8-2018

Modeling and Solution Approaches for Non-traditional Network Flow Problems with Complicating Constraints

Negin Enayaty Ahangar
University of Arkansas, Fayetteville

Modeling and Solution Approaches for Non-traditional Network Flow Problems with Complicating Constraints

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering with concentration in Industrial Engineering

by

Negin Enayaty Ahangar
Sharif University of Technology
Bachelor of Science in Industrial Engineering, 2011
University of Arkansas
Master of Science in Industrial Engineering, 2014

August 2018
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

Kelly Sullivan, Ph.D.
Dissertation Director

Edward Pohl, Ph.D.
Committee Member

Sarah Nurre, Ph.D.
Committee Member

J. Cole Smith, Ph.D.
Ex-officio Member

Abstract

In this dissertation, we model three network-based optimization problems. Chapter 2 addresses the question of what the operation plan should be for interdependent infrastructure systems in resource-constrained environments so that they collectively operate at the highest level. We develop a network-based operation model of these systems that accounts for interdependencies among them. To solve this large-scale model, a solution approach is proposed that relatively quickly generates high-quality solutions to the problem.

Chapter 3 presents a routing model for a single train within a railyard with the objective of minimizing the total length traveled by train. The difference between this problem and the traditional shortest path is that the route must accommodate the length of the train at any time, subject to yard tracks' configuration. This problem has application in the railway industry where they need to solve the single-train routing problem repeatedly for simulations of train movements in large complex yards. We develop an optimal polynomial-time algorithm that solves an important special case of the problem.

Chapter 4 extends the problem defined in Chapter 3 to a two-train routing problem with the objective of minimizing the overall time possible to schedule the routes in a conflict-free manner. We propose a routing problem that indirectly aims to decrease the overall scheduling time for the two trains. We develop a scheduling model that compares the performance of the solution obtained by the proposed routing model with the solutions obtained by solving the problem as two separate single-train yard routing problems. The comparison indicates a better performance obtained by the proposed routing model for specific problems.

Contents

1. Introduction

This dissertation is focused on non-traditional network flow problems with complicating constraints. Modeling approach and solution methodology are proposed to address each problem.

Chapter 2 is an interdependent multi-layered network flow study focused on the allocation of resources to maximize networks' operability. This study focuses on the application of operating interdependent infrastructure systems in a resource-constrained environment. When an event occurs that causes disruptions in these systems, it is vital to provide operation plans and to determine where to route the limited resources available such that the disrupted interdependent system of infrastructures collectively operates at the highest possible level.

This chapter provides a network-based approach to model the operations of interdependent infrastructure systems, where flow corresponds to the delivery of an infrastructure's services, to capture the performance of infrastructures given a set of operational components. In this network-based approach, unmet demand results in inoperable components and therefore loss of performance. We develop a mathematical formulation that plans the flow and network component operability decisions for the interdependent networks in a way that the overall performance is at the highest level. The objective of the formulation measures how well the set of interdependent infrastructure systems are operating by assuming a reward for making components operable and a cost for routing flow. Our model considers cost as a factor in its formulation since the satisfaction of demand or increasing the level of services from indirect routes is less efficient.

Our model has several unique features over previous network-based models of interdependent infrastructure systems: (i) it is a compact formulation; (ii) it has the capability to capture many common types of interdependence; (iii) it allows for non-uniform dependency requirements on the consumption of flow at a given node in a given layer; and (iv) it considers intra-layer dependency within a layer. Computational experiments indicate that failing to model nonuniform dependency requirements may drastically reduce the quality of strategic restoration decisions derived to restore a disrupted collection of infrastructure systems. We prove that this problem

is strongly NP-complete. We contribute valid inequalities for our model and develop a solution method that exploits a high degree of integrality in LP relaxation solutions to obtain strong upper and lower bounds quickly in comparison to CPLEX.

Chapter 3 is a network-based study focused on the routing of a single train within a railyard. One of the fundamental problems faced by rail freight transportation system is how to optimally move cars and shipments through the railyard. Railyard routing is of importance since such a problem must be solved repeatedly in a very short time for simulations of train movements in large complex yards. Therefore an efficient solution method for this problem is of importance. The existing literature on the yard routing problem has not addressed the routing of trains that accounts for the geometry of the track segments in the railyard as well as the fact that the train occupies space in the railyard.

This chapter presents a model to route a single train specified by its origin, destination, and length within a railyard subject to the geometry of the track segments. The objective of the model is to minimize the total length traveled by train to reach its destination. In this problem, the railyard is modeled as a network where nodes correspond to switches in the yard, and edges correspond to track segments. The proposed approach models the problem in two stages: a preprocessing stage that takes as input the specific length of the train and tracks' configuration to determine the feasibility of certain movements for the train in the railyard and a routing stage that finds a shortest origin-destination route given the output of the preprocessing stage. The routing stage of the problem is modeled as a shortest path problem which can be solved with classical shortest path algorithms. We prove that the preprocessing stage is NP-complete and develop an algorithm and an integer programming formulation to solve it. A linear programming formulation is developed that solves an important special case of the preprocessing stage in polynomial time which leads to an efficient solution method for the whole problem.

The proposed yard routing model is validated on a real railyard dataset. As the solution time is of importance in yard routing, our computational experiments point out the quick computational time needed for the approach to identify an optimal solution for the special case that

happens in real yard networks.

Chapter 4 extends the work in Chapter 3 to routing two trains in a railyard. With the increase in railway traffic and its growing demand, it is necessary to exploit the capacity of the current railway network as efficiently as possible since extending the rail tracks to enhance the railway infrastructure is very cost-intensive or may not be feasible in some areas. One of the challenges in doing so is planning a conflict-free movement of trains through a railyard. In the problem of routing trains in a yard network, there are two important types of decisions: the routing plan assigns trains to routes and the scheduling plan which assigns blocking times to the track segments and nodes in each route used by a train. These two components are not independent of each other as conflicts might be avoided by making changes to both of them. Looking for a conflict-free schedule requires keeping track of all the trains at all times which is computationally difficult. In this chapter, we propose the routing and scheduling problems to be solved separately to decrease the overall computational difficulty. The routing component is solved first which is not constrained by the scheduling components. Given the two routes obtained by the routing model, the scheduling model generates a conflict-free scheduling plan with minimum overall time.

We develop a model to route two trains specified by its origin, destination, and length within a railyard subject to the geometry of the track segments. The objective of the model is to indirectly decrease the overall scheduling time by accounting for the waiting times the train have on their own routes to avoid conflict. We also develop a scheduling model that takes as input the routes for the two trains to determine a conflict-free scheduling plan that minimizes the overall time. This model includes three sets of constraints: (1) the constraints to plan each individual route; (2) the constraints that enforce the minimum headway time between the two trains passing the same location in the railyard ; and (3) the constraints to prevent the two trains to pass through each other in the railyard. Computational experiments demonstrate that this approach, in generating a routing plan, has a better total travel time when compared to the case where we generate the train routes individually as a single-train yard routing problem in the yard network.

2. Interdependent Multi-layered Network Flow Problem

2.1 Introduction

Critical infrastructure systems are becoming increasingly dependent on each other to improve their performance and operational efficiency. Recent advancement in our information technology is one reason that interdependencies among these systems have increased (Pederson et al., 2006; Rinaldi et al., 2001). For example, telecommunication and power systems have a cyber-physical interdependency; telecommunication systems need power to operate while the power systems depend on the control signals which require the communication system. These interdependencies emerge in different forms such as physical flow of commodities and flow of information.

While interdependencies improve operational efficiency in infrastructure systems, they also increase system vulnerability. Recent natural events such as hurricanes, floods, earthquakes, and terrorist attacks, such as the 2001 World Trade Center attack, have shown that interdependencies among critical infrastructures magnify the impact of the initial failure and increase system vulnerability. The interconnectedness and mutual interdependence among critical infrastructures has become so significant that disruption in one may lead to disruptions in others causing cascading and catastrophic failures. The 2003 Northeast US power blackout (Liscouski and Elliot, 2004), the 2003 Italy power blackout (Buldyrev et al., 2010), Hurricane Sandy in 2012 (Beheshtian et al., 2018), and Hurricanes Harvey, Irma and Maria in 2017 (Nateghi, 2018) are examples of cascading disruptions across infrastructures. Additionally, restoring infrastructure systems after a disaster has become more difficult due to increased interconnectedness. The increase in vulnerability of our critical infrastructures is one motivation for studying these interdependent systems.

The application of our work to infrastructure interdependence is preceded (and motivated) by a body of literature that identifies and analyzes interdependencies in infrastructure systems. Rinaldi et al. (2001) classify interdependencies among infrastructure systems into physical, cy-

ber, geographical, and logical interdependencies. This classification is commonly used by other researchers as a criterion to compare different modeling approaches in this area of study. In addition, Rinaldi (2004) discusses complicating factors associated with modeling infrastructure interdependencies and suggests there may be no single modeling methodology that can address all of them. This statement is backed by a review of the literature on modeling interdependent infrastructures, which we now summarize.

Infrastructure interdependencies have been examined through a number of different modeling approaches. A number of these models are based upon the Leontief input-output model (Leontief, 1951), a framework used to study the interconnectedness among the sectors of an economy. In this model, the economy consists of n interacting sectors, each producing one product, and the output of one sector is proportional to its input. Haimes and Jiang (2001) propose a Leontief-based input-output model of infrastructure interdependencies that assumes each infrastructure has a set of inputs and is the input for a set of other infrastructures. The theory and methodology of the model was discussed later by Haimes et al. (2005). The proposed model can consider the interconnectedness among infrastructures as well as the intraconnectedness within each of them. A weakness of this modeling approach is that the model cannot be used to analyze the interdependencies at the infrastructure component level (Ouyang, 2014).

This weakness motivates the use of network-based models to analyze interdependent infrastructure systems. These approaches represent each infrastructure by a network, where nodes are the components of the infrastructure and arcs either represent the relational connections among components or the physical components of the infrastructure. Lee et al. (2007) propose an Interdependent Layered Network (ILN) mathematical formulation which can be used to model the services delivered by interdependent infrastructure systems. Recently, the planning emphasis of decision makers has shifted from prevention, protection, and hardening of infrastructure systems for disruptive events to resilience and recovery when there are disruptions (Baroud et al., 2014; Pant et al., 2014).

Research in the area of *recovery* includes efforts to reestablish the distribution of services disrupted after a disaster through restoration of infrastructure (McLay, 2015). Similar to the interconnectedness and interdependencies among infrastructure systems, restoration efforts of one infrastructure have dependency on other infrastructures' restoration planning. Sharkey et al. (2016) introduce the new concept of restoration interdependencies among interdependent infrastructures and classify them.

Some researchers define restoration efforts as the allocation of available resources to install temporary components or to repair the damaged ones in an infrastructure to reestablish the distribution of its disrupted services. With this definition there are two sets of decisions in restoration planning: determining the set of components that are going to be operational over the restoration process (i.e., design) and planning the installation or repair tasks (i.e., scheduling). Thus, Nurre et al. (2012) propose the use of an integrated network design and scheduling (IDNS) problem to model restoration activities in infrastructures.

A downside of INDS is that it does not consider the interdependencies among infrastructure networks. The ILN model proposed by Lee et al. (2007), which accounts for interdependencies, allows restoration planning by determining a set of components that need to be installed or repaired. However, their model does not take into account the cost of restoration and it does not give any information about when to perform the restoration tasks. Gong et al. (2009) present a modeling framework to schedule the optimal solution provided by the ILN model. The scheduling involves the assignment of repair tasks to work groups and sequence of activities to reestablish the distribution of services provided by the network.

There has been further research on incorporating restoration planning to the network-based operation model of interdependent infrastructure systems. Cavdaroglu et al. (2013) propose an integrated model that incorporates the restoration planning as well as its scheduling. The objective of their model measures how well the services are restored during the restoration effort.

González et al. (2016) introduce an interdependent network design model that accounts for interdependencies based on colocation and limited availability of resources. Sharkey et al. (2015)

provide an extension to the formulation proposed by Cavdaroglu et al. (2013) to consider different classes of restoration interdependencies discussed in Sharkey et al. (2016). Sharkey et al. (2015) also investigate the effect of decentralized restoration efforts (i.e., each infrastructure controls its own efforts independently) as well as value of sharing information across infrastructures to coordinate efforts.

2.2 Problem Definition

This chapter provides an approach to model the operations of interdependent infrastructure systems in order to analyze their performance. We focus on the application of operating interdependent critical infrastructure systems in a resource constrained environment. After a chemical, biological, radiological, nuclear, or high-yield explosive (CBRNE) event or natural disaster, damage to infrastructure may prevent the satisfaction of all demanded services. It is vital to provide operation plans and to determine where to route the limited resources available such that the disrupted interdependent system of infrastructures collectively operates at the highest possible level. We use a network-based approach to model the operation of interdependent infrastructure systems, where flow corresponds to the delivery of an infrastructure's services, to capture the performance of infrastructures given a set of operational components. In this network-based approach, unmet demand results in inoperable components and therefore loss of performance. We develop a mathematical formulation that plans the flow and network component operability decisions for the interdependent networks. The objective of the formulation measures how well the set of interdependent infrastructure systems are operating by assuming a reward for making components operable and a cost for routing flow. Our model considers cost as a factor in its formulation since the satisfaction of demand or increasing the level of services from indirect routes is less efficient.

