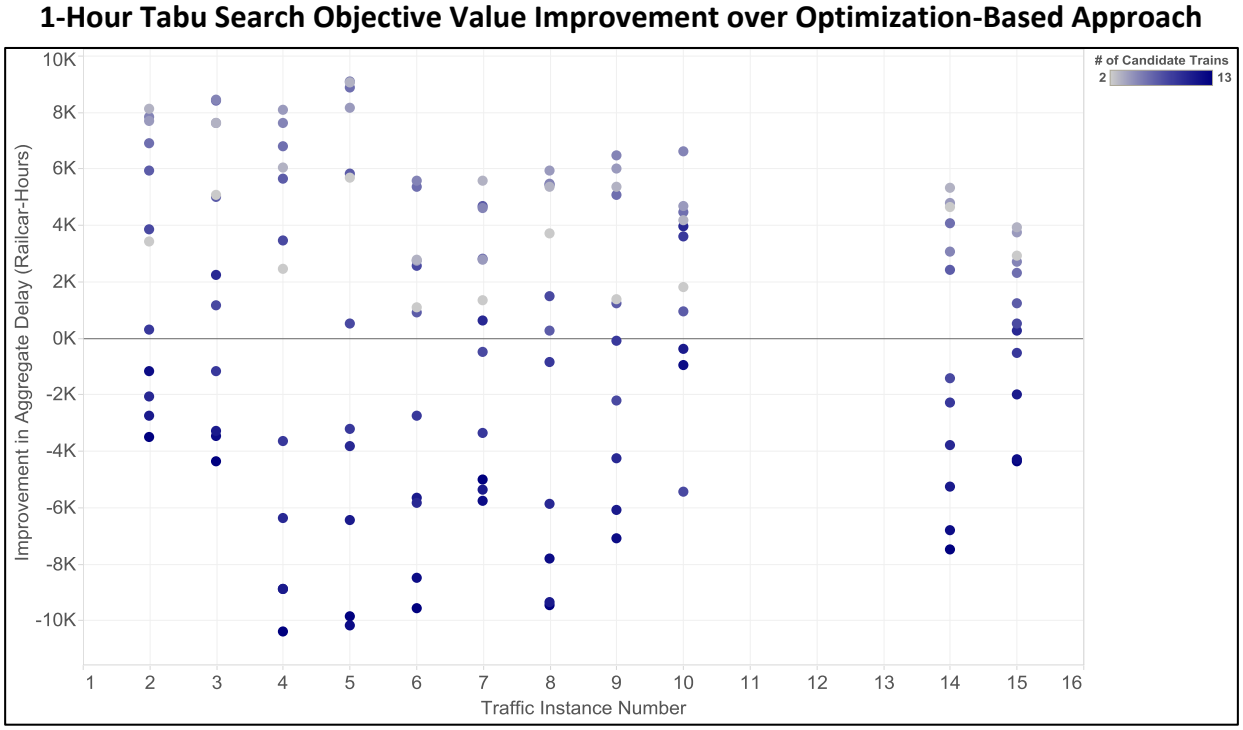
So far each of the two proposed methods in this chapter has been validated against the benchmark set in Chapter *II*. In this section we present a direct comparison between the results of the two approaches.

Recall from Figure 20 that the optimization-based approach completes within 1 hour for all problem instances being run. Therefore, comparing the results of the optimization-based approach with those of Tabu search with a 1-hour time cutoff allows us to understand the circumstances where the optimization-based approach dominates in terms of both solution time and solution quality. Furthermore, Figure 20 shows that for traffic instances with

the lowest number of commodities (i.e. #1, #11, #12, #13) the optimization-based method completes within 15 minutes; it is thus useful to compare optimization results to Tabu results with a 15-minute cutoff for this particular subset of traffic instances.

Figure 29 illustrates the improvement obtained by running Tabu search in comparison with the optimization-based solution as the benchmark. The graph at the top of the figure illustrates this result with a 1-hour Tabu search time cutoff for traffic instances other than the four mentioned above. The graph at the bottom of the figure illustrates this result with a 15-minute Tabu search cutoff for the four traffic instances mentioned above. Both figures (and the top figure in particular) generally show the relative improvement in the optimization-based solution as $|L|$ increases; this finding is consistent with the limitations of Tabu discussed in Section 3.5.2. While this pattern is helpful to understand in general, it is further helpful to study the $|L|$-value that can be considered as the "breaking point", after which the optimization-based method is preferable to Tabu search.

Figures 30 splits the graph at the top of Figure 29 into two graphs, each filtered by restricting $|L|$ to one side of the breaking point value. Figure 31 does the same for the graph at the bottom of Figure 29. These figures illustrate breaking points of $|L| = 8$ in the case of a 1-hour time cutoff and $|L| = 4$ in the case of a 15-minute time cutoff for the instances that are small enough to solve using the optimization-based method within 15 minutes.



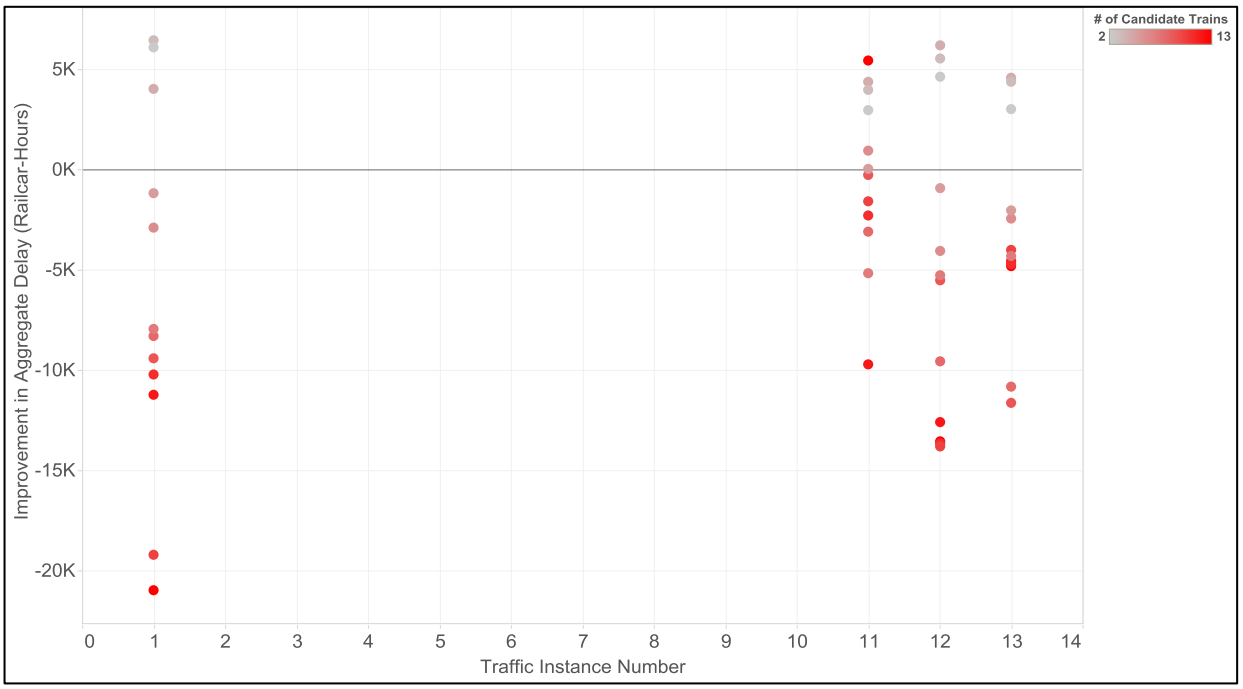


Figure 29: Objective function improvement of Tabu search approach, with the solution of optimization-based approach as basesline: 1-hour time cutoff (above), and 15-minute time cutoff (below)

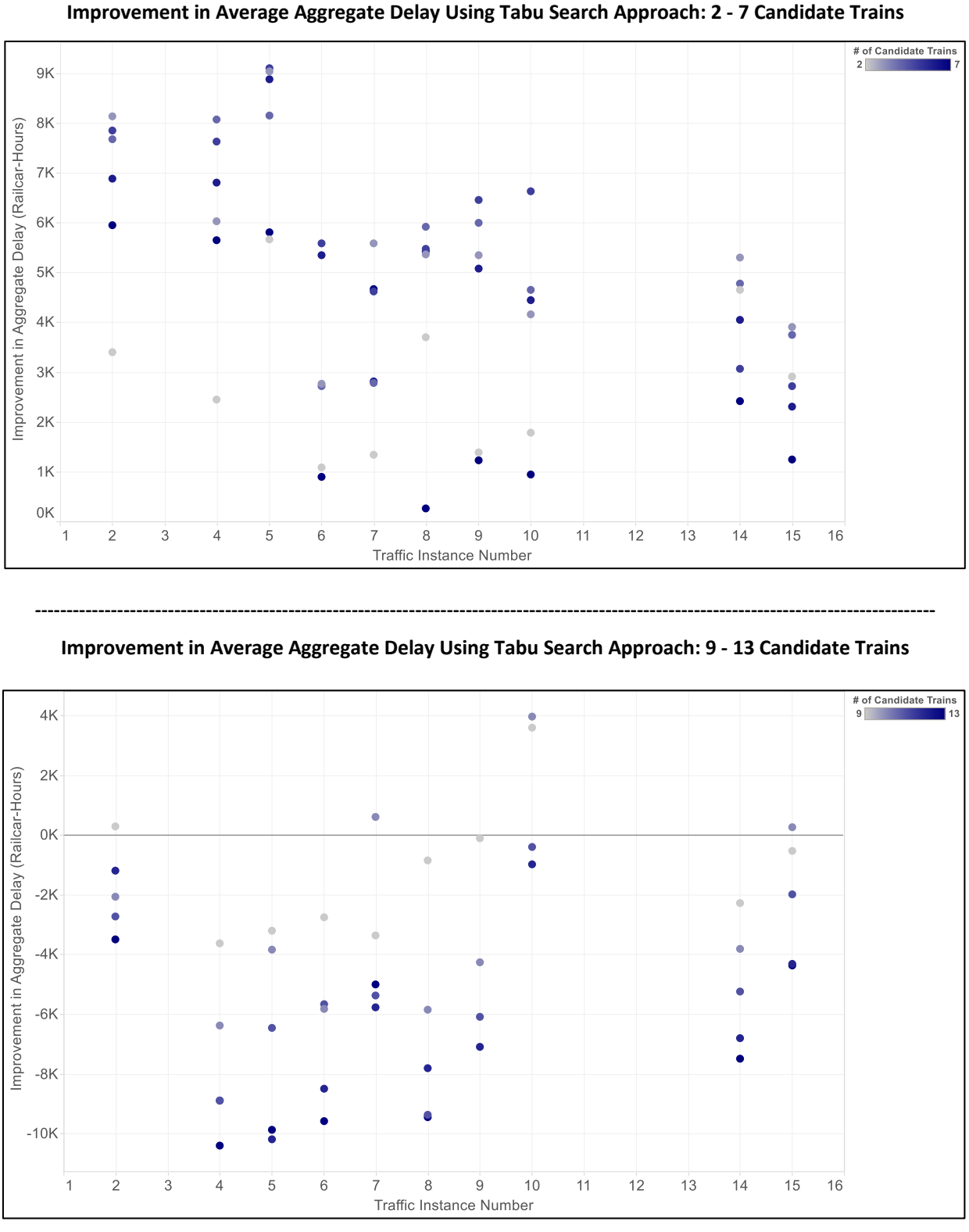


Figure 30: Impact of $|L|$ on performance of Tabu search approach (1-hour cutoff), with optimization-based approach as baseline