Capturing the performance of an infrastructure given a set of operational components is of main importance in restoration problems which are not computationally tractable for larger networks and more layers (Nurre and Sharkey, 2014). However, the existing literature (Cavdaroglu et al., 2013; González et al., 2016; Lee et al., 2007; Sharkey et al., 2015) has not explored the op-

erational problem's structure in detail, focusing instead on resource allocation decisions required to implement the restoration efforts. We contribute a thorough analysis of the operations of interdependent networks in resource constrained environments. It is our hope that as a result of this research, the operational problem is easier to solve thereby enabling better performance when including these developments in restoration problems.

The main contributions of this chapter are as follows: (i) our proposed model is stated more compactly than existing interdependent network flow models and has the capability to capture many common interdependence classifications; (ii) our work is the first to allow for either (ii:a) the network dependency on a node to be non-uniform across the network layers or (ii:b) the possibility of dependency within a single layer; (iii) we prove that incorporating this feature causes the decision version of the model to be strongly NP-complete, even the case of a single-layer network; (iv) we contribute valid inequalities for our model and develop a solution method that exploits a high degree of integrality in LP relaxation solutions to obtain strong upper and lower bounds quickly in comparison to CPLEX; and (v) we demonstrate that failing to incorporate (ii:a) could drastically alter and reduce the quality of restoration plans generated by existing optimization models.

2.3 Problem Formulation

The purpose of this section is to provide a mixed-integer programing model of the operations of interdependent infrastructure systems. We refer to the associated problem as the *interdependent multi-layered network flow* (IMN) problem. The IMN problem is to determine the flow of services through interdependent infrastructures systems to maximize their performance less cost.

The IMN model considers a set of directed networks $G^k = (N^k, A^k),\ k \in K$—hereafter referred to as *layers*—whose node sets are overlapping. Associated with each layer is a *flow*, i.e., an assignment of values to $x^k \in \mathbb{R}_+^{|A^k|}$ for each $k \in K$ that satisfies flow balance and capacity constraints on G^k. Each layer's flow is a separate *commodity* in the sense that it has its own supply

and demand nodes, and flow does not cross layers. We may therefore refer to the commodity associated with layer $k \in K$ as commodity k. As in a minimum-cost flow (MCF) model, we associate a linear cost with flow of each commodity. In addition to the multi-commodity aspect, IMN generalizes MCF in the following ways: (1) each node may become *inoperable* in one or more layers—no layer-k flow is permitted through any inoperable node in layer k; (2) there is only a soft requirement to satisfy demands within each layer—this is accomplished by assigning a reward for each operable node in each layer and forcing each node $i \in N^k$ to become inoperable in layer k if its layer-k demand is not satisfied; and (3) each node $i \in N^k \cap N^{k'}, k \neq k'$, will additionally become inoperable in layer k' if the *imbalance* at node i in layer k (i.e., $\sum_{j \in N^k:(j,i) \in A^k} x^k_{ji} - \sum_{j \in N^k:(i,j) \in A^k} x^k_{ij}$) is less than a threshold value.

We now define the sets, parameters and variables used in the IMN problem.

SETS

- K : the set of layers
- N^k : the set of nodes in layer k
- A^k : the set of arcs in layer k
- $G^k = (N^k, A^k)$: the directed network representing layer k

PARAMETERS

- $u^k_{ij}, (i,j) \in A^k$: the flow capacity on arc $(i,j) \in A^k$
- $c^k_{ij}, (i,j) \in A^k$: the cost per unit flow of commodity k on arc $(i,j) \in A^k$
- $f^k_i, i \in N^k$: the reward when node $i \in N^k$ is operable in layer k
- $d^{kk}_i, i \in N^k$: the supply (if $d^{kk}_i < 0$) or demand (if $d^{kk}_i > 0$) of commodity k at node $i \in N^k$
- $d^{kk'}_i, i \in N^k \cup N^{k'}, k \neq k'$: the required consumption of commodity k at node i for node i to be operable in network layer k'

VARIABLES

- x_{ij}^k : the flow on arc $(i,j) \in A^k$
- y_i^k : whether (1) or not (0) node i is operable in layer k

The mixed-integer formulation of the IMN problem is given by:

$$\text{maximize} \quad \sum_{k \in K} \sum_{i \in N^k} f_i^k y_i^k - \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij}^k x_{ij}^k \tag{2.1a}$$

$$\text{subject to} \quad \sum_{j:(i,j) \in A^k} x_{ij}^k - \sum_{j:(j,i) \in A^k} x_{ji}^k \leq -d_i^{kk'} y_i^{k'} \qquad \forall k,k' \in K, \forall i \in N^k \cap N^{k'} \tag{2.1b}$$

$$x_{ij}^k \leq u_{ij}^k y_i^k \qquad \forall k \in K, \forall (i,j) \in A^k \tag{2.1c}$$

$$x_{ij}^k \leq u_{ij}^k y_j^k \qquad \forall k \in K, \forall (i,j) \in A^k \tag{2.1d}$$

$$y_i^k \in \{0,1\} \qquad \forall k \in K, \forall i \in N^k \tag{2.1e}$$

$$x_{ij}^k \geq 0 \qquad \forall k \in K, \forall (i,j) \in A^k. \tag{2.1f}$$

Objective (2.1a) seeks to maximize the reward obtained from activating network components less the total cost of flow across all networks. Constraint (2.1b) ensures the flow balance at each node in each layer, effectively requiring that the net flow into node i in each layer k is at least $d_i^{kk'}$ if flow of commodity k' is permitted through node i, i.e., node i is operational in layer k'. This constraint guarantees that if node i is to be operable in layer k', then node i must consume the required $d_i^{kk'}$ units of each commodity $k \in K$. Constraint (2.1c) and (2.1d) simultaneously ensure that (i) flow on arc (i,j) is permissible only if both nodes i and j are operable in layer k and (ii) all flows satisfy capacity requirements. Constraint (2.1e) and (2.1f) define the binary and non-negativity restrictions on the decision variables.

Before describing specific relationships of Model (2.1) to the literature on interdependent networks, we first provide some additional discussion regarding the types of interdependency that can be incorporated via the d-parameters, a unique feature that allows our model to incorporate many types of interdependence. If $d_i^{kk} < 0$, the right-hand side (r.h.s.) of Constraint (2.1b) cor-

responding to node i and layer $k' = k$ takes a positive value (unless node i is forced inoperable in layer k via its dependence on some other layer), and node i therefore acts as a supply node for commodity k. If $d_i^{kk} > 0$, the r.h.s. of this constraint takes a positive value, and it therefore acts as a demand node for commodity k. Unless this demand is satisfied (i.e., the left-hand side of this constraint is no more than $-d_i^{kk}$), then y_i^k must equal zero. A positive value of d_i^{kk} imposes a form of *intralayer dependence*—commodity k may not enter or leave node i unless d_i^{kk} units of the commodity will be consumed at node i. *Interlayer dependence* can be incorporated via positive-valued $d_i^{kk'}$, $i \in N^k \cap N^{k'}, k \neq k'$: From Constraint (2.1b) corresponding to $i \in N^k \cap N^{k'}$, node i must become inoperable in layer k' unless node i consumes $d_i^{kk'}$ units of commodity k. One potential application of this feature is a power network where the physical components need to consume power (e.g., to operate sensors) in order to distribute or produce power.

2.4 Model Capabilities

Previous models on the operations of infrastructure systems or coordinating restoration efforts (Cavdaroglu et al., 2013; Gong et al., 2009; González et al., 2016; Sharkey et al., 2015) follow the structure of the ILN model proposed by Lee et al. (2007). The following list, identified by Lee et al. (2007) provides a structure to explain how IMN can capture different types of interdependence. These types of interdependence imply that an impact on one infrastructure system is also an impact on one or more other infrastructure system. All of these forms of dependence can be incorporated into IMN as well, as we now summarize.

Input dependence: This interdependency is one of the important operational interdependencies where a component of an infrastructure requires the services of other infrastructures to be operational. For example, power is needed at a central office in the telecommunications infrastructure to ensure proper operations (Cavdaroglu et al., 2013). This type of dependence is directly modeled in IMN via the $d_i^{kk'}$ parameter. If $d_i^{kk'} > 0$, then node i must consume flow in layer k in order to be operable in layer k'.

Mutual dependence: Two infrastructures are said to be mutually interdependent if both are de-

pendent on each other, meaning an output of an infrastructure is an input to another infrastructure and vice versa. This type of dependence can be incorporated into IMN by setting both $d_i^{kk'} > 0$ and $d_i^{k'k} > 0$ for some node $i \in N^k$.

Shared dependence: The application of this type of dependence is when physical components of some infrastructures used in providing the services are shared. To model this dependence, an arc that is used in different layers is shared. This type of dependence would add a limit on the total flow of a set of layers $R \subseteq K$ for the arcs with shared capacity. Shared dependence can be incorporated by adding constraints of the form,

$$\sum_{k \in R} x_{ij}^k \leq u_{ij}^R \tag{2.2}$$

in the IMN model. An application of this extension to the IMN formulation is in modeling multiple commodities flowing in each network layer where each arc's capacity is shared across the set of commodities. That is, let W^ℓ represent a group of layers with the same network topology (i.e., (N^k, A^k, c^k) are the same for each $k \in W^\ell$), such that $\cup_{\ell \in L} W^\ell = K$. Let u_{ij}^ℓ be the flow capacity on arcs (i, j) in group ℓ. Then the multicommodity IMN model can be obtained by replacing Constraints (2.1c)–(2.1d) in Model (2.1) with

$$\sum_{k \in W^\ell} x_{ij}^k \leq u_{ij}^\ell y_i^k, \ \forall k \in K, \forall \ell \in L, \forall (i, j) \in A^k, \tag{2.3a}$$

$$\sum_{k \in W^\ell} x_{ij}^k \leq u_{ij}^\ell y_j^k, \ \forall k \in K, \forall \ell \in L, \forall (i, j) \in A^k. \tag{2.3b}$$

In this version of the model, arc capacity constraints on individual commodities can be captured by defining groups with a singleton layer. In general this type of dependence does not require the restriction that all layers have all of the same topology as it is defined on components level. This general case can also be incorporated in the IMN formulation.

Exclusive-or dependence: Physical components of a set (R) of infrastructures are allowed to

provide one of the services at a time. This type of dependency would add a constraint of the form

$$\sum_{k \in R} y_i^k \leq 1, \tag{2.4}$$

in the IMN model if either arc (i, j) or node i is to provide one service at a time.

Colocated dependence: This type of interdependency would add constraints on the condition of the components situated within a prescribed geographical region. The colocated dependence can be applied in constructing the IMN formulation where physical components of different infrastructures, such as nodes and arcs, exist in multiple network layers.

As compared to previous network-flow-based models of infrastructure interdependencies, IMN offers several computational and theoretical advantages. Whereas Lee et al. (2007) and Matisziw et al. (2010) utilize an exponential model that enumerates o-d paths in order to represent operational dependencies, the IMN formulation is polynomial in the number of network components (i.e., nodes and arcs). Additionally, IMN improves upon the (single time-period version of the) operational dependency models used in Cavdaroglu et al. (2013), Paredes and Dueñas-Osorio (2015), and Sharkey et al. (2015) in that the number of binary variables is reduced by a factor of K, the number of network layers. The operational dependency models of Lee et al. (2007), Cavdaroglu et al. (2013), Sharkey et al. (2015), and Shen (2013) utilize a representation of dependency that requires enumerating dependencies among all pairs of components that exist in different layers. By contrast, our model takes a simpler form that allows dependencies across layers only within a node that exists in both layers. The resulting model can be stated more compactly, thereby leading to easier theoretical analysis; furthermore, as we now demonstrate, the simplification is made without loss of generality. Toward this end, suppose node i in layer k needs to consume d units of flow in order for node j in layer k' to be operational. To incorporate this dependency into our model, we define a new layer k'' with node set $\{i, j, s\}$ and arc set $\{(s,i),(i,j)\}$. Node s has one unit of resource in layer k'' (i.e., $d_s^{k''k''} = -1$). Layer k'' is dependent on layer k at node i with d units of flow (i.e., $d_i^{kk''} = d$). Layer k' is dependent on layer k''

at node j with one unit of flow (i.e., $d_i^{k''k'} = 1$). These settings will model the general case where there is dependency between uncommon nodes in two layers.

In addition to the advantages described above, IMN has some practical advantages over previous network-based models of operational dependency. In modeling input dependency, previous work assumed that the dependent nodes (child nodes) are all either operational or not, depending on the unmet demand of the node (parent node) that the child nodes depend upon (Cavdaroglu et al., 2013; Lee et al., 2007; Sharkey et al., 2015). This way of modeling the input dependency can be captured as a special case of the IMN model where the dependence parameters of the child nodes on the parent node are equal. Suppose the parent node i is in layer k and a set (R) of layers depend on the unmet demand (d_i^{kk}) of the parent node, then $d_i^{kk'} = d_i^{kk}$, $\forall k' \in R$. However, the IMN model generally allows child nodes to require different levels of support from a common parent node. For instance, the power requirement for a node in infrastructure A can be different from its power requirement in infrastructure B. The interdependent network design model proposed by González et al. (2016) also allows partial functionality in the set of child nodes but the interdependency defined in their model is based on the idea that the functionality of the child node depends on the functionality of other nodes not the flow going through them.

In summary, IMN generalizes the functionality of previous operations models of interdependent infrastructure systems in a compact way. It allows for different dependency requirements for dependent components with a common parent component, and it incorporates the new feature, interlayer dependence. We now proceed by discussing the complexity of IMN due to this new feature.

2.5 Complexity

We now establish the complexity of the decision version of Model (2.1) (which we denote as IMN-MCFd) based on a reduction from the "exact cover by 3-sets" problem (denoted as X3C). Towards this end, we first define the two problems:

IMN-MCF Decision Problem (IMN-MCFd)

Question: Given $H \in \mathbb{R}$ and an instance of Model (2.1), does there exist a feasible solution to Model (2.1) with objective value at least H?

Exact cover by 3-sets (X3C)

Instance: A finite set $Q = \{q_j : 1 \leq j \leq 3m\}$ and a collection $\mathcal{C} = \{C_i : 1 \leq i \leq n\}$ of three-element subsets of Q.

Question: Does $\mathcal{C}$ contain an *exact cover*, i.e., a sub-collection $\mathcal{C}^\star \subseteq \mathcal{C}$ such that each element of Q is included in exactly one element of $\mathcal{C}^\star$?

We now establish strong NP-completeness of IMN-MCFd when $|K| = 1$ based on a reduction from X3C, which is known to be strongly NP-complete (Garey and Johnson, 1979). Towards this end, we first discuss implications of Constraints (2.1b)–(2.1d) on feasible flows. When $x_{ij}^{k'} > 0$ for some $k' \in K$ and $(i,j) \in A^k$, Constraints (2.1c)–(2.1d) imply that $y_i^{k'} = y_j^{k'} = 1$. Due to Constraints (2.1b), it follows that $\sum_{j:(j,i)\in A^k} x_{ji}^k - \sum_{j:(i,j)\in A^k} x_{ij}^k \geq d_i^{kk'}$, $\forall k \in K$, and $\sum_{i:(i,j)\in A^k} x_{ij}^k - \sum_{i:(j,i)\in A^k} x_{ij}^k \geq d_j^{kk'}$, $\forall k \in K$. In particular, in the case of a single layer network (i.e., $K = \{1\}$), these relationships imply that d_i^{11} and d_j^{11} units of flow must be consumed at nodes i and j respectively, if any flow is to be transported along arc (i,j). The following proof draws upon this relationship to establish strong NP-completeness of the single-layer version of IMN-MCFd.

Theorem 1 *IMN-MCFd is NP-complete.*

Proof. Inclusion in NP is clear due to the fact that the objective value and all constraints of the IMN model can be computed in polynomial time.

We prove NP-completeness of IMN-MCFd when $|K| = 1$ via reduction from X3C. Given an instance of X3C, define an instance of IMN-MCFd as follows: Construct node sets $U = \{v_i : 1 \leq i \leq n\}$ and $V = \{v_{n+i} : 1 \leq i \leq 3m\}$. The single layer G^1 of the transformed network is defined

by setting $N^1 = \{s,t\} \cup U \cup V$ and $A^1 = D \cup E \cup F$, where

$$D = \{(s, v_i) : 1 \leq i \leq n\},$$
$$E = \{(v_i, v_{n+j}) : 1 \leq i \leq n,\ q_j \in C_i\},$$
$$F = \{(v_{n+i}, t) : 1 \leq i \leq 3m\}.$$

Capacities $u^1_{vv'}$ are assigned as follows: Define all arcs in D as uncapacitated, and let arcs in $E \cup F$ have unit capacity. Assign a supply of $d^{11}_s = -4m$ units at node s, and let $c^1_{vv'} = 0,\ \forall (v,v') \in A^1$. Define $f^1_t = 1$ and $f^1_i = 0,\ \forall i \in N^1 \setminus \{t\}$, and let $H = 1$ so that IMN-MCFd seeks only to identify whether it is possible to make node t active.

Summarizing the notation defined thus far, a feasible solution to this instance of IMN-MCFd consists of an s-t flow, plus some additional flow that is consumed to satisfy dependency requirements $d^{11}_i,\ i \in N$. We define these dependency requirements now. First, let $d^{11}_t = 3m$. Thus, if node t is to be active, it must be possible to send at least $3m$ units of flow from s to t subject to capacities on arcs $E \cup F$. (Since $|F| = 3m$ and $u^1_{vv'} = 1,\ \forall (v,v') \in F$, this is only possible if $x^1_{vv'} = u^1_{vv'},\ \forall (v,v') \in F$.) Now, let $d^{11}_v = 1,\ \forall v \in U$, and $d^{11}_v = 1$ for all other vertices v; hence, in sending $3m$ units of flow from s to t, at most m vertices $v \in U$ can be used because each one consumes one unit of flow and only $4m$ units are supplied at s. Figure 2.1 illustrates the construction of G^1, with dependency relationships illustrated by dashed arcs.

We now prove that existence of an s-t flow of value $3m$ (yielding $y^1_t = 1$) implies that $\mathcal{C}$ contains an exact cover of Q. Suppose such a flow exists, and note (by the argument given in the previous paragraph) that this flow passes through at most m nodes—say $v_i,\ i \in I$—in the set U. Noting (by capacities on arcs E) that at most 3 units of flow may pass through any node $v_i,\ i \in I$, it follows that $|I| = m$, and each arc $(v_i, v_{n+j}),\ i \in I,\ q_j \in C_i$ carries unit flow. However, noting that arcs $(v_{n+j}, t),\ 1 \leq j \leq 3m$ have unit capacity, this flow can reach the sink only if all subsets $C_i,\ i \in I$ are non-overlapping. Because the collection $\mathcal{C}^\star \equiv \{C_i : i \in I\}$ is pairwise mutually exclusive and $\sum_{i \in I} |C_i| = 3m$, $\mathcal{C}^\star$ is an exact cover of Q and a solution to X3C.

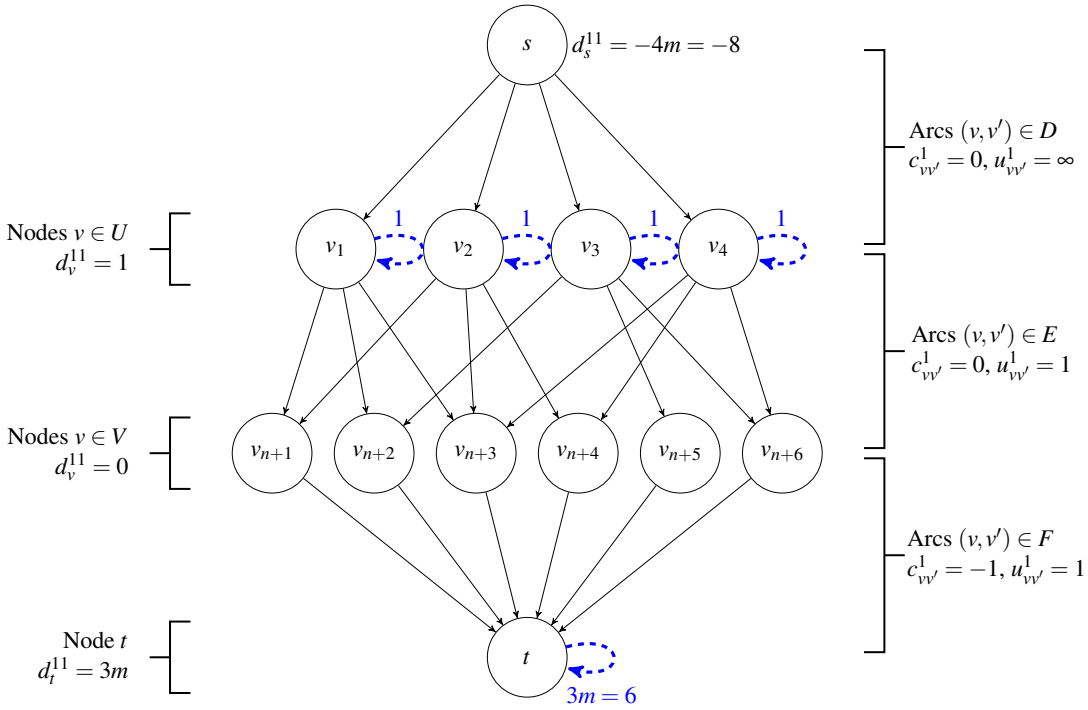


Figure 2.1: Construction of IMN-MCFd instance corresponding to an instance of X3C with $Q = \{1,2,3,4,5,6\}$, $C_1 = \{1,2,3\}$, $C_2 = \{1,3,4\}$, $C_3 = \{2,5,6\}$, and $C_4 = \{3,4,6\}$. A dashed loop from a node v onto itself indicates $d_v^{11} > 0$. Each such loop is labeled with the corresponding value d_v^{11}.

Conversely, suppose $\mathcal{C}^\star \equiv \{C_i : i \in I\}$ is an exact cover by 3-sets. Then by applying the logic of the previous paragraph in reverse order—a solution to IMN-MCFd can be constructed by activating nodes $v_i,\ i \in I$ (and supplying these nodes with the required flow), assigning unit flow to arcs $(v_i, v_{n+j}),\ i \in I,\ q_j \in C_i$, and assigning appropriate flow to arc sets D and F as required to satisfy flow balance. □

This proof establishes NP-completeness for IMN-MCFd even when (i) the network contains only a single layer and (ii) the network is acyclic. The NP-completeness of IMN problem motivates our development of valid inequalities in the following section.

2.6 Model Improvement

This section is focused on identifying classes of valid inequalities in order to strengthen the bounds provided by the linear programming relaxation of the MIP formulation.

2.6.1 Mixing Inequalities

Let $0 \equiv a_0 \leq a_1 \leq \cdots \leq a_m$ and consider the set

$$Z = \{(\psi, z) \in \mathbb{R} \times \{0,1\}^m | \psi \geq a_j z_j, \forall j = 0, ..., m\}, \tag{2.5}$$

where $z_0 \equiv 0$ (arbitrarily). Define $M = \{0, ..., m\}$ and let $H \subseteq M$ such that $\{0\} \subset H$. Letting $q \equiv |H| - 1$, number the elements of H as $H = \{h(0), h(1), ..., h(q)\}$ such that $h(0) = 0$, $h(q) = m$, and $h(0) < h(1) < ... < h(q)$. The so-called mixing inequality

$$\psi \geq \sum_{i=1}^{q} \left(a_{h(i)} - a_{h(i-1)}\right) z_{h(i)}, \tag{2.6}$$

due to Günlük and Pochet (2001), is known to be valid for Z. This class of valid inequalities has been implemented to tighten MIP formulations incorporating the quantity $\max\{a_j z_j | z_j \in \{0,1\}^m\}$. See, for example, the "step inequalities" of Morton et al. (2007) and Sullivan et al. (2014), which specialize to the mixing inequalities for a special case of the models examined. We augment Model (2.1) by generating inequalities corresponding to each $\psi_i^k \equiv -\sum_{j:(i,j)\in A^k} x_{ij}^k + \sum_{j:(j,i)\in A^k} x_{ji}^k$, where Constraints (2.1b) used to define the set Z associated with a given $k \in K$ and $i \in N^k$. Specifically, number the elements of $\bar{K}$ as $\bar{K} = \{\bar{k}(0), \bar{k}(1), \bar{k}(2), ..., \bar{k}(|K|)\}$ such that $\bar{K} = \{0, 1, ..., K\}$ and $d_i^{k\bar{k}(1)} \leq d_i^{k\bar{k}(2)} \leq ... \leq d_i^{k\bar{k}(|K|)}$ and $\bar{k}(0) = 0$. Define Z by setting $m = K$, $a_j = d_i^{k\bar{k}(j)}$ and $z_j = y_i^{\bar{k}(j)} \quad \forall j = 1, ..., m$. Inequality (2.6) specializes to mixing inequality

$$-\sum_{j:(i,j)\in A^k} x_{ij}^k + \sum_{j:(j,i)\in A^k} x_{ji}^k \geq \sum_{s=1}^{q} (d_i^{k\bar{k}(h(s))} - d_i^{k\bar{k}(h(s-1))}) y_i^{\bar{k}(h(s))}, \tag{2.7}$$

and must therefore be valid for Model (2.1) for each $H \subseteq \{0,1,\ldots,|K|\}$, $\{0\} \subset H$, because Z corresponds directly to the set of Constraints (2.1b) and therefore contains the feasible region of Model (2.1). Additionally, because the Inequality (2.7) generated under $H \subseteq \{0,1,\ldots,|K|-1\}$ is dominated by the Inequality (2.7) generated under $H' \equiv H \cup \{|K|\}$, we may restrict the choice of H to those subsets of $\{0,1,\ldots,|K|\}$ that contain both 0 and $\bar{k}(|K|)$. We now illustrate Inequality (2.7) for our model.