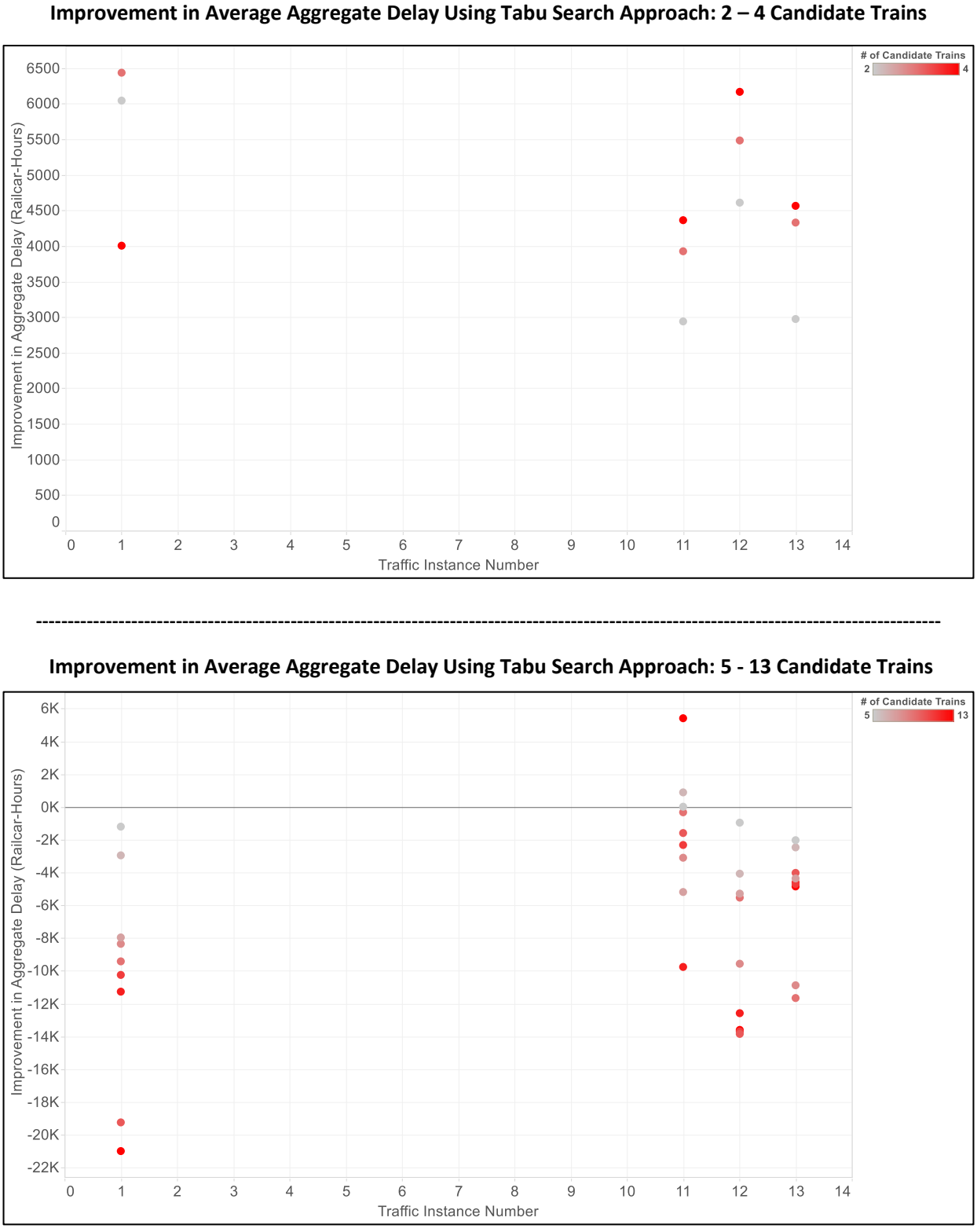


Figure 31: Impact of $|L|$ on performance of Tabu search approach (15-minute cutoff), with optimization-based approach as baseline

3.6 Conclusion

Having used the framework developed in the previous chapter as a building block, we have introduced two methods for modeling and solving bigger instances of the train re-routing problem. We have tested both approaches using real data from the physical rail network of a Class I railroad in the United States. Furthermore, we have presented comprehensive comparisons between the performance of each approach and that of systematic enumeration introduced in Chapter *II*. The results can be summarized as follows:

1. The Tabu search heuristic approach is superior to systematic enumeration if the decision timeline allows more than one iteration of Tabu search to complete. If that is not the case the search order of systematic enumeration is preferable to the search order of the first iteration of Tabu search.

2. The optimization-based approach is superior to systematic enumeration unless the solution space is small enough for systematic enumeration to find the optimal solution, or search the majority of the solution space within the decision timeline.

3. The optimization-based solution has better objective function value compared to the Tabu search solution as the number of candidate trains increases. However, the solution time of the Tabu search approach is more robust against increases in network traffic, while the optimization-based approach slows considerably under scenarios where we pressure-test increases in network traffic.

3.7 Future Work

Based on the results in this chapter the recommended extension to the Tabu search method presented in this chapter is a time-threshold-aware Tabu search algorithm. Specifically for problem instances with a very narrow time cutoff it will be interesting to see the impact of prematurely stopping the neighborhood search in order to ensure that the algorithm runs a

minimum number of iterations. Recall that if only one iteration is practically doable in the allotted time, then the algorithm reduces to systematic enumeration with a different search order. An example of a time-threshold-aware algorithm could be Algorithm 9:

Algorithm 9 Tabu Local Search Algorithm Extension

Definitions
Let τ represent time elapsed since start of algorithm.
Let τ' represent the time threshold.
Let m represent the minimum acceptable number of iterations.
Procedure
Step 1: Set $\phi^0 = (\phi^0_1, \phi^0_2, ..\phi^0_{|L|})$, where ϕ^0_ℓ is the original destination terminal for train ℓ. Set $T = \emptyset$, $\phi^*_{local} = \phi^0$, $\phi^*_{global} = \phi^0$, $z^*_{global} = z(\phi^0)$, and iteration counter $i = 1$.
Step 2: Set $z^*_{local} = \infty$ and $\phi_{prev} = \phi^*_{local}$.
For each $\phi \in N(\phi^*_{local})\backslash T$:
  Compute $z(\phi)$ using algorithm 1.
  If $z(\phi) < z(\phi^*_{local})$:
    set $\phi^*_{local} = \phi$ and $z^*_{local} = z(\phi)$.
  If $z(\phi) < z(\phi^*_{global})$:
    set $\phi^*_{global} = \phi$ and $z^*_{global} = z(\phi)$.
  If $\tau \geq \frac{i\tau'}{m}$:
    Break loop.
i++
Step 3: Set $T = \{\phi | \nu(\phi, \phi_{prev}) \leq 1\} \cap \{\phi | \nu(\phi, \phi^*_{local}) = 1\}$
Step 4: If $\tau \geq \tau'$, stop. Otherwise, go to step 2.

In addition, the results in section 3.5.2 showed an empirical advantage for performing deep neighborhood search as opposed to broad neighborhood search in real-time applications with extremely tight time cutoffs. This should be further explored in a broader context since, as shown, extremely short time cutoffs have the potential to render sophisticated search heuristics less advantageous to systematic solution enumeration. This complication can inspire research into optimizing overall search strategy in situations where smart phase changes in the search strategy are rendered ineffective due to the time cutoff constraint.

Chapter *IV*: A Study of Freight Train Re-Routing Problems Under Traffic-Dependent Railyard Processing Rate Assumptions

4.1 Introduction

The models developed in Chapters *II* and *III* have treated the classification section of a rail terminal as a queue where the server rate is independent of the queue length. In the particular context of this research, the items in the queue are railcars stationed on the rail terminal arrival tracks. As discussed in Chapter *I*, these railcars are eventually moved from the arrival tracks through the classification section, then to the departure tracks of the same rail-yard. Depending on the detailed structure of an individual rail terminal, it is conceivable that the presence of more railcars on the arrival tracks may potentially reduce the speed with which railcars are processed. The main reason for such reduced processing rate is the added difficulty in handling railcars when there is higher congestion.

In this chapter we generalize the train re-routing problem by relaxing the assumption that all rail terminals have a processing rate that is a constant linear function of the number of cars awaiting processing. We extend the railcar movement simulation algorithm from Chapter *II* to handle the generalized train re-routing problem. Using this, we extend the heuristic solution method developed in Chapter *III* to address the problem variant. We further show that the optimization-based solution method developed in Chapter *III* can be extended to address the problem variant without losing the linearity of the constraints and objective function in the case that rail terminal processing rate is a linear function of the number of railcars awaiting processing. The results of these extensions are furthermore tested computationally on problem instances that reflect the size of operations of a Class I railroad in the United States.

4.2 Literature Review

Queuing models with non-independent service rate have been studied in the literature. Posner [29] considers a single-server queueing system where server rate is dependent on waiting time, particularly the case where service time is exponential with its parameter dependent on waiting time and where that dependence is characterized as a step-function. Bekker et al. [5] develop the steady-state waiting time for an "initially single-server" queueing system where a second server is added as soon as there is a customer whose wait time exceeds a given threshold.

Part of our work in this chapter involves generalizing the bundling constraints of the multi-commodity network design methodology developed in Chapter *III* to address an extension of the original problem definition. Current academic literature does contain other examples of similar extensions to multi-commodity network flow problems being applied to realistic problem instances. For instance, Castro [9] implements a multi-commodity network flow problem with generalized bundling constraints, where multiple arcs can have joint upper or lower bounds [9]. As will be shown in section 4.4.2, we will also need to generalize the bundling constraints in our multi-commodity network design formulation in order to correctly model the problem variant.