Example. Suppose node i in layer k has a demand of $d_i^{kk} = 5$. Additionally, suppose layers k' and k'' depend on layer k at node i such that $d_i^{kk'} = 10$ and $d_i^{kk''} = 20$.
If H is defined as $\{k,k'\}$, the Inequality (2.7) is

$$-\sum_{j:(i,j)\in A^k} x_{ij}^k + \sum_{j:(j,i)\in A^k} x_{ji}^k \geq 5y_i^k + (10-5)y_i^{k'}. \tag{2.8}$$

If H is defined as $\{k,k''\}$, the Inequality (2.7) is

$$-\sum_{j:(i,j)\in A^k} x_{ij}^k + \sum_{j:(j,i)\in A^k} x_{ji}^k \geq 5y_i^k + (20-5)y_i^{k''}. \tag{2.9}$$

If H is defined as $\{k',k''\}$, the Inequality (2.7) is

$$-\sum_{j:(i,j)\in A^k} x_{ij}^k + \sum_{j:(j,i)\in A^k} x_{ji}^k \geq 10y_i^{k'} + (20-10)y_i^{k''}. \tag{2.10}$$

If H is defined as $\{k,k',k''\}$, the Inequality (2.7) is

$$-\sum_{j:(i,j)\in A^k} x_{ij}^k + \sum_{j:(j,i)\in A^k} x_{ji}^k \geq 5y_i^k + (10-5)y_i^{k'} + (20-10)y_i^{k''}. \tag{2.11}$$

Table 2.1 shows the minimum net flow requirement and the right hand sides of the four inequalities for node i in layer k for different combinations of the y-variables. These inequalities are valid since their right hand side is less than equal to the minimum net flow requirement. Also

Table 2.1: Inequality (2.11)

Combinations	Minimum net flow	r.h.s. (2.8)	(2.9)	(2.10)	(2.11)
$y_i^k = y_i^{k'} = y_i^{k''} = 0$	0	0	0	0	0
$y_i^k = 1, y_i^{k'} = y_i^{k''} = 0$	5	5	5	0	5
$y_i^k = 1, y_i^{k'} = y_i^{k''} = 1$	20	10	20	20	20
$y_i^k = 1, y_i^{k'} = 0, y_i^{k''} = 1$	20	5	20	10	15
$y_i^k = 1, y_i^{k'} = 1, y_i^{k''} = 0$	10	10	5	10	10

it can be seen that Inequality (2.8) is dominated by Inequality (2.11) which justifies why we force subset H to include $\bar{k}(|K|)$.

When K is large, the number of subsets H will increase exponentially. In this case, we can define a separation problem that could be used within a branch-and-cut approach to identify a most-violated inequality of the form (2.7) as Morton et al. (2007) and Sullivan et al. (2014) described for their step inequalities.

For $H \subseteq \{0,1,\ldots,|K|\}$, $\{0,\bar{k}(|K|)\} \subseteq H$, let $g_i^k(H;y)$ denote the r.h.s. of Inequality (2.7). Given an LP solution $(\hat{x},\hat{y})$ to a node in the branch-and-bound tree, the separation problem identifies a subset $H \subseteq \{0,1,\ldots,|K|\}$, $\{0,\bar{k}(|K|)\} \subseteq H$ that maximizes $g_i^k(H;\hat{y})$ in order to generate a most-violated mixing inequality of the form

$$-\sum_{j:(i,j)\in A^k} x_{ij}^k + \sum_{j:(j,i)\in A^k} x_{ji}^k \geq g_i^k(H;\hat{y}), \tag{2.12}$$

or prove that no such inequality is violated by $(\hat{x},\hat{y})$. As we will demonstrate, the maximum value of $g_i^k(H;\hat{y})$ corresponds to the case where each element $j \in H \setminus \{0\}$ is such that $\hat{y}_i^{\bar{k}(j)} \geq \hat{y}_i^{\bar{k}(j')}$, $\forall j' = j+1, j+2, \ldots, \bar{k}(|K|)$. Algorithm 1 provides a polynomial separation routine based upon this observation.

Theorem 2 *Algorithm 1 identifies an inequality of the form* (2.12) *that is most violated with respect to a solution* $(\hat{x},\hat{y})$ *or establishes that no such inequality is violated.*

Proof. Let $H^* = \{h^*(0), h^*(1), ..., h^*(q)\}$ be the set obtained with Algorithm 1 such that $d_i^{k\bar{k}(h^*(0))} \leq$

Algorithm 1: Identify most-violated mixing inequality with respect to solution $(\hat{x},\hat{y})$

1 $c=0$, $q=0$, $H=\{0\}$
2 **while** $q<s$ **do**
3 $\quad q=\text{argmax}_j\{\hat{y}_i^j | j\in\{c+1,2,...,s\}\}$
4 $\quad H=H\cup\bar{k}(q)$
5 $\quad c=q$
6 **end**
7 **Return** H

$d_i^{k\bar{k}(h^*(1))}\leq ...\leq d_i^{k\bar{k}(h^*(q))}$. Let $H=\{h(0),h(1),...,h(r)\}\subseteq\{0,...,m\}$ be another set such that $d_i^{k\bar{k}(h(0))}\leq d_i^{k\bar{k}(h(1))}\leq ...\leq d_i^{k\bar{k}(h(r))}$. We show that for any set H, the inequality generated by H does not dominate the inequality generated by H^*. Suppose we have a case in which $h^*(j)<h(a)<h^*(j+1)<h(a+1)$. We show that for the interval $[d_i^{k\bar{k}(h^*(j))},d_i^{k\bar{k}(h^*(j+1))}]$, $\hat{y}_i^{\bar{k}(h^*(j+1))}$ increases $g_i^k(H^*;\hat{y})$ at least as much as $\hat{y}_i^{\bar{k}(h(a))}$ and $\hat{y}_i^{\bar{k}(h(a+1))}$ increase $g_i^k(H;\hat{y})$. Based on line 3 of the Algorithm 1, we know that $\hat{y}_i^{\bar{k}(h^*(j+1))}\geq\hat{y}_i^{\bar{k}(h(a))}$ and $\hat{y}_i^{\bar{k}(h^*(j+1))}\geq\hat{y}_i^{\bar{k}(h(a+1))}$. Then

$$(d_i^{k\bar{k}(h(a))}-d_i^{k\bar{k}(h^*(j))})\hat{y}_i^{\bar{k}(h^*(j+1))}\geq(d_i^{k\bar{k}(h(a))}-d_i^{k\bar{k}(h^*(j))})\hat{y}_i^{\bar{k}(h(a))} \tag{2.13}$$

$$(d_i^{k\bar{k}(h^*(j+1))}-d_i^{k\bar{k}(h(a))})\hat{y}_i^{\bar{k}(h^*(j+1))}\geq(d_i^{k\bar{k}(h^*(j+1))}-d_i^{k\bar{k}(h(a))})\hat{y}_i^{\bar{k}(h(a+1))} \tag{2.14}$$

By summing 2.13 and 2.14, we have

$$(d_i^{k\bar{k}(h^*(j+1)}-d_i^{k\bar{k}(h^*(j))})\hat{y}_i^{\bar{k}(h^*(j+1)}\geq(d_i^{k\bar{k}(h(a))}-d_i^{k\bar{k}(h^*(j))})\hat{y}_i^{\bar{k}(h(a))}+(d_i^{k\bar{k}(h^*(j+1))}-d_i^{k\bar{k}(h(a))})\hat{y}_i^{\bar{k}(h(a+1)} \tag{2.15}$$

which shows that for the interval of interest, H^* increases $g_i^k(H^*;\hat{y})$ as much as any other set H increases $g_i^k(H;\hat{y})$. This proof is for the case that one element of the set H is between two consecutive elements of the set H^*. Other cases, there may be $0,2,...,r-1$ elements of the set H between two consecutive elements of H^*. The proof for these cases are analogous. Therefore, for the interval $[0,d_i^{k\bar{k}(|K|)}]=\bigcup_{j=0}^{q}[d_i^{k\bar{k}(h^*(j))},d_i^{k\bar{k}(h^*(j+1))}]$, H^* increases the $g_i^k(H^*;\hat{y})$ at least as much as any other set H increases $g_i^k(H;\hat{y})$, meaning $g_i^k(H^*;\hat{y})\geq g_i^k(H;\hat{y})$. □

2.6.2 Precedence Inequalities

The second set of valid inequalities identified for Model (2.1) are logical inequalities stating that a dependent node i can be operational if and only if it is operational in all the supporting layers. This valid inequality is

$$y_i^{k'} \leq y_i^k, \ \forall i \in N^k \cap N^{k'}, \ d_i^{kk'} > 0. \tag{2.16}$$

Inequality (2.16) is valid because $d_i^{kk'} > 0$ implies the imbalance of node i in layer k must be at least $d_i^{kk'}$ if i is to be operable in layer k' (i.e., $y_i^{k'} = 1$), but this is only possible if flow is permitted to enter node i in layer k. Therefore, if $y_i^{k'} = 1$, we must also have that $y_i^k = 1$, and the inequality is therefore valid. Preliminary computational experience suggests that this valid inequality is likely to be violated by an optimal solution to the LP relaxation of IMN for values of $i \in N^k \cap N^{k'}$ such that solutions for the scenarios in which $d_i^{kk} > d_i^{kk'} > 0$. We now demonstrate that the LP relaxation of Model (2.1) may have an optimal solution that violates Inequality (2.16).

Example. Consider a node i that appears in two layers, k and k', such that $d_i^{kk} = 10$ and $d_i^{kk'} = 5$. Suppose these are the only non-zero d-values associated with node i. Furthermore, suppose node i carries a positive reward in both layers, i.e., $f_i^k > 0$ and $f_i^{k'} > 0$. Let $e^k(i) = -\sum_{j:(i,j)\in A^k} x_{ij}^k + \sum_{k:(j,i)\in A^k} x_{ji}^k$ denote the imbalance of commodity k at node i. Constraints (2.1b) associated with $e^k(i)$ reduce to $e^k(i) \geq 10y_i^k$ and $e^k(i) \geq 5y_i^{k'}$. Note that y_i^k and $y_i^{k'}$ appear elsewhere in the model only in (1) the objective, with a positive coefficient, and (2) in Constraints (2.1c)–(2.1d), on the right-hand side of a "$\leq$" constraint. For fixed $e^k(i)$, it will therefore be optimal to set $y_i^k = \min\{1, e^k(i)/10\}$ and $y_i^{k'} = \min\{1, e^k(i)/5\}$ in the LP relaxation. Therefore, if $e^k(i) = 5$ in the solution to the LP relaxation, then $y_i^k = 0.5$ and $y_i^{k'} = 1$, violating Inequality (2.16).

In Section 2.8, we show that the two valid inequalities do not directly enhance the CPLEX solver. However, our experiments revealed that these valid inequalities increase the percentage of binary variables that take integer values in solutions to the LP relaxation of the MIP, which CPLEX does not exploit to quickly find better solutions. This observation motivates the solution

approach we present in the following section.

2.7 Solution Method

The purpose of this section is to develop a solution approach for the IMN problem. Anticipating the difficulty of solving the IMN problem due to the strong NP-completeness classification, an efficient solution approach is necessary to decrease the computational effort. Based upon preliminary computational experiments, we did not observe the valid inequalities presented in Section 2.6 to be effective at improving solution times of the MIP formulation. What we did observe, however, is that the valid inequalities both (i) strengthen the bound provided by the LP relaxation of the MIP and (ii) increase the percentage of binary variables that take integer values in solutions to the LP relaxation of the MIP. The latter observation motivated the solution approach that we describe in this section. We will summarize experimental results to demonstrate (i) and (ii) in Section 2.8. Before doing so, we first develop the solution approach that was motivated by these observations.

The main idea in our solution approach, which is inspired by the approach in Helber and Sahling (2010), is to solve a sequence of LP relaxations to IMN in an iterative fix-and-optimize approach to find feasible solutions. The "fix" part of the approach is determined after solving the LP relaxation of Model (2.1), augmented with any valid inequalities we choose to add. The "optimize" part of the approach solves an additional optimization program, this time an integer program, to determine if it is possible to convert the fractional solution resulting from the "fix" step into an integer solution without modifying any of the variables that were integral in the fractional solution. Because solutions to the LP relaxation, augmented with our inequalities, tend to have a high concentration of integer-valued variables, the additional effort that we do in the "optimize" part is not significant.

In our implementation of this algorithm, we terminate after identifying any integer-feasible solution. Our reason for doing this is because our experience suggests that this usually happens within a few iterations of our algorithm, and the resulting solution usually has an objective value

within 1% of the LP bound. It should be noted that, although our implementation does provide both an upper and lower bound on the optimal objective value, we have not guaranteed convergence to a global optimal solution because we terminate at the first found feasible solution.