Desaulniers et al. [14] studies another interesting application of the multi-commodity network flow problem. It formulates the crew pairing problem at Air France as a multi-commodity network flow problem with non-linearity in both the objective and the constraints. The objective of the model is creating the min-cost set of crew pairings across all flight legs. The constraints arise from regulatory and other rules the airline must follow in its schedule. The paper extends the Dantzig-Wolf decomposition to construct a branch-and-bound solution methodology for the formulation [14].

In addition to the supply chain and transportation industry, there are examples in the literature of extensions to the multi-commodity network flow problem to routing problems

in other contexts. For instance, Ouorou et al. [26] surveys convex multi-commodity network flow problems and studies their application to the message routing problem. It studies and tests four solution methods which leverage the linearity of the constraints to solve the problem, and compares the results from each method [26].

4.3 Problem Definition

In Chapter *I*, section 2.3 we defined the train re-routing problem: Given a set of candidate trains L, the problem is to determine a potential new destination terminal for each $\ell \in L$. We now define the modified train re-routing problem as an extension of the train re-routing problem, where a key assumption about railcar processing at rail terminals is modified.

Recall how we view the operations of a rail terminal as they pertain to this study, illustrated again in Figure 32. Any rail terminal s has an arrival yard, classification section, and departure yard. The railcars in the arrival yard are processed per an a priori rate, μ_s and sent to the departure yard, where each railcar k joins the train which will carry k along the next leg of its trip plan.

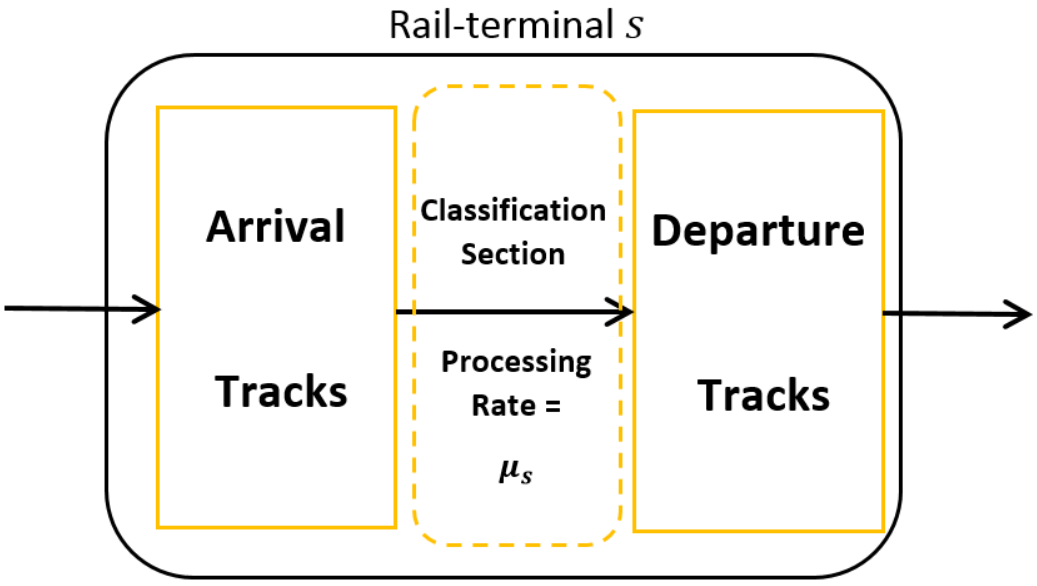


Figure 32: Rail terminal as it is modeled in this paper

In the previous two chapters, independence was assumed between μ_s and the number of railcars awaiting processing in the arrival yard of s. In other words, μ_s could have been thought of as a constant linear function of the number of railcars in the arrival yard, a_s. In this chapter we examine the train re-routing problem defined in Chapter *I*, section 2.3, but with the following generalized assumption:

$$\mu_s = g_s(a_s) \tag{20}$$

, where g_s is a monotonically non-increasing function. This generalization reflects the fact that excess traffic may slow down or partially impede the processing operations of the terminal.

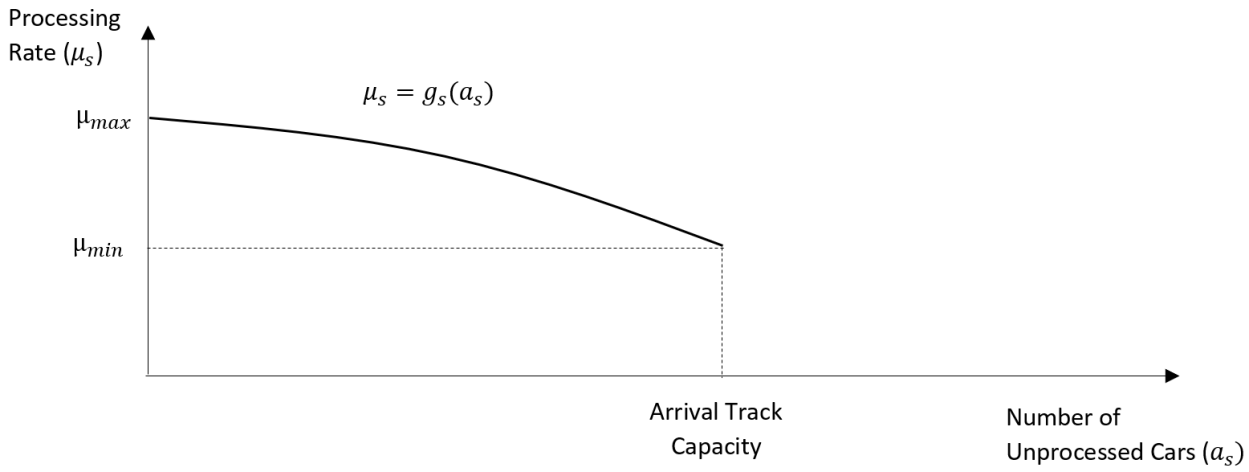


Figure 33: Monotonically Non-Increasing processing rate

4.4 Model

Given the modified problem definition, let us revisit both modeling approaches introduced in Chapter *III* and discuss the necessary modifications to each to solve the modified train re-routing problem. We will show that:

1. The optimization-based approach can be extended to the train re-routing problem without losing the linear structure if g_s is a linear function $\forall s$, and

2. We can extend the railcar movement simulation model and search-based solution method to solve the train re-routing problem, without the need to assume g_s is linear.

Recall the time-space network we created in Chapter *II*, a portion of which is re-illustrated in Figure 34. Using the time-space network we developed a railcar movement simulation in Chapter *II* as a framework to evaluate the performance of any solution to the train re-routing problem (algorithm 1), which we used as the building block for the Tabu search algorithm introduced in Chapter *III* (algorithm 8).

Let us recall a few definitions from chapter *II*:

$G = (V, A)$: The mathematical representation of the time-space network.

U_{ij}: upper bound, measured in number of railcars, of arc $(i, j) \in A$.

x_{ij}^k: number of railcars from commodity k to flow on arc $(i, j) \in A$.

K: Set of all commodities present in the rail network over the intended horizon.

$t(i)$: Value of time associated with time-space node $i \in V$.

In Chapters *II* and *III* U_{ij} was set a priori of running the railcar simulation model $\forall (i, j) \in A$. For the modified train re-routing problem characterized with the addition of Equation 20, U_{ij} cannot be determined a priori to running the railcar movement simulation model if $(i, j) \in A$ is a processing arc. We will now go over how to set U_{ij} in this scenario.

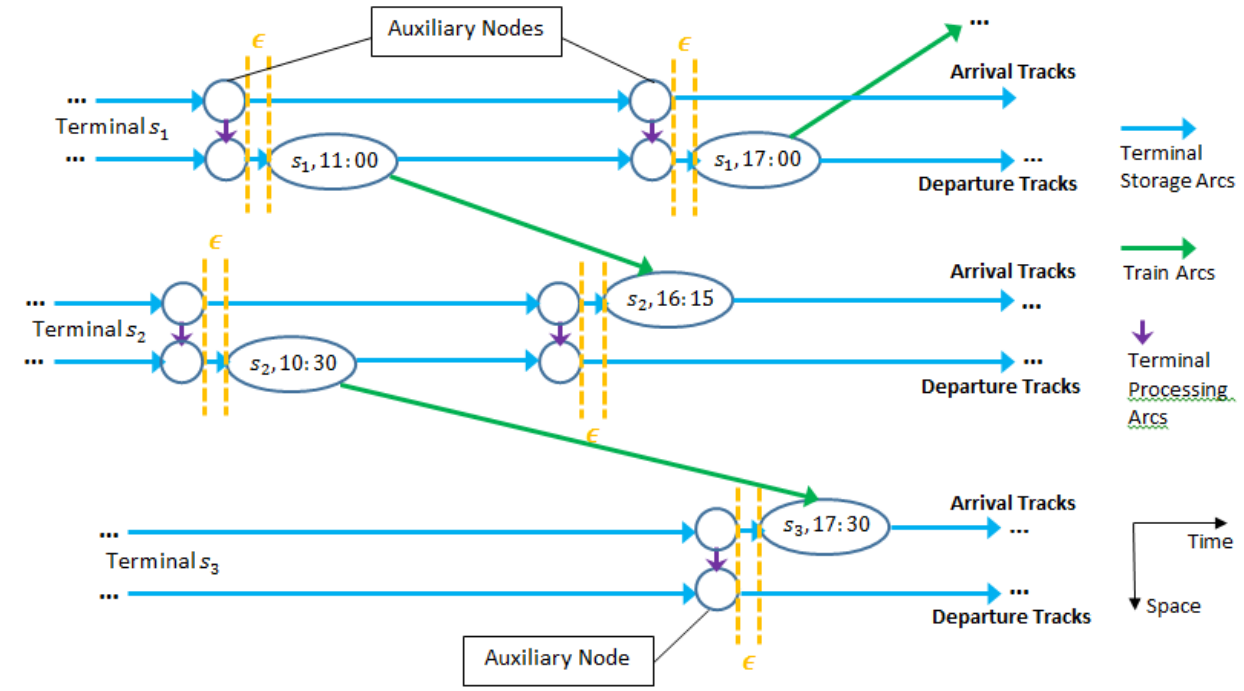


Figure 34: Time-space network with processing arcs

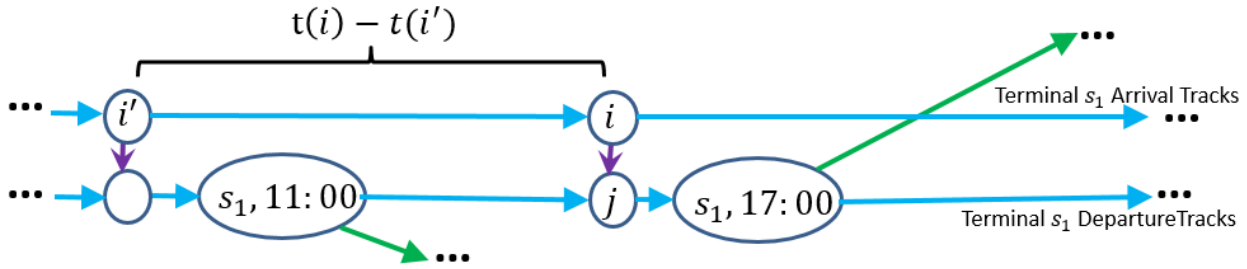


Figure 35: Small portion of time-space network with processing arc $(i, j) \in A$

Consider any processing arc (i, j) such as the one in Figure 35. Recall from Chapter *II*, Equation 3 that our model uses U_{ij} to account for railcar processing capacity between time $t(i')$ and time $t(i)$, where $i' \in V$ is the predecessor node of $i \in V$. Note the following observations:

1. Let $t(i') + dt$ denote the time an infinitesimal unit of time after $t(i')$. The number of railcars at the arrival yard at time $t(i') + dt$ is represented by the flow on arc (i', i), which we denote with $\sum_{k \in K} x^k_{i',i}$.

2. If more than $\sum_{k\in K} x^k_{i',i}$ can be processed between time $t(i')$ and time $t(i)$, setting $U_{ij} = \sum_{k\in K} x^k_{i',i}$ is sufficient for allowing the processing of all cars in the arrival yard by time $t(i)$.

Based on the above observations, for our modeling purposes U_{ij} can be treated as mathematically equivalent to the number of railcars processed between time $t(i')$ and time $t(i)$. As railcars are processed during this time period, the effective processing rate increases from μ_0 toward μ_{max}, the theoretical highest processing rate when the terminal is empty. This dynamic is shown in Figure 36:

- At time $t(i')$, point $(\sum_{k\in K} x^k_{i',i}, \mu_0)$ represents the situation in the rail terminal.
- At time $t(i)$, point $(\sum_{k\in K} x^k_{i',i} - U_{ij}, \mu_{t(i)})$ represents the situation in the rail terminal.

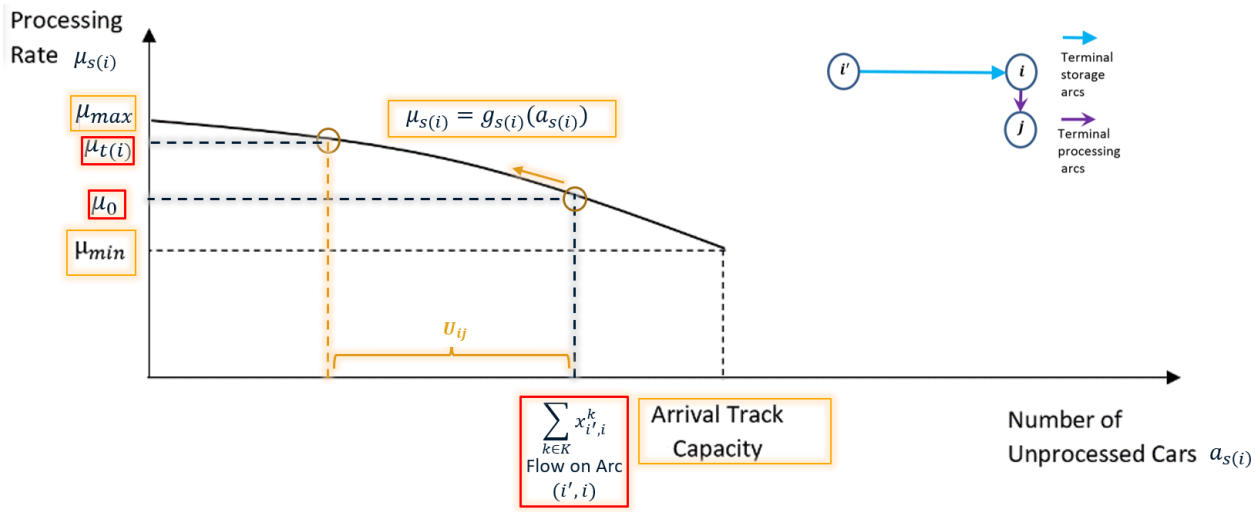


Figure 36: Illustration of U_{ij} calculation. Quantities enclosed in golden boxes are known a-priori; quantities enclosed in red boxes are not.

For simplicity, we define $t(i',i) \triangleq t(i) - t(i')$. Note that U_{ij} can be calculated as a function of $\sum_{k\in K} x^k_{i',i}$, $t(i',i)$, and $g_s(i)$. Since $t(i',i)$ and $g_s(i)$ are known a-priori in the context of the modified train re-routing problem, the challenge in setting U_{ij} is the fact that it is a function of $\sum_{k\in K} x^k_{i',i}$:

$$U_{ij} = f_{ij}(\sum_{k \in K} x^k_{i',i}) \tag{21}$$

Claim 1: The following differential equation characterizes the relationship between U_{ij} and μ_0.

$$\frac{dU_{ij}}{dt(i',i)} = \begin{cases} \mu_{t(i)}, & U_{ij} < \sum_{k \in K} x^k_{i',i} \\ 0, & U_{ij} \geq \sum_{k \in K} x^k_{i',i} \end{cases} \tag{22}$$

The top case in Equation 22 reflects the scenario where not all railcars present at the arrival yard of $s(i)$ at $t(i')$ can be processed by $t(i)$. Note from Figure 36 that in this case, at time $t(i)$ we will have $\sum_{k \in K} x^k_{i',i} - U_{ij}$ railcars in the arrival yard of $s(i)$ and our processing rate will be $\mu_{t(i)}$.

The bottom case in Equation 22 reflects the scenario where the arrival yard of $s(i)$ is emptied before $t(i)$. In this case having had more time between i' and i does not contribute to the ability to process more railcars during that time.

Combining Equations 20 and 22 we can write:

$$\frac{dU_{ij}}{dt(i',i)} = \begin{cases} g_{s(i)}(\sum_{k \in K} x^k_{i',i} - U_{ij}), & U_{ij} < \sum_{k \in K} x^k_{i',i} \\ 0, & U_{ij} \geq \sum_{k \in K} x^k_{i',i} \end{cases} \tag{23}$$

Thus, f_{ij} in Equation 21 is the solution to the differential equation above.

Claim 2: Consider Equation 24 below.

$$\frac{dU_{ij}}{dt(i',i)} = g_{s(i)}(\sum_{k \in K} x^k_{i',i} - U_{ij}) \tag{24}$$

Given the same initial condition, let U^A_{ij} refer to the solution to Equation 23 and U^B_{ij} refer to the solution to Equation 24. Then Equation 25 holds so long as $g_{s(i)}(a_{s(i)}) \geq 0 \, \forall \, a_{s(i)}$ holds. Note that this is not an onerous assumption since in the context of the train re-routing

problem it is not meaningful for the processing rate at a terminal is to fall below zero.

$$U_{ij}^{A} \leq U_{ij}^{B} \tag{25}$$

Claim 3: Given that $U_{ij} \geq \sum_{k \in K} x_{i',i}^{k}$, then the set of possible feasible flows for the constructed time-space network G remains unchanged regardless of the particular value of U_{ij}.

Based on the above three claims, we can solve Equation 24 in place of Equation 23 without impacting the set of possible feasible flows of G. Since the theoretical number of railcars that can be processed in zero time is zero, the initial condition for solving Equation 24 is as follows:

$$U_{ij}(0) = 0 \tag{26}$$

Lemma: f_{ij} is a linear function if $g_{s(i)}$ is a linear function.

Proof of Lemma:

Consider a linear relationship between $\mu_{s(i)}$ and the number of railcars in the arrival yard, $a_{s(i)}$.

$$\mu_{s(i)} = -ma_{s(i)} + \mu_{max} \tag{27}$$

, where m is the absolute value of the slope of the linear relationship and μ_{max} is the fastest theoretical processing rate, when the arrival yard is empty.

This relationship is illustrated in Figure 37. Let μ_0 denote the processing rate at $s(i)$ at time $t(i')$, and note the simple linear relationship between μ_0 and $\sum_{k \in K} x_{i',i}^{k}$:

$$\mu_0 = \mu_{max} - m \sum_{k \in K} x_{i',i}^{k} \tag{28}$$

It is therefore sufficient to proof there exists a linear relationship between U_{ij} and μ_0, which we focus on for the remainder of the proof.

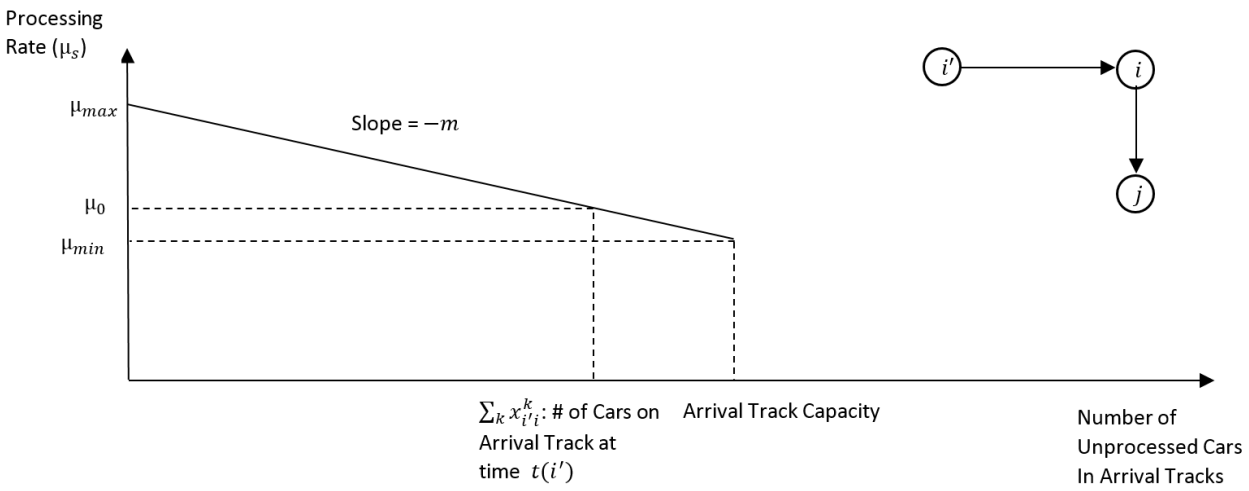


Figure 37: Processing rate μ_0 at timepoint $t(i')$ corresponding to $i' \in V$

Recall the dynamic described using Figure 36. Figure 38 illustrates the same dynamic under the case where $g_{s(i)}$ is a linear function, in which case Equation 24 can be simplified as follows:

$$\frac{dU_{ij}}{dt(i',i)} = mU_{ij} + \mu_0 \tag{29}$$

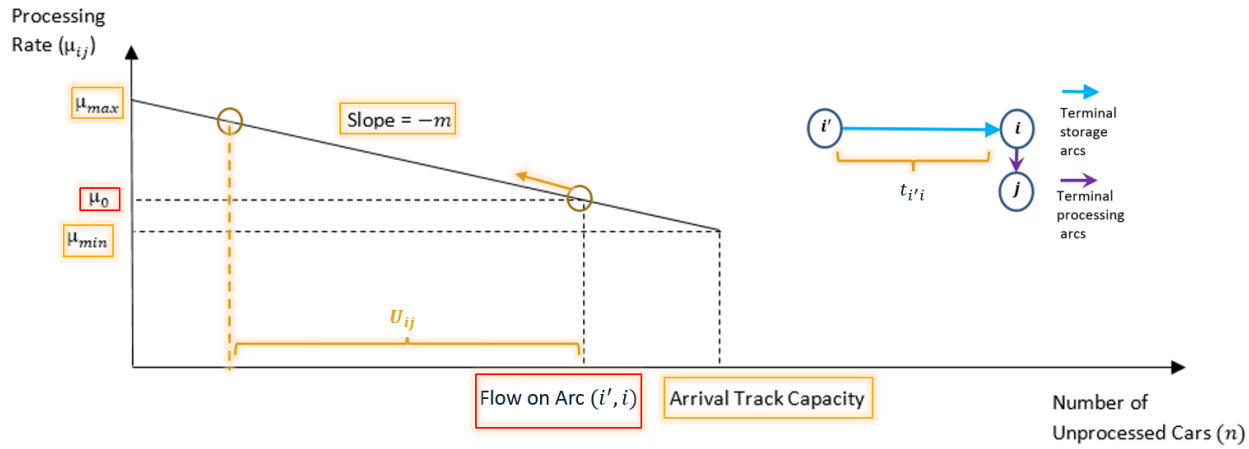


Figure 38: Illustration of U_{ij} for linearly declining processing rate

Solving Equation 29, we get:

$$U_{ij} = \begin{cases} \frac{e^{mt(i',i)}-1}{m}\mu_0 & ,m \neq 0; \\ t(i',i)\mu_0 & ,m = 0. \end{cases} \tag{30}$$

Note that the $m = 0$ case simply represents instances where processing rate is independent of the number of cars at the arrival tracks, which were explored in Chapter *III*. For the $m \neq 0$ case, we combine Equations 28 and 30 to write:

$$U_{ij} = \mu_{max}\frac{e^{mt(i',i)} - 1}{m} - (e^{mt(i',i)} - 1)\sum_{k \in K} x_{ij}^k \tag{31}$$

Since m, $t(i', i)$, and μ_{max} are determined by the characteristics of the associated rail terminal they can be known a priori. As such, $\exists$ a linear relationship between U_{ij} and $\sum_{k \in K} x_{ij}^k$.

$$U_{ij} = \alpha - \beta \sum_{k \in K} x_{i'i}^k \tag{32}$$

, where $\alpha = \mu_{max}\frac{e^{mt(i',i)}-1}{m}$ and $\beta = (e^{mt(i',i)} - 1)$ are constants. QED

4.4.1 Modification of Railcar Movement Simulation

Recall that in Chapter *II* we introduced a railcar movement simulation method (Algorithm 1) to evaluate the objective function value (aggregate railcar delay) in the network. We now extend the algorithm as needed in order to accomplish the same thing given the additional assumption characterized by Equation 20.

Algorithm 1 in Chapter *II* required the inputs in the following list:

1. The constructed time-space network G, including $U_{ij} \forall (i, j) \in A$.
2. The set of railcars presently in the network C_p.
3. The set of railcars entering the network in the future and during the planning horizon C_f.
4. The deadline of each railcar τ_c.

5. The operating rules of the rail network (e.g. FIFO).

In effect, the algorithm gives us the capability to simulate future railcar movements in the time-space network given a situation characterized by the above set of inputs. Having this same capability for the modified train re-routing problem requires us to modify Algorithm 1 to accurately simulate railcar movements given the modified problem definition. In the previous section we showed that, for the modified problem definition, U_{ij} cannot be set a-priori of running the simulation since the relationship between the upper bound of any processing arc $(i, j) \in A$ and the flow on the terminal storage arc leading to $i \in V$ is captured by Equation 21. As such, the equivalent algorithm for the modified train re-routing problem must dynamically set U_{ij} for any $(i, j) \in A$ which is a processing arc. Since the railcar flow along processing arcs is determined in Step 5d of 1, we have modified this step to make the dynamic U_{ij} calculation using Equation 21. Algorithm 10 is the formal result of making this change.

Algorithm 10 Railcar movement model applied to network with FIFO rule and non-linear processing rate

Require: $G = (V, A)$, C_p, C_f.

Require: $t(i), s(i) \quad \forall i \in V$.

Require: $\tau(c) \quad \forall c \in C_p \cup C_f$.

Require: The set of functions $g_s \forall s$ from Equation 20.

Step 0 - Initialization:

For each rail terminal s, initialize $Q_A(s)$ and $Q_D(s)$ as two sorted sequences or queues of railcars, each starting with no elements.

For each train arc (i, j), Initialize an empty sorted sequence $Q_{(i,j)}$ as the queue of railcars to traverse (i, j).

Set $x_{ij}^c = 0 \quad \forall (i, j) \in A, \forall c \in C_p \cup C_f$.

Step 1 - Add current railcars to queue

Step 1a: Sort the railcars in C_p into sequence $\overline{C}_p$ in non-decreasing order of $\tau(c)$.

Step 1b: Remove the first railcar c in $\overline{C}_p$ and note both its rail terminal, $\overline{s}$, and whether it is on the arrival or departure tracks.

Step 1c: If c is on the arrival tracks of $\overline{s}$:

Add c to the end of $Q_A(\overline{s})$.

If c is on the departure tracks of $\overline{s}$:

Add c to the end of $Q_D(\overline{s})$.

Step 1d: If $\overline{C}_p$ is empty:

Go to Step 2.

Else:

Go to Step 1b.

Step 2 - Create sequences for near-future railcars: For each rail terminal s:

Place the subset of railcars in C_f that come into the network at s into sequence $\overline{C}_f(s)$.

Sort cars in $\overline{C}_f(s)$ in non-decreasing order of $\tau(c)$.

Step 3 - Sort nodes by point in time: Sort the nodes in V into sequence V_{sorted} in order of non-decreasing $t(i)$. Break ties by placing nodes that correspond to arrival tracks prior to nodes that correspond to departure tracks.

Step 4 - Pick next node: If there are no remaining nodes in V_{sorted}:

Terminate the algorithm.

Else:

Remove the first node i from V_{sorted}.

If i corresponds to the arrival tracks of $s(i)$:

Go to Step 5.

Continued on next page

If i corresponds to the departure tracks of $s(i)$:

Go to Step 6.

Step 5 - Arrival node steps

Step 5a: If $\overline{C}_f[s(i)]$ is empty or the first railcar c in $\overline{C}_f[s(i)]$ has $\tau(c) > t(i)$, skip to Step 5b. Otherwise, remove c from $\overline{C}_f[s(i)]$, add it to the end of $Q_A[s(i)]$, and repeat Step 5a.

Step 5b: For all nodes $\hat{i} \in V$ such that $\exists$ a train arc $(\hat{i}, i) \in A$: Add all railcars in $Q(\hat{i}, i)$ to the end of $Q_A[s(i)]$.

Step 5c: If $Q_A[s(i)]$ is an empty sequence, go to Step 4.

Step 5d: If processing arc (i, j) departs i:

Let $i' \in V$ represent the predecessor node of $i \in V$ s.t. (i', i) is a terminal storage arc.

Let $|Q_A[s(i)]|$ represent the number of railcars in $Q_A[s(i)]$.

Compute U_{ij} by solving $\frac{dU_{ij}}{dt(i',i)} = g_{s(i)}(|Q_A[s(i)]| - U_{ij})$, $U_{ij}(0) = 0$.

If $g_{s(i)}$ is linear, $g_{s(i)}(a_s) = -ma_s + \mu_{max}$:

Setting U_{ij} simplifies to $U_{ij} = \mu_{max}\frac{e^{m[t(i)-t(i')]}-1}{m} - (e^{m[t(i)-t(i')]} - 1)|Q_A[s(i)]|$.

Set $n = max(U_{ij}, |Q_A[s(i)]|)$.

For each of the first n cars from $Q_A[s(i)]$, denoted c:

Remove c from $Q_A[s(i)]$.

Set $x_{ij}^c = 1$.

Add c to the end of $Q_D[s(i)]$.

Step 5e: Let (i, j) denote the rail terminal storage arc or feasibility arc leaving node i. Set $x_{ij}^c = 1 \quad \forall c$ such that c is in sequence $Q_A[s(i)]$.

Step 6 - Departure node steps

Step 6a: If $Q_D[s(i)]$ is an empty sequence, go to Step 4.

Step 6b: If $\exists j \in V$ such that (i, j) is a train arc, let $Q_D^{(i,j)}$ represent the ordered sub-queue of railcars in $Q_D[s(i)]$ whose trip plan includes $s(i) \rightarrow s(j)$ as a leg.

Set $n = max(U_{ij}, |Q_D^{(i,j)}|)$.

Remove up to n railcars from $Q_D^{(i,j)}$. Remove the same set of railcars from $Q_D[s(i)]$.

For railcar c removed from $Q_D[s(i)]$:

Set $x_{ij}^c = 1$.

If $s(j)$ is the final rail terminal in the trip plan of c:

Set $x_{ji_{sink}} = 1$.

Else:

Add c to $Q_{(i,j)}$.

Step 6c: Let (i, j) now denote the rail terminal storage arc or feasibility arc leaving node i. Set $x_{ij}^c = 1 \quad \forall c$ where c is in sequence $Q_D[s(i)]$.