We term this approach Iterative Fractional Remover (IFR) because in each iteration we seek to remove fractional LP relaxation solutions if they cannot be extended into feasible integer solutions. In Algorithm 4, we formalize the IFR approach. In each iteration, we first solve the LP relaxation of Model (2.1) plus inequalities (2.7) and (2.16). If the solution of this step is integral, then the optimal solution is found and we stop. Otherwise, we solve Model (2.1) with additional constraints that fix a subset of binary variables to the values determined in the LP relaxation solution if they were integral numbers. We solve the problem over the subset of binary variables that have not been fixed. If this problem is infeasible, a cut will be added to make sure that we cannot fix those binary variables simultaneously. We continue this process until we find a feasible solution to the IP. In line 3 of Algorithm 4, we augment the LP relaxation with the solution eliminating inequality that mandates some y-variable for which $\hat{y}_i^k$ has integer value must be changed to yield any MIP-feasible solution.

Algorithm 2: Iterative Fractional Remover (IFR)

```
1  NotFeasible = true
2  while NotFeasible do
3      Solve the LP relaxation with solution (x̂, ŷ)
4      if the solution is integral then
5          NotFeasible = false
6      else
7          Solve the MIP that results from IMN when the integer-valued y-variables in (x̂, ŷ)
           are fixed
8          if the MIP is feasible then
9              NotFeasible = false
10             Let (x̂*, ŷ*) denote the integer solution to IMN
11         else
12             Add inequality Σ_{k∈K, i∈N^k : ŷ_i^k ∈ {0,1}} y_i^k ≤ |{k ∈ K, i ∈ N^k : ŷ_i^k ∈ {0,1}}| − 1
13         end
14     end
15 end
16 return (x̂*, ŷ*)
```

2.8 Computational Results

This section presents a computational analysis of the proposed model and the IFR solution approach. All tests were conducted on an Intel Xeon E5-2670 2.60 GHz, 32 GB RAM computer using IBM ILOG CPLEX Optimization Studio 12.6.3 to implement the IMN formulation and IFR.

2.8.1 Model Capabilities

The focus of this section is to demonstrate IMN capabilities over previous models on the operations of interdependent infrastructure systems. As discussed in Section 3.2, previous work (Cavdaroglu et al., 2013; Lee et al., 2007; Sharkey et al., 2015) assumed that the dependent nodes (child nodes) are all either operational or not, depending on the unmet demand of the parent node that the child nodes depend upon which can be considered as a special case of IMN when $d_i^{kk'} = d_i^{kk''}\ \forall k, k' \in K \setminus k$. However, IMN generally allows child nodes to require different levels of support from a common parent node which may result in different operational levels in dependent networks and better utilization of available resources.

Consider a 3-layer IMN instance where the only dependency defined in the problem is that layer 2 and 3 depend on layer 1. Each network is generated to have nodes (hubs) with many more connections than others. There is one source node in the first layer which acts as a hub. Layer 1 in this instance is shown in Figure 2.2 with the source node in the middle of the network with green color. The three layers have the same set of 50 nodes but the arc set for each layer is different. The probability of adding an arc between any pair of nodes (i.e., arc density) in all layers are set to 0.04, and the flow cost of each arc is zero. We solve a sequence of 11 IMN instances, $s = 0, 5, 10, ..., 50$, in which all of the instances in the sequence are identical with the exception that the interlayer dependency values (i.e., $d_i^{kk'}, k \neq k'$) differ among instances. Instances $s = 0, 5, 10, ..., 50$ are defined with $s = |d_i^{12} - d_i^{13}|, \forall i \in N^1 \cap N^2 \cap N^3$. Therefore, as s increases one layer becomes more heavily dependent on layer 1 while the dependence of the other layer

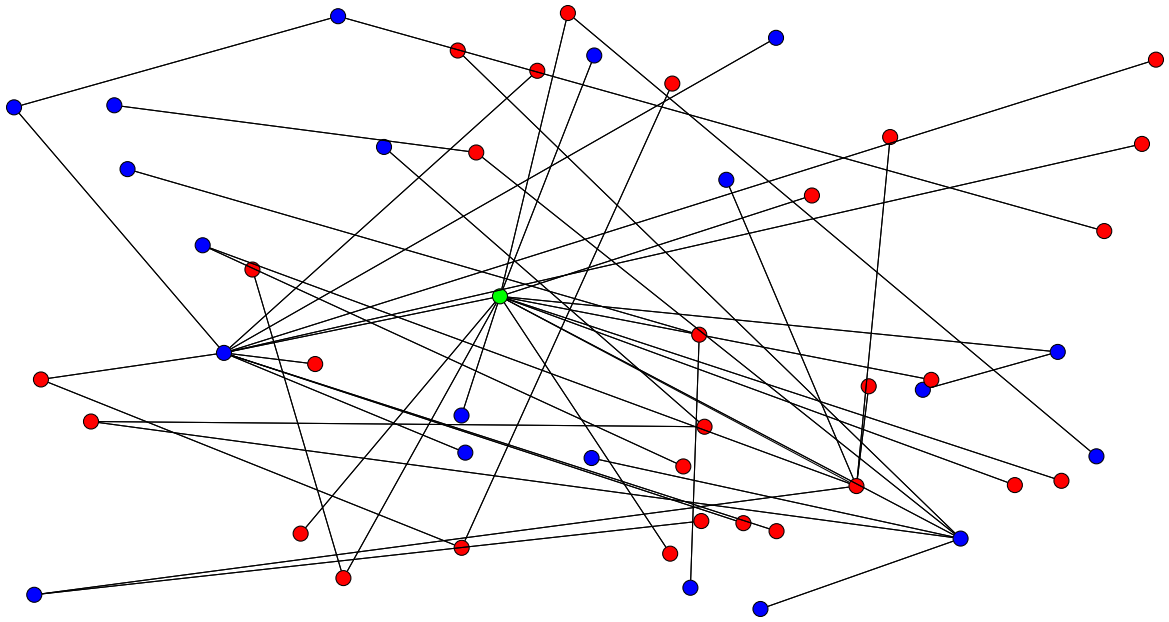


Figure 2.2: Layer 1 - Source node is colored green. Other nodes are colored red and blue according to whether it is the dependency of layer 2 or 3 that is increasing, respectively.

remains constant ($\min\{d_i^{12}, d_i^{13}\} = [8, 12]$). The probability that either one the layers 2 or 3 to be chosen as the layer with a constant dependency on layer 1 is 50% at each node. Hereafter, we denote this first sequence of instances as IMN-Seq(s), $s = 0, 5, 10, ..., 50$. From this sequence of instances, we construct a second sequence of 11 instances PREV-Seq(s), $s = 0, 5, 10, ..., 50$, that would have been solvable using a previously existing model from the literature without restoration decisions (Cavdaroglu et al., 2013; Lee et al., 2007; Sharkey et al., 2015). Specifically, for each $s = 0, 5, 10, ..., 50$, we construct an instance satisfying $d_i^{kk'} = d_i^{kk''}, \forall k, k', k'' \in K, \forall i \in N^k \cap N^{k'} \cap N^{k''}$ by setting $\bar{d}_i^{12} = \bar{d}_i^{13} = \max\{d_i^{12}, d_i^{13}\}$. We refer to $\bar{d}$ as the estimated d parameter. We define instance PREV-Seq(s), $s = 0, 5, 10, ..., 50$, exactly as instance IMN-Seq(s) with the exception that $\bar{d}$ is used instead of d to define the interlayer dependence for each node. We consider two environments for this IMN instance as follows

1. **Resource-constrained environment:** The available resources in layer 1 are 80% of the required amount to activate all the nodes in other layers when s is equal to zero. Arcs are uncapacitated in this environment. Figure 2.3 shows the *optimal operational level* (i.e., the proportion of nodes that are functional in a given layer) for layer 2 and 3 in IMN-Seq and

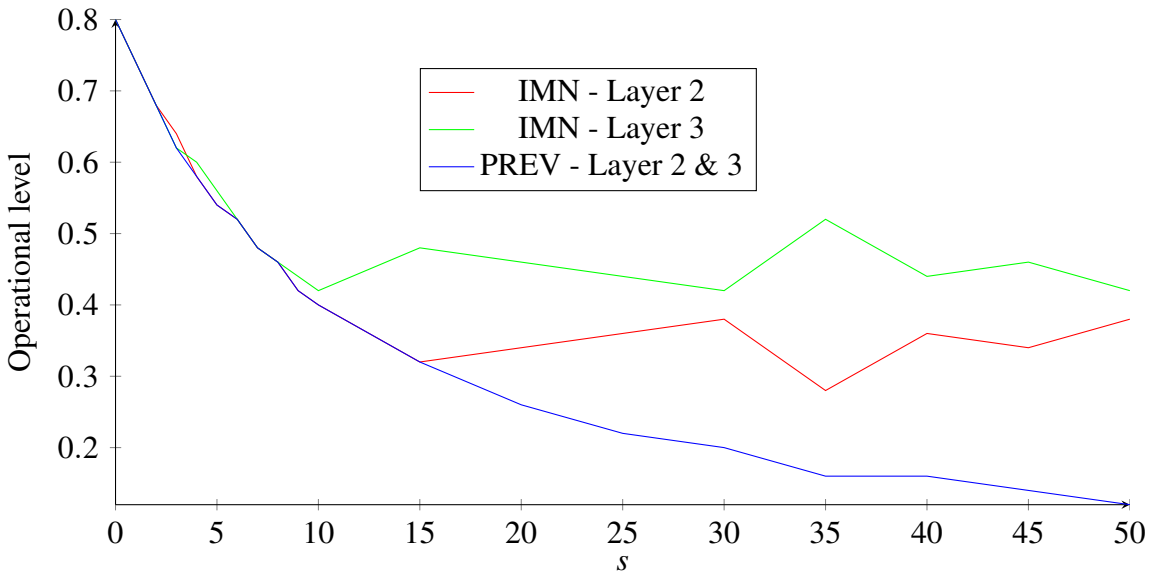


Figure 2.3: Effect of requiring different levels of support from a common system on the solution in a resource-constrained environment

PREV-Seq. The horizontal axis shows the instance number s (i.e., the value of $|d_i^{13} - d_i^{12}|$) for every node in the set $N^1 \cap N^2 \cap N^3$. As seen in Figure 2.3, an increase in s decreases the IMN-Seq operational levels in both layers until the point where the resources are just enough to activate the nodes that have constant dependency (independent of s) on layer 1. However, as s increases in the PREV-Seq, the operational level of both layer 2 and 3 converges to zero since there are not enough resources to satisfy the maximum dependency parameter (i.e., $\max\{d_i^{12}, d_i^{13}\}$). Even though there are enough resources to activate almost 40% of the components, both layers will become inactive which shows that we are not using the resources to increase the overall functionality of the systems.

2. **Arc capacity-constrained environment:** In this case, there are enough resources in layer 1 to activate all the dependent nodes. However, the arc capacity constraints are binding. Similar to Figure 2.3, the horizontal and vertical axes in Figure 2.4 show the variable s and the operational level. As s increases, the solution for PREV-Seq(s) will converge to inactivity of both layer 2 and 3 as the arc capacities do not allow for an increase in the amount of flow to satisfy the demands. However, the arc capacities allow the resource in layer 1 to

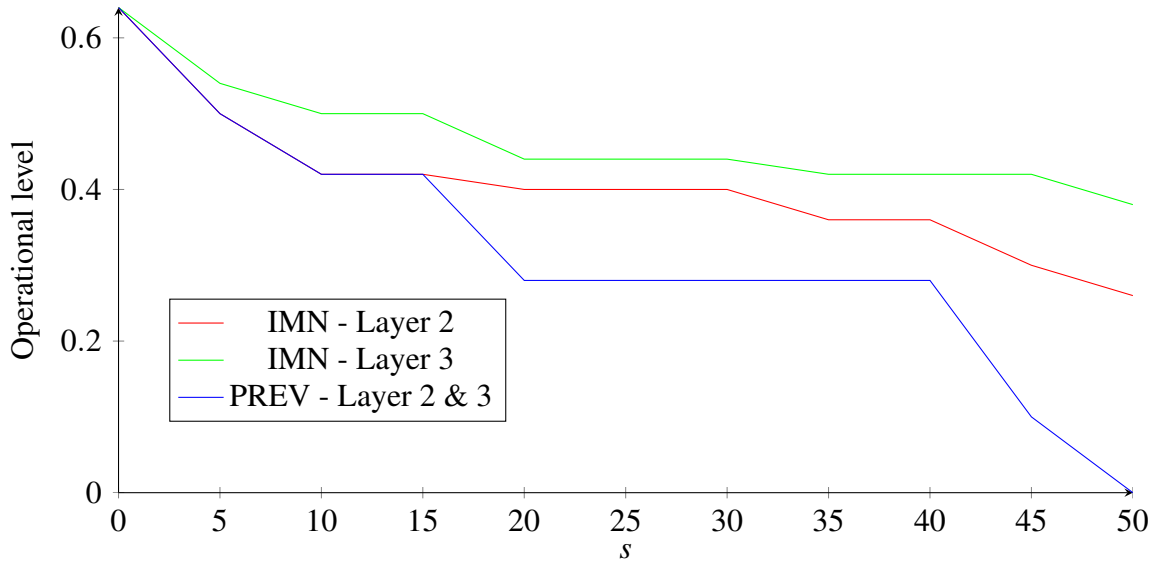


Figure 2.4: Effect of requiring different levels of support from a common system on the solution in an arc capacity-constrained environment

reach the demand nodes with constant dependency (independent of s) which can be captured with the solution of IMN. Therefore, using previously available modeling approaches to solve an instance in an arc capacity-constrained environment may result in not fully exploiting the resource to increase the operational levels for dependent systems.

In summary, applying existing operation models (Cavdaroglu et al., 2013; Lee et al., 2007; Sharkey et al., 2015) to the cases where dependent systems require different levels of support from independent systems may result in solutions that differ drastically from IMN solutions. Now for the same sequences, suppose that half of the arcs in the independent network, layer 1, have been disrupted and we need to determine 5 arcs to install in this layer as a restoration plan to increase the operational efficiency in the dependent networks, layer 2 and 3. One type of restoration problem consists of two stages: (i) selecting components to install and (ii) determining the resulting operational efficiency from the decision made in the first stage. Let the the binary variable z_{ij}^k represent the decision to install the temporary arc (i, j) in layer k. Let $\bar{A}^k$ denote the set of disrupted arcs in layer k. The first stage has a constraint that bounds the number of temporary arcs installed which is written as

$$\sum_{k\in K}\sum_{(i,j)\in\bar{A}^k} z_{ij}^k \leq B. \tag{2.17}$$

There is also a constraint that links together the restoration parts of the model and the network flow parts of the model which is denoted as the following inequality. This constraint, given as

$$x_{ij}^k \leq u_{ij}^k z_{ij}^k,\ \forall k \in K, \forall (i,j) \in \bar{A}^k, \tag{2.18}$$

ensures that flow is not permitted on disrupted arcs that were not repaired.

Using the IMN-Seq and PREV-Seq instance sequences defined in the previous subsection, we solve the restoration problem described above. We then compare the optimal objective value v_s^* for the IMN-Seq instances to the IMN objective value v_s^{IMN} that would result if the z-solution obtained from the PREV-Seq instances is fixed and the resulting IMN instance is re-solved. We then compute the *relative loss of operational efficiency*, defined as $100\%(v_s^* - v_s^{\text{IMN}})/v_s^*$ for each instance $s = 0, 5, 10, ..., 50$. Figure 2.5 shows the relative loss of operational efficiency across the 11 instances. As s increases, the loss increases overall. The inconsistency in the increasing trend is caused by the existence of multiple optimal restoration solutions in the PREV-Seq instances which correspond to different operational efficiencies in the IMN problem. Figures 2.6 and 2.7 show the solutions to $s = 50$ for layer 1 in the resource constrained environment. Green arcs denote the temporary arcs installed and the black arcs have positive flow. Based upon these figures, the restoration solutions for PREV-Seq may prefer to install temporary arcs to connect nodes with smaller estimated d-parameters to gain more reward with the limited resources. With the IMN model, temporary arcs are mostly used to satisfy the demands that are not affected by the value s directly from the resource node to gain the rewards in either layer 2 or 3. In summary, the restoration solution greatly depends on the value of the d-parameters as well as the difference between the d-parameters associated with the two dependent layers at a specific node. The difference in restoration solutions are even more significant if the estimated d-parameters used in the existing

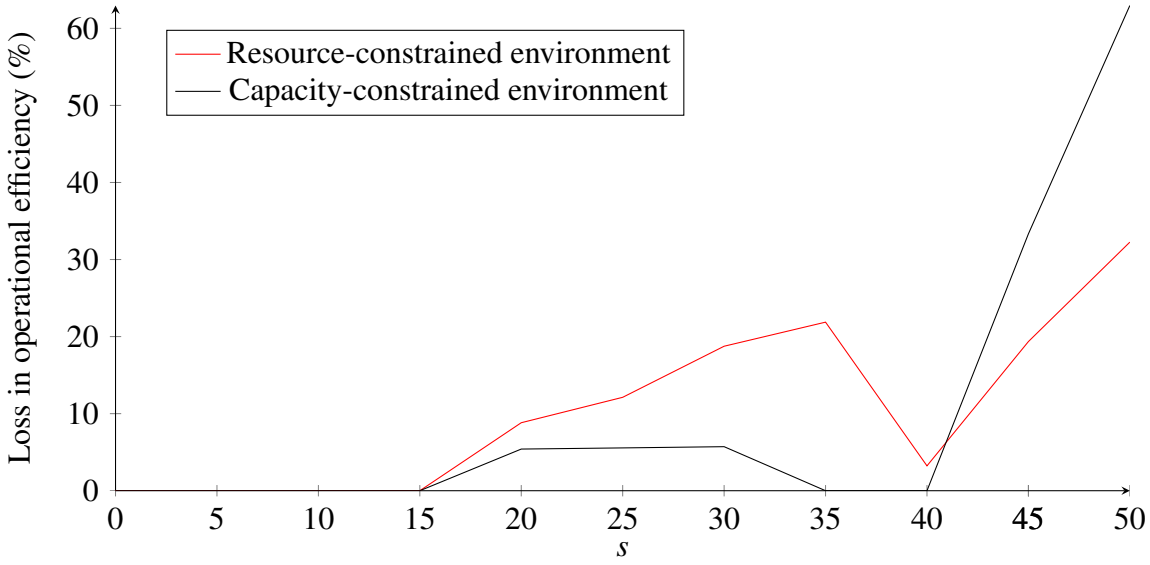


Figure 2.5: Loss in dependent networks operational efficiency due to utilizing previous operational model instead of IMN

models are further from the actual d-parameters of the layers. This difference in restoration solutions can result in a significant reduction in operational efficiency if existing models are used to model non-uniform dependencies.

2.8.2 MIP and IFR

The focus of this section is to (i) demonstrate the effect of valid inequalities on Model (2.1) and (ii) evaluate the performance of the proposed solution approach (IFR) in comparison to the standard MIP solver, CPLEX. This computational analysis will be done by examining synthesized data sets of interdependent networks. The instances are made with different parameter settings to account for their effects on the performance of both solution approaches. Table 2.2 shows the different parameters used to generate these instances.

The structure of networks is either random or scale-free. A scale-free network has nodes (hubs) that have more connections than others which is appropriate to represent real-world networks. We have used the Barabási and Albert (1999) model to create scale-free networks that correspond to a specific arc density. To generate a random network, we add a directed arc between any pair of nodes with a probability that is equal to the arc density. Network layers have

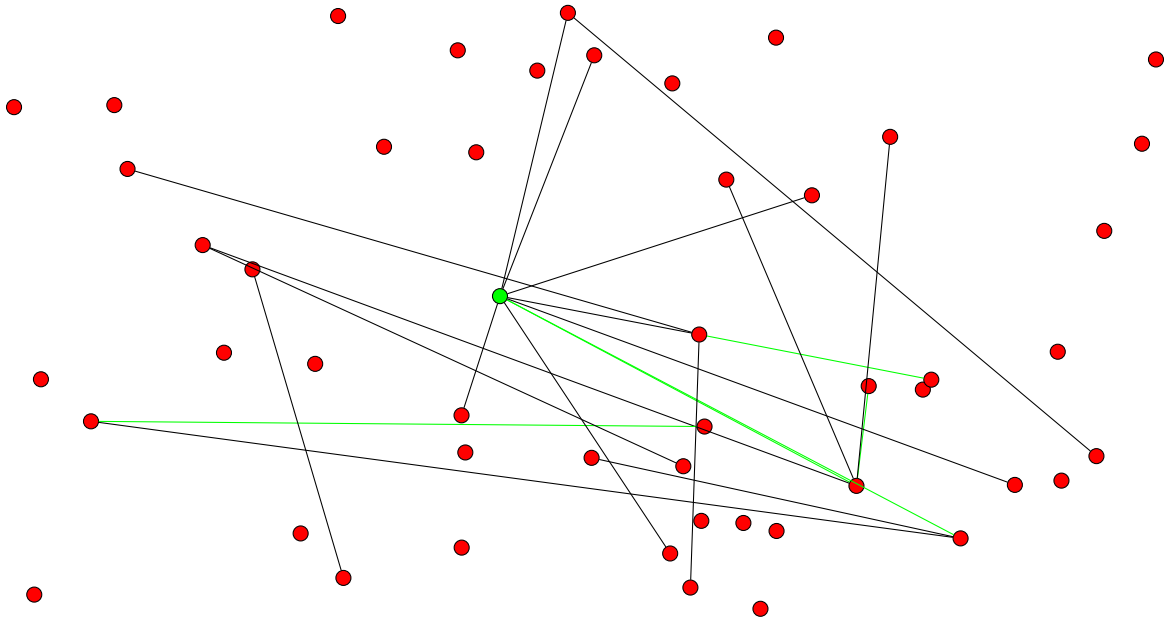


Figure 2.6: Layer 1 restoration solution to instance PREV-Seq(50)

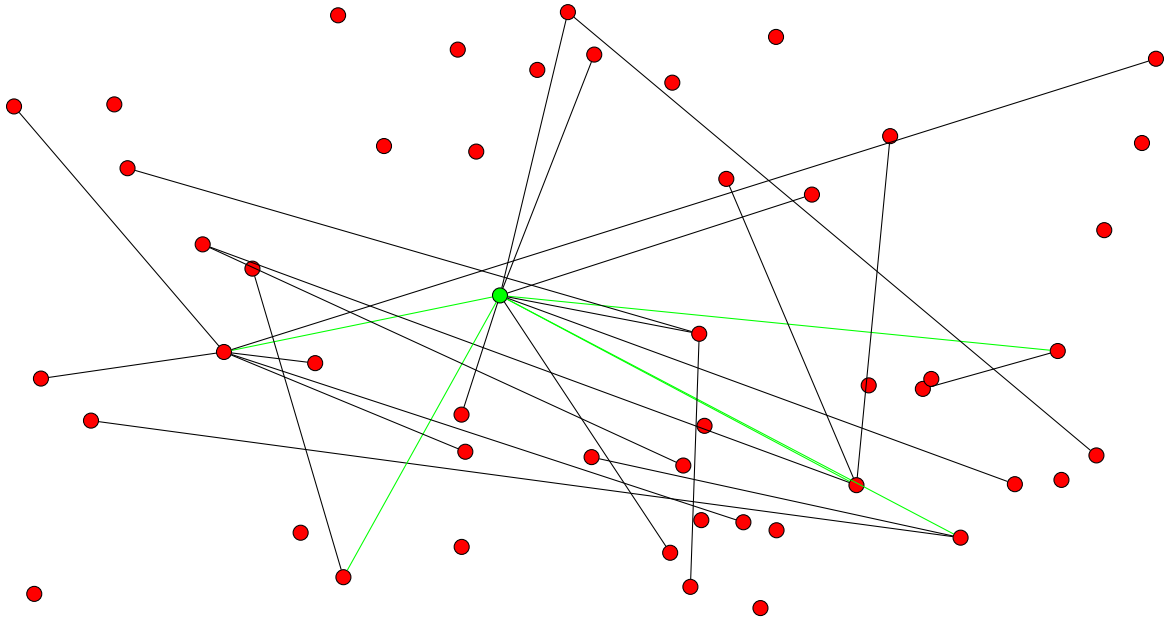


Figure 2.7: Layer 1 restoration solution to instance IMN-Seq(50)

Table 2.2: Parameter settings

Parameter	Settings
Network structure	Random, Scale-free
Interdependency structures	All on one, One on All, Random
Number of layers	$3, 5$
Number of nodes in each layer	2000
Amount of resources in each layer	$50\%, 150\%$
Number of resource nodes in each layer	$4, 10$
Arc capacities	$[10, 15], [20, 25]$
Intralayer dependency probability	$0.1, 0.2, 0.4, 0.5$
Arc density	0.01, 0.05
Arc cost	$[3, 7], [4, 6]$, no cost
Reward (if Arc cost is "no cost")	$[1, 3]$
Reward (if Arc cost is $[3, 7]$ or $[4, 6]$)	$[500, 1000], [1000, 1500], [1500, 2000]$ $[2000, 2500], [2500, 3000]$
Dependency parameter	$[10, 20], [5, 25]$

the same node sets. There are three types of interdependency structures we considered for this computational study: all layers are dependent on one layer (e.g., the dependency of water, telecommunications, and transportation on the electric power infrastructure), one layer is dependent on all other layers (e.g., the dependency of transportation systems on natural gas, oil and electric power systems) and random interdependency. In each layer, the total amount of resources are set to be 50% or 150% of the average total within-layer demand to represent resource-constrained and resource-abundant environments. Arc capacities are generated uniformly from one of the two ranges mentioned in Table 2.2 as preliminary computational experiments revealed that these ranges tend to create binding constraints in Model (2.1). Because cost and reward parameters can be scaled without impacting the set of optimal solutions, we fix the cost and change the reward parameter. Cost parameters are generated uniformly from the two ranges, one of which is a subset of the other. We also consider the case where there is no arc cost. For the instances that do not have an arc cost, we use the range $[1, 3]$ for the node rewards so that we can create a combinatorial problem where only the resources and arc capacities are important in solving the problem. For the cases with positive arc cost, we have used five disjoint reward ranges to account for the effect of increasing node rewards. There are two ranges chosen as the dependency parameter in

which one of them is a subset of the other in order to see the effect of a wider range on the results, especially the solution to the LP relaxation. We have generated 3876 IMN instances after removing 348 instances where both approaches find a solution with objective value equal to zero. The 4224 $(3876 + 348)$ instances are generated with all combination of the parameters listed in Table 2.2 except for the intralayer dependency probability parameter which was equally assigned to the instances.