Similarly to Algorithm 1 in section 2.4.2, once Algorithm 10 is run to simulate railcar delay, x_{ij}^c is determined $\forall (i,j) \in A, c \in C_p \cup C_f$. The remainder of the solution method is unchanged, and is completed by following the below sequential steps:

1. Set ρ_{ij}^c in the same manner as section 2.4.2 and, for a given simulation run with corresponding x_{ij}^c $\forall (i,j) \in A$, calculate the delay, δ_c for any railcar c using Equation 5.
2. Run Algorithm 2: in Step 2, call Algorithm 10 instead of Algorithm 1.
3. If solving using systematic enumeration, run Algorithm 3.
4. If solving using Tabu search, run Algorithm 8.

4.4.2 Modification of Optimization-Based Approach

Recall the arc-based multi-commodity flow optimization model we introduced in section 3.3 of Chapter *III*, equations 11-18. In this section we extend the model to solve the modified train rerouting problem. First we add a few necessary definitions:

- Let $A_{mod} \subset A$ represent the set of processing arcs that are associated with rail terminals where processing has a decreasing linear relationship with the number of railcars present in the arrival yard.
- Let $s(i)$ represent the terminal corresponding to time-space node $i \in V$.
- Let $\mu_{max,s}$ represent the theoretical maximum processing rate at terminal s.
- Let m_s represent the absolute value of the slope of the linear relationship between processing rate and number of railcars in arrival yard, for terminal s.

In section 4.4 we explained the changes to U_{ij} for any processing arc $(i,j) \in A_{mod}$. Since U_{ij} was part of the bundling constraint formulation, for any such arc (i,j) we will introduce constraint 33 in lieu of constraint 16 in the original optimization formulation:

$$\sum_{k \in K_{(i,j)}} x_{ij}^k \leq \alpha_{s(i)} - \beta_{s(i)} \sum_{k \in K} x_{i'i}^k \qquad \forall (i,j) \in A_{mod} \tag{33}$$

, where $\alpha_{s(i)} = \mu_{max,s(i)} \frac{e^{m_{s(i)} t(i',i)} - 1}{m_{s(i)}}$ and $\beta_{s(i)} = (e^{m_{s(i)} t(i',i)} - 1)$.

The new constraint class shown in Equation 33 can be thought of as a modified bundling constraint where there is an upper bound on the sum of the total flow on $(i, j) \in A$ and an a priori fraction of the total flow on $(i', i) \in A$. Since the multi-commodity network flow problem is NP-hard, so is this extended formulation.

Solving the modified optimization formulation, we obtain the network design variables, $y_{\ell,s}\ \forall \ell \in L, s \in S_\ell$. We then evaluate the corresponding re-routing solution by simulating it in the same manner as in section 3.3.4. The only difference is that Algorithm 2 calls the modified simulation Algorithm 10 instead of the original simulation algorithm in Chapter *II*.

4.5 Computational Results

Recall that in section 2.5 we defined a 540 problem instances characterized by crossing 15 traffic instances, 12 instances where the set of candidate trains L contains anywhere from 2 to 13 trains, and 3 disrupted terminals. We have solved the above instances using methods developed in Chapters *II* and *III*. In this chapter we slightly alter the 540 problem instances to reflect linearly declining railcar processing rates at each of the terminals in the network, thus creating 540 modified problem instances. Specifically, for any terminal s in the physical network, we use $\mu_{max} = 60$ and $m = 0.02$.

We have independently employed each of the two methods discussed in the previous section to solve the modified problem instances using computers with 12-core Xeon E5645 CPUs and 48 GBs of RAM. Here we discuss the advantages and disadvantages of each method and directly compare the results obtained using each for identical problem instances.

Let us begin by discussing the solve time of the optimization-based approach. Figure 39 outlines the amount of time the optimization-based approach required for each traffic instance and value of $|L|$. Similarly to the results obtained in Chapter *III* that the number of commodities, defined by traffic instance, we see that the solution time is heavily influenced by the traffic instance number, which dictates the number of commodities in the system during the horizon of interest. In particular:

- Traffic instances with less commodities (#1, #11, #12, #13) solve relatively rapidly.
- The heaviest traffic instances 9 and 10 are not solvable using the given amount of RAM. Furthermore traffic instance 5 seems to run into the same problem for sufficiently large values of $|L|$.

	Traffic Instance Number														
|L|	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	8.75	37.92	45.03	57.44	59.97	38.74	54.15	62.24			7.25	10.23	10.80	37.60	45.48
3	8.91	38.08	44.63	53.74	59.13	38.66	51.14	64.40			7.19	10.91	13.74	37.92	45.98
4	10.73	38.21	44.80	55.41	59.81	41.05	50.73	60.24			7.66	13.12	12.26	38.35	45.50
5	11.66	38.80	45.81	55.09	59.87	41.85	53.50	59.08			7.54	11.59	14.02	38.45	46.30
6	13.18	38.08	46.64	55.21	59.98	41.34	54.34	61.15			8.30	14.76	14.27	38.96	47.29
7	10.55	38.32	44.37	55.05	61.56	42.81	54.15	63.11			9.67	12.29	15.74	39.59	47.40
8	12.49	38.82	47.01	56.73	59.67	42.01	59.53	63.58			9.79	11.98	17.09	42.05	48.39
9	10.78	40.14	47.49	58.73		46.59	59.77	62.82			11.33	12.48	17.60	40.70	49.09
10	14.34	37.81	48.70	56.76		43.92	62.12	65.06			9.83	13.55	15.06	39.89	50.21
11	13.54	37.80	48.97	58.42		44.28	57.47	62.26			11.43	13.08	17.48	40.60	49.78
12	14.25	39.37	49.94	59.05		44.50	59.07				11.05	15.01	17.34	41.29	49.85
13	18.17	41.89	50.10	59.18		45.29	59.80				11.48	14.55	17.37	41.87	51.59

Figure 39: Solve time (minutes) for optimization-based approach

Next we compare the performance of the solution presented by the optimization-based method with that of the heuristic. Given the solve times outlined in Figure 39 we assert that for traffic instances with fewer commodities (#1, #11, #12, #13) the optimization-based results should be compared with the results of the heuristic obtained at the 15-min time cutoff. We present these results in Figures 40 and 41; Figure 40 illustrates the cases where the heuristic method performs better while Figure 41 illustrates the cases where the optimization-based method generally performs better. The results indicate that increasing $|L|$ to 5 or more trains favors the optimization-based method for these "light-commodity" instances. A similar result was obtained in Chapter *III*, where we asserted that 15 minutes

is too little time for the heuristic to move beyond its first iteration, during which it is simply a systematic enumeration method.

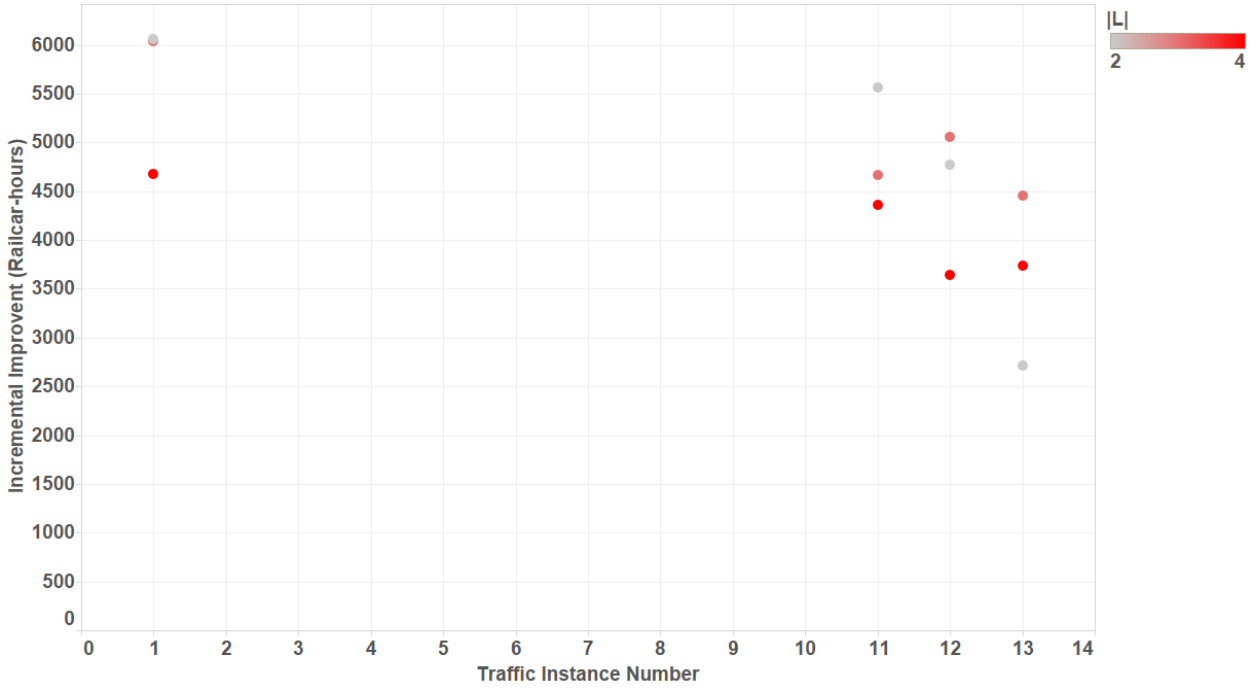


Figure 40: Objective function improvement of Tabu search heuristic solution with 15-minute cutoff, with optimization-based solution as baseline: $|L| \leq 4$

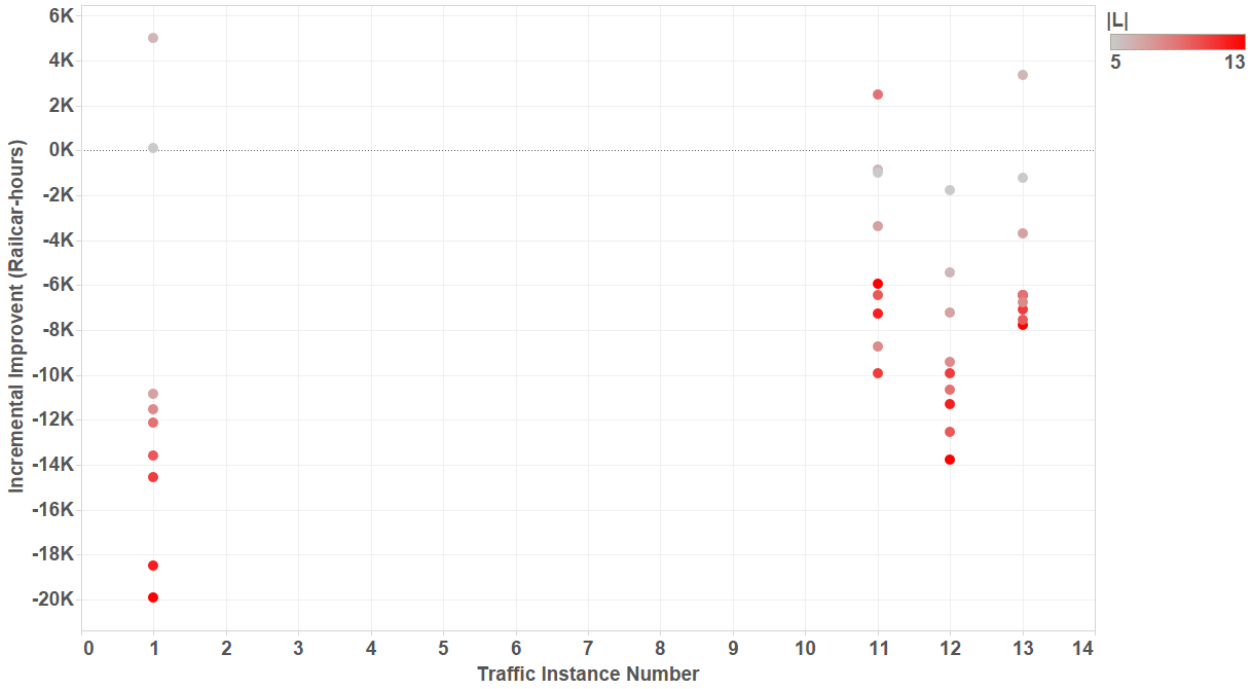


Figure 41: Objective function improvement of Tabu search heuristic solution with 15-minute cutoff, with optimization-based solution as baseline: $|L| \geq 5$

Let us now produce similar graphs for the "heavy-commodity" traffic instances where the heuristic takes considerably more than 15 minutes. Here we compare the results of the optimization-based method with those of the heuristic method with a 1-hour time cutoff, detailed in Figures 42, 43, and 44.

In these graphs we once again see the pattern where the optimization-based method is favored for its performance as $|L|$ increases. As the cutoff time increases however, the additional time allows for the heuristic to produce better results for values of $|L|$ up to 8.

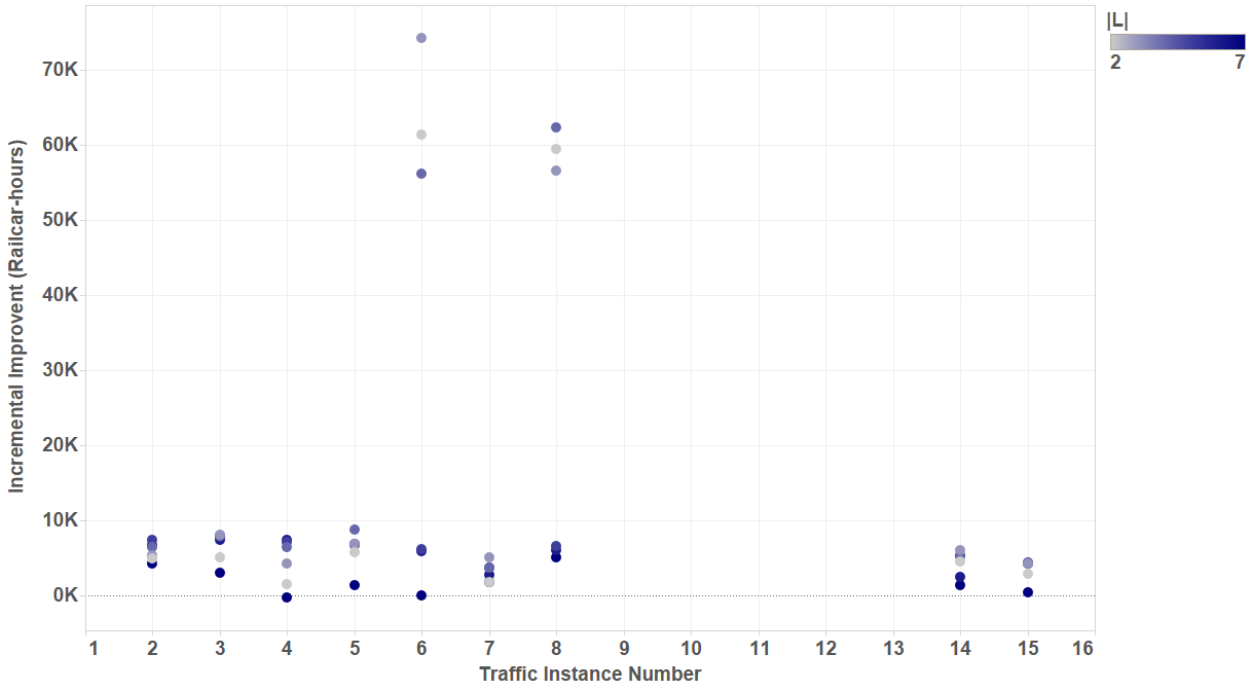


Figure 42: Objective function improvement of Tabu search heuristic solution with 1-hour cutoff, with optimization-based solution as baseline: $|L| \leq 7$

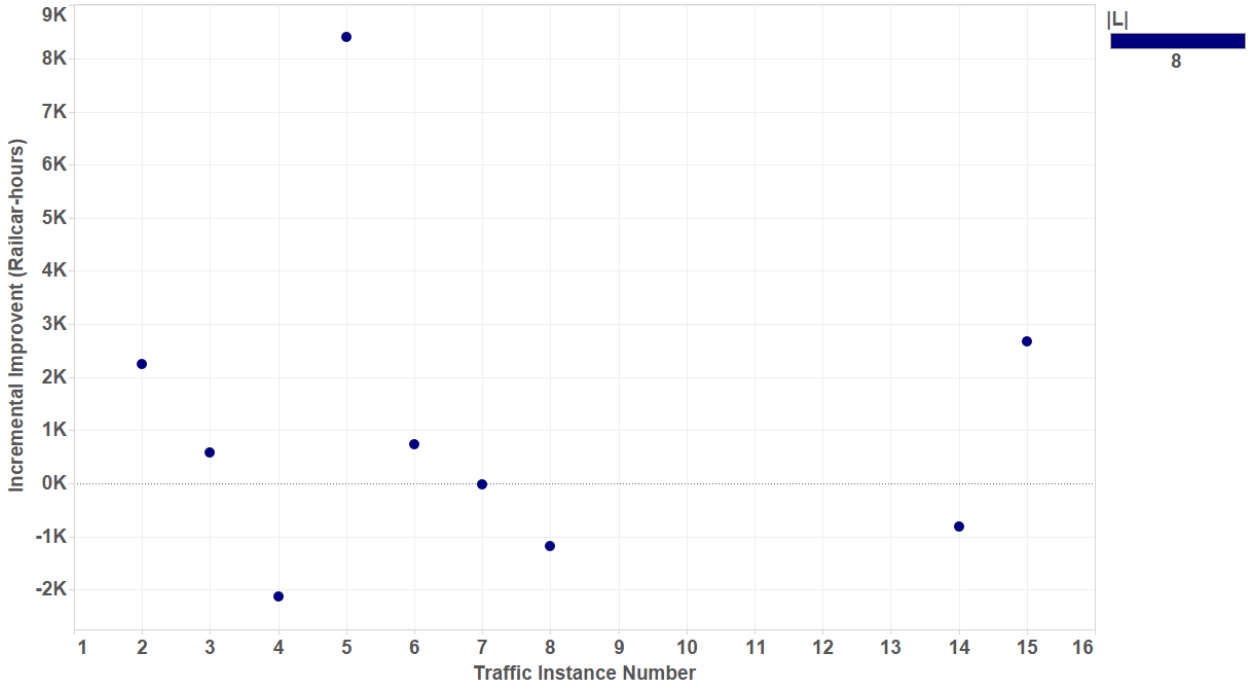


Figure 43: Objective function improvement of Tabu search heuristic solution with 1-hour cutoff, with optimization-based solution as baseline: $|L| = 8$

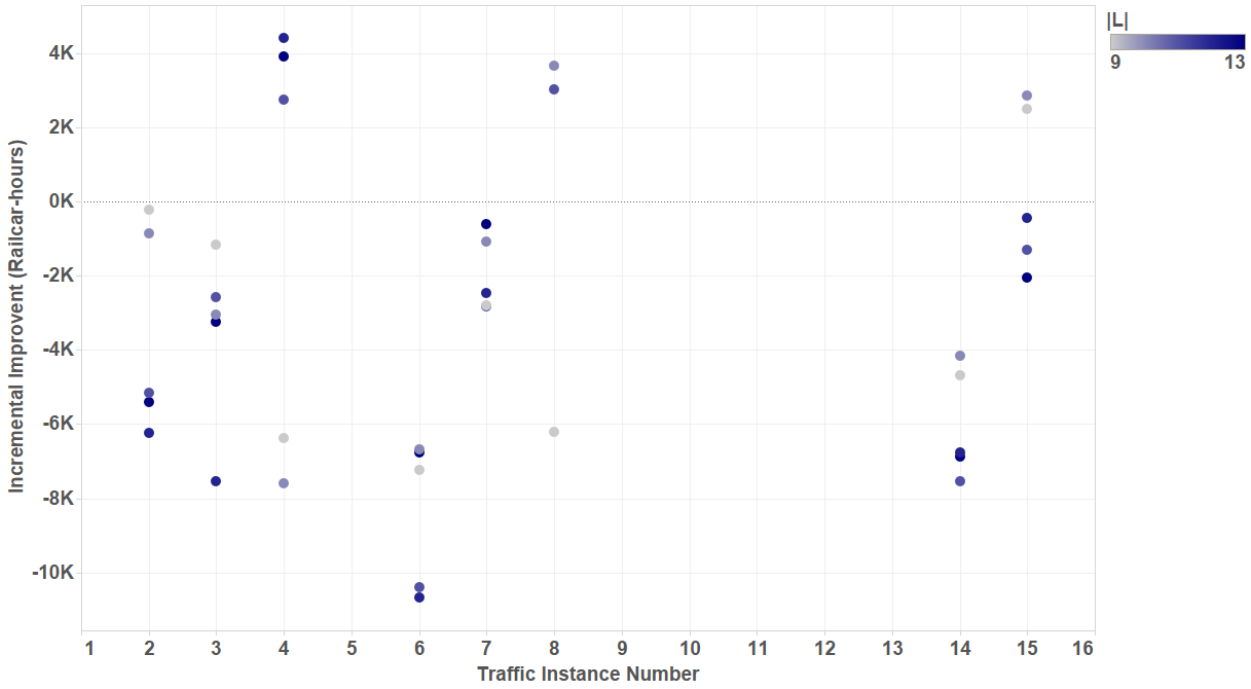


Figure 44: Objective function improvement of Tabu search heuristic solution with 1-hour cutoff, with optimization-based solution as baseline: $|L| \geq 9$

4.6 Conclusion

Having challenged the constant processing rate assumptions of the previous two chapters, we have extended the flexibility of the problem definition and adapted our solution methodologies to solve the modified train re-routing problem. We have established that the linearly decreasing terminal processing rate allows us to maintain the linear structure of the optimization-based solution methodology. Having modified the railcar movement simulation algorithm to reflect the non-constant processing rate has allowed us to apply the Tabu search heuristic to the modified train re-routing problem. The results showed that although the optimization method can be applied to the modified train re-routing problem, the efficacy of this approach is dependent on the shipment volume in the network in the horizon of interest. The optimization proved too cumbersome for the worst-case scenarios of traffic that were constructed. By contrast, the Tabu search heuristic method does not pose this problem. However, for cases where the commodity traffic in the rail network is not particularly

high but the number of candidate trains is higher, the optimization-based method shows significant promise.