We first examine the effect of valid inequalities (mixing and precedence) on the solution to the LP relaxation of Model (2.1) by comparing the percentage of binary variables y that are integral in the solution and how it improves the objective value. Since the number of layers in the test instances are small, we added all of the mixing inequalities to the LP relaxation formulation. Figure 2.8 shows this percentage for the test instances with and without the valid inequalities. As seen in this figure, valid inequalities increase the percentage of integral binary variables to more than 97% for almost every instances which motivated us to develop the solution approach, IFR. Based on our observations, high intralayer dependency probability and smaller arc capacities have significant effects on decreasing this percentage for the test instances that do not have the valid inequalities. For these test instances, the LP bound improvement due to including the valid inequalities is the difference between the LP bound with and without valid inequalities divided by the LP bound without valid inequalities. The improvement is reported to be between 0% and 11% in all the test instances.

Second, we solve our test instances with the IFR and the standard MIP solver, CPLEX, with 30 minutes as the time limit. The time limit for the LP and the MIP components in the IFR is set to be 500 and 200 seconds, respectively. Figure 2.9 shows the percentage gap obtained with these two approaches. For all these test instances, it takes IFR at most 2 iterations to find a feasible solution. Note that the valid inequalities are not present in the MIP formulation since CPLEX can have the same improvement on the upper bound of the formulation with its own built-in cuts in the root node of the branch and bound approach. We also examine the percentage gap of MIP if terminated after the amount of time required for IFR to find a feasible solution. A summary

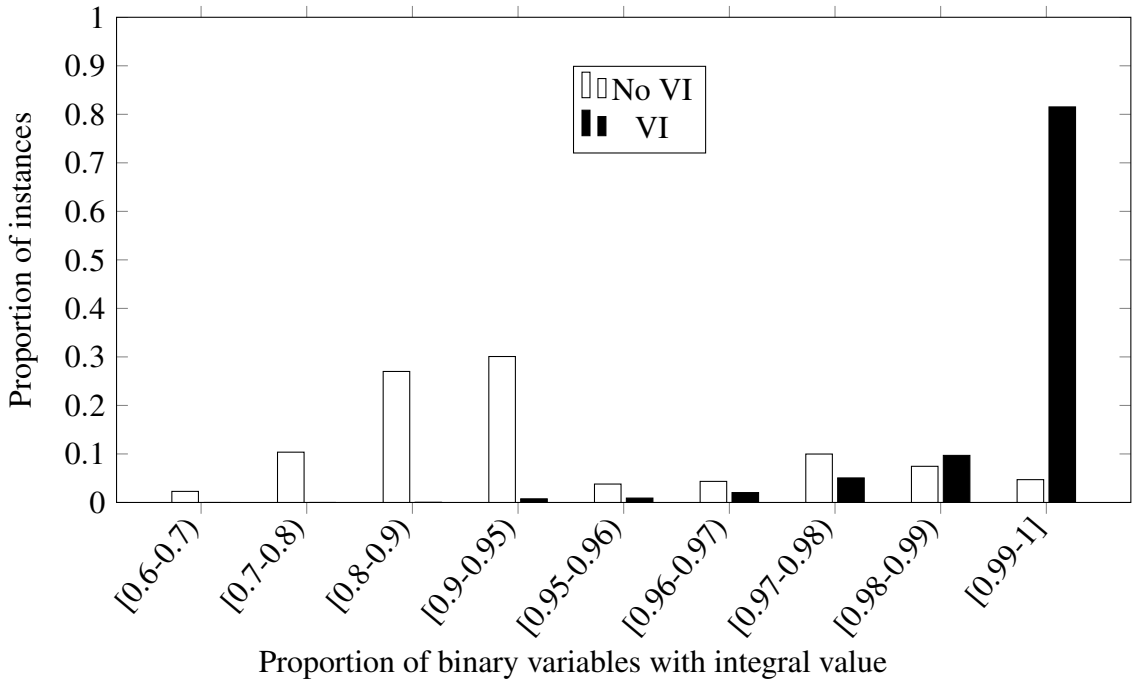


Figure 2.8: Proportion of binary variables with integral value in the LP relaxation solution of IMN

of this set of experiments is presented in Figure 2.9. From Figure 2.9, we can conclude that IFR outperforms CPLEX (with respect to optimality gap and solution time criteria) and the gains are even more clear when considering what CPLEX accomplishes if allowed to run for only as long as IFR. We also compare the cumulative proportion of instances for which IFR or MIP terminates within a specific time for the two approaches in Figure 2.10. As can be seen from Figure 2.10, IFR is faster than MIP in solving the instances to the percentage gap shown in Figure 2.9.

Third, we take a closer look at the instances for which MIP hit the 30 minutes time limit without solving to optimality. The percentage gap obtained by the two approaches is shown in Figures 2.11. The better performance of IFR in comparison to CPLEX in solving these instances is even more significant in these cases. Figure 2.11 reveals that fewer than 30% of IFR instances result in at least a 10% optimality gap; however the optimality gap exceeds 10% for more than 50% of MIP instances.

In summary, the valid inequalities bring the LP relaxation solution closer to a feasible MIP solution. IFR exploits this effect of valid inequalities to find the feasible MIP solution. With this

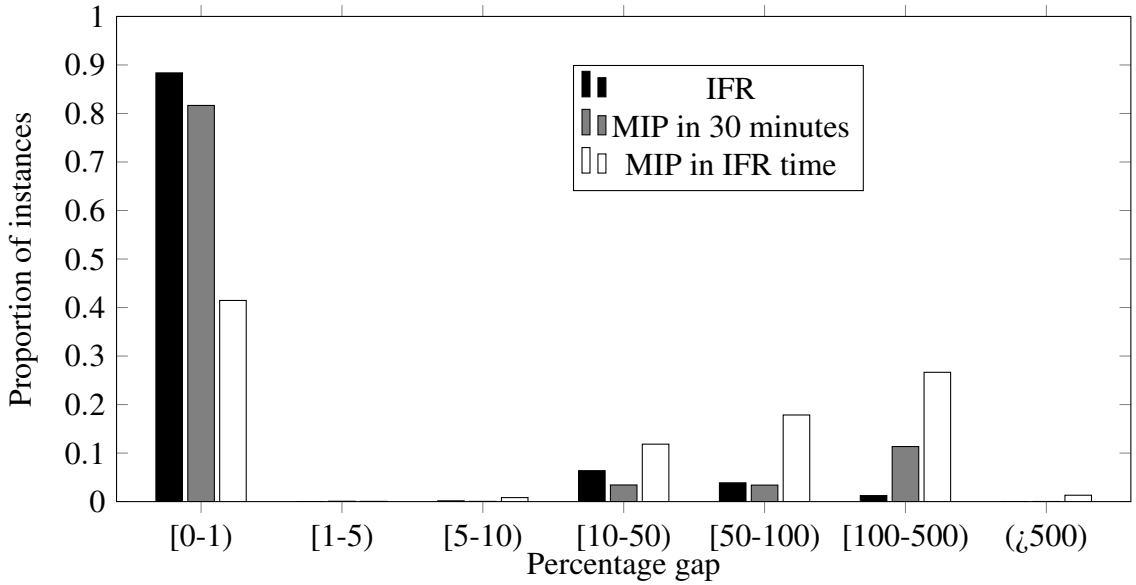


Figure 2.9: Percentage gap obtained with IFR and MIP

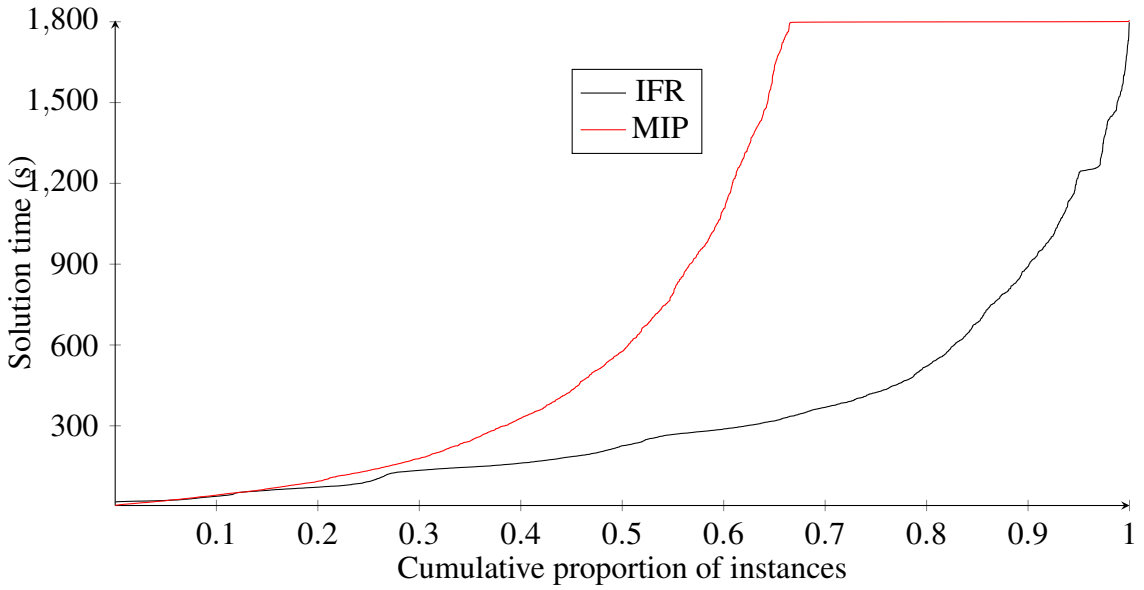


Figure 2.10: IFR vs. MIP with regards to the solution time

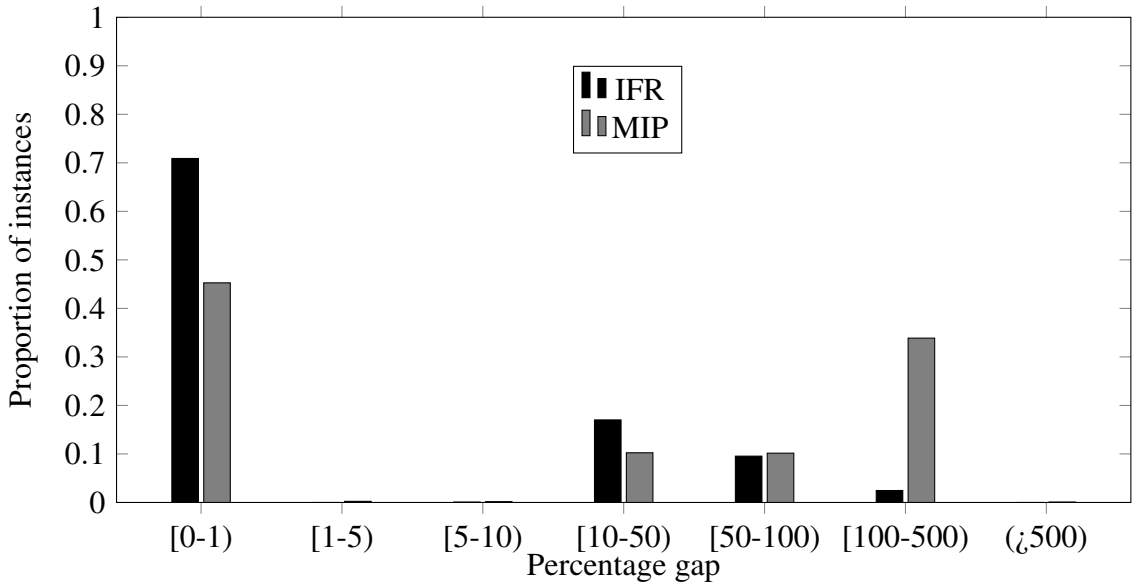


Figure 2.11: Percentage gap obtained with IFR and MIP for the instances in which MIP hit the time limit

computational study we have shown that IFR can find a better solution with less computational time in comparison with the MIP formulation for most of the instances.