4.7 Future Work

Recall that, having derived Equation 32 enabled us to maintain the linear structure of the optimization-based solution methodology as it was extended to solve the train re-routing problem. This was possible since Equation 32 showed the bundling constraints can be generalized with a linear structure, since there is a linear relationship between the upper bound of any processing arc $(i, j) \in A$ and the number of railcars flowing on the terminal storage arc $(i', i) \in A$. This linear relationship cannot be assumed if the railcar processing rate at terminal $s(i)$ declines non-linearly as the number of railcars in the arrival yard increases, which is reflected in the more general case by Equation 20. The reason is two-fold:

1. Recall the linear differential equation quantifying the relationship between number of railcars processed and time available: $\frac{dU_{ij}}{dt(i',i)} = mU_{ij} + \mu_0$. This equation is linear only when processing rate decreases linearly with number of railcars in the arrival yard.

2. The solution to the above equation established a linear relationship between U_{ij} and μ_0. This implies a linear relationship between U_{ij} and $\sum_{k \in K} x^k_{i',i}$ when processing rate decreases linearly with number of railcars in the arrival yard.

As such, when the processing rate declines non-linearly, the optimization-based method loses its linear structure. Further work is required to explore whether the approach can be viable in the non-linear case under certain conditions.

REFERENCES

[1] Ravindra K. Ahuja, Kurt Mehlhorn, James Orlin, and Robert E. Tarjan. Faster algorithms for the shortest path problem. *J. ACM*, 37(2):213–223, April 1990. ISSN 0004-5411. doi: 10.1145/77600.77615. URL http://doi.acm.org/10.1145/77600.77615.

[2] Michael F. Argüello, Jonathan F. Bard, and Gang Yu. A grasp for aircraft routing in response to groundings and delays. *Journal of Combinatorial Optimization*, 1(3):211–228, Oct 1997. ISSN 1573-2886. doi: 10.1023/A:1009772208981. URL https://doi.org/10.1023/A:1009772208981.

[3] PIERRE Arnold, DOMINIQUE PEETERS, ISABELLE THOMAS, and HUGUES MARCHAND. Pour une localisation optimale des centres de transbordement intermodaux entre reseaux de transport: formulation et extensions. *Canadian Geographer / Le Geographe canadien*, 45(3):427–436, 2001. ISSN 1541-0064. doi: 10.1111/j.1541-0064.2001.tb01192.x. URL http://dx.doi.org/10.1111/j.1541-0064.2001.tb01192.x.

[4] Cynthia Barnhart, Hong Jin, and Pamela H. Vance. Railroad blocking: A network design application. *Operations Research*, 48(4):603–614, 2000. doi: 10.1287/opre.48.4.603.12416. URL http://pubsonline.informs.org/doi/abs/10.1287/opre.48.4.603.12416.

[5] R. Bekker, G.M. Koole, B.F. Nielsen, and T.B. Nielsen. Queues with waiting time dependent service. *Queueing Systems*, 68(1):61–78, 2011. ISSN 0257-0130. doi: 10.1007/s11134-011-9225-2. URL http://dx.doi.org/10.1007/s11134-011-9225-2.

[6] Lawrence D. Bodin, Bruce L. Golden, Allan D. Schuster, and William Romig. A model for the blocking of trains. *Transportation Research Part B: Methodological*, 14(1):115 – 120, 1980. ISSN 0191-2615. doi: https://doi.org/10.1016/0191-2615(80)90037-5. URL http://www.sciencedirect.com/science/article/pii/0191261580900375.

[7] A. Caprara, Laura Galli, Leo Kroon, Gabor Maroti, and Paolo Toth. Robust train routing and online re-scheduling. December 2010. doi: 10.4230/OASIcs.ATMOS.2010.24.

[8] Alberto Caprara, Laura Galli, and Paolo Toth. Solution of the train platforming problem. *Transportation Science*, 45(2):246–257, May 2011. ISSN 1526-5447. doi: 10.1287/trsc.1100.0366. URL http://dx.doi.org/10.1287/trsc.1100.0366.

[9] Jordi Castro. A specialized interior-point algorithm for multicommodity network flows. *SIAM journal on Optimization*, 10(3):852–877, 2000.

[10] Teodor Gabriel Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288, 2000.

[11] Teodor Gabriel Crainic and Gilbert Laporte. Planning models for freight transportation. *European journal of operational research*, 97(3):409–438, 1997.

[12] Theodor Gabriel Crainic, Michel Gendreau, and Judith M. Farvolden. A simplex-based tabu search method for capacitated network design. *INFORMS Journal on Computing*, 12(3), 2000.

[13] P.A. Leal de Matos and P.L. Powell. Decision support for flight re-routing in europe. *Decision Support Systems*, 34(4):397 – 412, 2003. ISSN 0167-9236. doi: https://doi.org/10.1016/S0167-9236(02)00066-0. URL http://www.sciencedirect.com/science/article/pii/S0167923602000660.

[14] Guy Desaulniers, Jacques Desrosiers, Yvan Dumas, S Marc, B Rioux, Marius M Solomon, and Francios Soumis. Crew pairing at air france. *European journal of operational research*, 97(2):245–259, 1997.

[15] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, Dec 1959. ISSN 0945-3245. doi: 10.1007/BF01386390. URL https://doi.org/10.1007/BF01386390.

[16] Ales Florian. An efficient sampling scheme: Updated latin hypercube sampling. *Probabilistic Engineering Mechanics*, 7(2):123 – 130, 1992. ISSN 0266-8920. doi: https://doi.org/10.1016/0266-8920(92)90015-A. URL http://www.sciencedirect.com/science/article/pii/026689209290015A.

[17] L.R. Foulds. A multi-commodity flow network design problem. *Transportation Research Part B: Methodological*, 15(4):273 – 283, 1981. ISSN 0191-2615. doi: http://dx.doi.org/10.1016/0191-2615(81)90013-8. URL http://www.sciencedirect.com/science/article/pii/0191261581900138.

[18] Fred Glover. Tabu search - part i. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[19] John J. Jarvis and H. Donald Ratliff. Note - some equivalent objectives for dynamic network flow problems. *Management Science*, 28(1):106–109, 1982. doi: 10.1287/mnsc.28.1.106. URL http://dx.doi.org/10.1287/mnsc.28.1.106.

[20] Daeki Kim. *Large Scale Transportation Service Network Design: Models, Algorithms, and Applications*. PhD thesis, Massachusetts Institute of Technology, 1997.

[21] Natalia Kliewer, Taieb Mellouli, and Leena Suhl. A time-space network based exact optimization model for multi-depot bus scheduling. *European Journal of Operational Research*, 175(3):1616 – 1627, 2006. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2005.02.030. URL http://www.sciencedirect.com/science/article/pii/S0377221705002274.

[22] Edwin Kraft. The yard. *Trains*, 62(7):36, 2002. ISSN 00410934.

[23] Mark William Shannon Land. *Evolutionary Algorithms with Local Search for Combinatorial Optimization*. PhD thesis, University of California San Diego, 1998.

[24] Richard M. Lusby, Jesper Larsen, Matthias Ehrgott, and David M. Ryan. A set packing inspired method for real-time junction train routing. *Comput. Oper. Res.*, 40(3):713–724, March 2013. ISSN 0305-0548. doi: 10.1016/j.cor.2011.12.004. URL http://dx.doi.org/10.1016/j.cor.2011.12.004.

[25] Association of American Railroads. Overview of americas freight railroads. April 2017.

[26] Adamou Ouorou, Philippe Mahey, and J-Ph Vial. A survey of algorithms for convex multicommodity flow problems. *Management science*, 46(1):126–147, 2000.

[27] Michael Berliner Pedersen, Theodor Gabriel Crainic, and Oli B. G. Madsen. Models and tabu search metaheuristics for service network design with asset-balance requirements. *Transportation Science*, 43(2):158–177, 2009.

[28] E. R. Petersen. Railyard modeling: Part i. prediction of put-through time. *Transportation Science*, 11(1):37–49, 1977. doi: 10.1287/trsc.11.1.37. URL http://dx.doi.org/10.1287/trsc.11.1.37.

[29] M. Posner. Single-server queues with service time dependent on waiting time. *Operations Research*, 21(2):610–616, 1973. doi: 10.1287/opre.21.2.610. URL http://dx.doi.org/10.1287/opre.21.2.610.

[30] Michael Rhodes. *North American Railyards*. MBI Publishing Company, 2003.

[31] Jay M. Rosenberger, Andrew J. Schaefer, David Goldsman, Ellis L. Johnson, Anton J. Kleywegt, and George L. Nemhauser. A stochastic model of airline operations. *Transportation Science*, 36(4):357–377, 2002. doi: 10.1287/trsc.36.4.357.551. URL https://pubsonline.informs.org/doi/abs/10.1287/trsc.36.4.357.551.

[32] Jay M. Rosenberger, Ellis L. Johnson, and George L. Nemhauser. Rerouting aircraft for airline recovery. *Transportation Science*, 37(4):408–421, 2003. doi: 10.1287/trsc.37.4.408.23271. URL https://doi.org/10.1287/trsc.37.4.408.23271.

[33] Kennth Sorensen, Christine Vanovermeire, and Sylvie Busschaert. Efficient metaheuristics to solve the intermodal terminal location problem. *Computers and Operations Research*, 39(9):2079–2090, 2012.

[34] J. A. Tomlin. Minimum-cost multicommodity network flows. *Operations Research*, 14(1):45–51, 1966. doi: 10.1287/opre.14.1.45. URL http://dx.doi.org/10.1287/opre.14.1.45.

[35] Endong Zhu, Theodor Gabriel Crainic, and Michel Gendreau. Scheduled service network design for freight rail transportation. *Operations Research*, 62(2):383–400, 2014.