2.9 Conclusion

We have developed a mathematical formulation that models the operation of interdependent infrastructure systems in order to optimize operations in a resource constrained environment. The objective of the model is to maximize the overall functionality of interdependent networks while keeping the cost as low as possible. Our model has several unique features over previous network-based models of interdependent infrastructure systems: (i) it is a compact formulation; (ii) it has the capability to capture many common types of interdependence; (iii) it allows for non-uniform dependency requirements on the consumption of flow at a given node in a given layer; and (iv) it considers intra-layer dependency. We showed that this problem is strongly NP-complete. We then identified two sets of valid inequalities which we utilized to develop a solution approach that exploits the effect of valid inequalities on the LP relaxation solution to expedite finding feasible solution to the problem. We tested IFR on a set of synthesized instances repre-

senting interdependent infrastructure systems. Our computational experiments point out the quick computational time needed for the approach to identify high-quality solutions for most of the instances. Further, we demonstrate that failing to model feature (iii) described above (i.e., non-uniform dependency requirements) may drastically reduce the quality of strategic restoration decisions derived to restore a disrupted collection of infrastructure systems. The work in this paper will be of value to future research in the area of restoration planning as it provides a compact operational level that can be solved in an efficient way. The model can also be extended to optimize mitigation strategies over a set of potential disasters.

Acknowledgment

The authors would like to acknowledge the computational support from the University of Arkansas High Performance Computing Center which is funded through multiple National Science Foundation grants and the Arkansas Economic Development Commission.

Bibliography

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

Baroud, H., Ramirez-Marquez, J. E., Barker, K., and Rocco, C. M. (2014). Stochastic measures of network resilience: Applications to waterway commodity flows. *Risk Analysis*, 34(7):1317–1335.

Beheshtian, A., P. Donaghy, K., Gao, H. O., Safaie, S., and Geddes, R. (2018). Impacts and implications of climatic extremes for resilience planning of transportation energy: A case study of New York city. *Journal of Cleaner Production*, 174:1299–1313.

Buldyrev, S. V., Parshani, R., Paul, G., Stanley, H. E., and Havlin, S. (2010). Catastrophic cascade of failures in interdependent networks. *Nature*, 464(7291):1025–1028.

Cavdaroglu, B., Hammel, E., Mitchell, J. E., Sharkey, T. C., and Wallace, W. A. (2013). Integrating restoration and scheduling decisions for disrupted interdependent infrastructure systems. *Annals of Operations Research*, 203(1):279–294.

Garey, M. R. and Johnson, D. S. (1979). Computers and intractability: a guide to np-completeness.

Gong, J., Lee, E. E., Mitchell, J. E., and Wallace, W. A. (2009). Logic-based multiobjective optimization for restoration planning. In *Optimization and Logistics Challenges in the Enterprise*, pages 305–324. Springer US.

González, A. D., Dueñas-Osorio, L., Sánchez-Silva, M., and Medaglia, A. L. (2016). The interdependent network design problem for optimal infrastructure system restoration. *Computer-Aided Civil and Infrastructure Engineering*, 31(5):334–350.

Günlük, O. and Pochet, Y. (2001). Mixing mixed-integer inequalities. *Mathematical Programming*, 90(3):429–457.

Haimes, Y. Y., Horowitz, B. M., Lambert, J. H., Santos, J. R., Lian, C., and Crowther, K. G. (2005). Inoperability input-output model for interdependent infrastructure sectors. I: Theory and methodology. *Journal of Infrastructure Systems*, 11(2):67–79.

Haimes, Y. Y. and Jiang, P. (2001). Leontief-based model of risk in complex interconnected infrastructures. *Journal of Infrastructure Systems*, 7(1):1–12.

Helber, S. and Sahling, F. (2010). A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics*, 123(2):247–256.

Lee, E. E., Mitchell, J. E., and Wallace, W. A. (2007). Restoration of services in interdependent infrastructure systems: A network flows approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(6):1303–1317.

Leontief, W. W. (1951). Input-output economics. *Scientific American*, 185(4):15–21.

Liscouski, B. and Elliot, W. (2004). Final report on the August 14, 2003 blackout in the United States and Canada: Causes and recommendations. Technical report, U.S.-Canada Power System Outage Task Force.

Matisziw, T. C., Murray, A. T., and Grubesic, T. H. (2010). Strategic network restoration. *Networks and Spatial Economics*, 10(3):345–361.

McLay, L. A. (2015). Discrete optimization models for homeland security and disaster management. In Aleman, D. M. and Thiele, A. C., editors, *The Operations Research Revolution: INFORMS Tutorials in Operations Research*, chapter 7, pages 111–132.

Morton, D. P., Pan, F., and Saeger, K. J. (2007). Models for nuclear smuggling interdiction. *IIE Transactions*, 39(1):3–14.

Nateghi, R. (2018). Multi-dimensional infrastructure resilience modeling: An application to hurricane-prone electric power distribution systems. *IEEE Access*, PP(99):1–1.

Nurre, S. G., Cavdaroglu, B., Mitchell, J. E., Sharkey, T. C., and Wallace, W. A. (2012). Restoring infrastructure systems: An integrated network design and scheduling (INDS) problem. *European Journal of Operational Research*, 223(3):794–806.

Nurre, S. G. and Sharkey, T. C. (2014). Integrated network design and scheduling problems with parallel identical machines: Complexity results and dispatching rules. *Networks*, 63(4):306–326.

Ouyang, M. (2014). Review on modeling and simulation of interdependent critical infrastructure systems. *Reliability Engineering & System Safety*, 121:43–60.

Pant, R., Barker, K., Ramirez-Marquez, J. E., and Rocco, C. M. (2014). Stochastic measures of resilience and their application to container terminals. *Computers & Industrial Engineering*, 70:183–194.

Paredes, R. and Dueñas-Osorio, L. (2015). A time-dependent seismic resilience analysis approach for networked lifelines. In: Proceedings of the 12th International Conference on Applications of Statistics and Probability in Civil Engineering.

Pederson, P., Dudenhoeffer, D., Hartley, S., and Permann, M. (2006). Critical infrastructure interdependency modeling: A survey of US and international research. Technical report, Idaho National Laboratory.

Rinaldi, S. M. (2004). Modeling and simulating critical infrastructures and their interdependencies. in: Proceedings of the 37th annual Hawaii International Conference on System Sciences.

Rinaldi, S. M., Peerenboom, J. P., and Kelly, T. K. (2001). Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE Control Systems*, 21(6):11–25.

Sharkey, T. C., Cavdaroglu, B., Nguyen, H., Holman, J., Mitchell, J. E., and Wallace, W. A. (2015). Interdependent network restoration: On the value of information-sharing. *European Journal of Operational Research*, 244(1):309–321.

Sharkey, T. C., Nurre, S. G., Nguyen, H., Chow, J. H., Mitchell, J. E., and Wallace, W. A. (2016). Identification and classification of restoration interdependencies in the wake of Hurricane Sandy. *Journal of Infrastructure Systems*, 22(1):04015007.

Shen, S. (2013). Optimizing designs and operations of a single network or multiple interdependent infrastructures under stochastic arc disruption. *Computers & Operations Research*, 40(11):2677–2688.

Sullivan, K. M., Morton, D. P., Pan, F., and Smith, J. C. (2014). Securing a border under asymmetric information. *Naval Research Logistics*, 61(2):91–100.

3. Single-train Yard Routing Problem

3.1 Introduction

One of the fundamental and difficult problems faced by rail freight transportation systems is how to move cars and shipments through the railyard. Upon receiving a request from shippers, the railroads must decide which route should be taken to move cars from one location in the railyard to another (Haghani, 1987). Railyard routing is of importance since such a problem must be solved repeatedly in a very short time for simulations of train movements in large complex yards.

There appear to be different streams of research related to routing trains. In the first stream, the focus is on finding the fastest or cost-efficient routes in a network-level routing problem. Hernando et al. (2018) propose a method to find the fastest route from one station to another under consideration of both the train's acceleration and deceleration time and the time required to reverse direction. Fu and Dessouky (2018) study a single-train station-to-station routing problem in a rail network with only double track segments, where the objective is to route one train as fast as possible subject to acceleration and deceleration restrictions. This problem is an NP-hard special case of the problem examined by Nagarajan and Ranade (2008), which incorporates conflicts due to the existence of pre-scheduled routes. Researchers have also considered the problem of minimizing transportation cost over multiple origin-destination pairs with fuzzy input parameters (Yang et al., 2011) and multiple transportation modes (Sun et al., 2016) and combined it with a train formation plan (Lin, 2017). Haghani (1989) presents a model to combine the tactical train routing problem and the operational empty car distribution model. This combined problem is to determine the optimal routing and empty car distribution policies based on a forecast of demands to be shipped between the yards of its network.

Other researchers focus on a yard-level routing problem that optimizes a specific objective. Examples of optimization criteria are minimizing the number of direction changes and manual crossovers (Riezebos and Van Wezel, 2009) and finding the most economical car paths (Fügenschuh et al., 2013).

The second stream of work related to the train routing problem focuses on developing a set of possible routes for each origin-destination pair in a network of rail stations. Railway operations (i.e., the processes that take place on and around the rail infrastructure) are usually designed in advance in terms of routing assigned to each train and their timings (timetable). However, the timetable may become infeasible when disturbances arise at the operational level. The real-time railway traffic management problem (rtRTMP) aims to respond to such disturbances by generating a new efficient feasible plan of operations by determining a conflict-free set of train routes. The most widely used objective function in this problem is to minimize the propagation of delays. Rather than solve this NP-hard problem directly, researchers have approached rtRTMP by first solving a separate optimization problem in order to identify a subset of alternative routes for each train (Sama et al., 2016, 2017).

The existing literature on the yard routing problem has focused on choosing a route from a predefined set of routes for each train. In order to automate the process of generating the routes (e.g., for incorporation in one of the existing routing models), it is necessary to account for the geometry of the track segments in the railyard as well as the fact that the train occupies space in the railyard. There is some precedent, the context of convoy routing (Akgün and Tansel, 2007; Chardaire et al., 2005; Goldstein et al., 2010; Gopalan, 2015), for non-rail-related routing of commodities that take up space on the network, but the combination of this feature with complex track geometry has not been addressed. In this paper, we seek to optimize the route selection from among all the possible routes in the yard network while explicitly considering the geometry of the yard network and the space taken up by the train.

Given a railyard and a single train specified by its origin, destination, and length, our work is to determine a feasible route for the train with a minimum total length subject to the geometry of the yard tracks. The length of a train equals to the length of its locomotive and the cars that it moves in the railyard. The difference between this problem and the traditional shortest path problem is that the route must accommodate, subject to the geometry of the yard tracks, the length of the cut of cars plus the length of the locomotive at any time. In this chapter, we will propose an

approach to model this problem and find an optimal route for the train in the railyard. Cordeau et al. (1998) models the railyard as a network. Hence, we use a network-based approach to model the single train routing problem where edges and nodes correspond to track segments and connections between those track segments in the yard, respectively. The proposed approach models the problem in two stages: a *preprocessing* stage that takes as input the specific length of the train and tracks' configuration to determine the feasibility of certain movements for the train in the railyard and a *routing* stage that finds a shortest origin-destination route given the output of the preprocessing stage.

The main contributions of this chapter are as follows: (i) our proposed modeling approach for a train routing problem is the first to consider the geometry of the track segments in a railyard; (ii) we prove that the preprocessing problem is NP-complete and develop an algorithm and an integer programming formulation to model it; (iii) we develop a linear programming formulation that solves an important special case of the preprocessing stage in polynomial time; (iv) our approach to model the routing stage can be solved with classical shortest path algorithms; (v) we develop a solution algorithm that allows the preprocessing stage to be solved as needed with regard to the solution of the routing stage; and (vi) our modeling approach is validated on a real railyard dataset provided by a major rail company.

3.2 Problem Definition

We consider the problem of routing a train of length $L \geq 0$ through a railyard defined by the undirected network $G = (N, E)$ with nodes N and edges E. Edges in this network correspond to track segments and nodes correspond to connections between those track segments. The train consists of a locomotive and a cut of cars attached to the locomotive. The locomotive can either pull or push the cars in the railyard. Let $c_e \geq 0,\ e \in E$, denote the length of edge e. For each node $i \in N$, let $N(i) \subseteq N \setminus \{i\}$ denote the set of nodes adjacent to i. Number the nodes adjacent to $i \in N$ as $N(i) = \{N_k(i)\}_{k=1}^{|N(i)|}$ and the edges adjacent to $i \in N$ as $\{e(i,k)\}_{k=1}^{|N(i)|}$ where $e(i,k) \equiv [i, N_k(i)]$. Let node i_O and i_D denote the origin and destination of the train route, respectively. The railyard

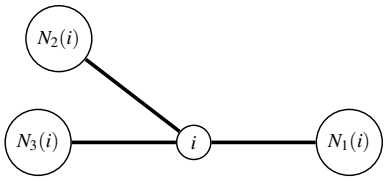


Figure 3.1: Switch node i

network G has special structure as described below:

Property 1 $|N(i)| \leq 3,\ \forall i \in N$. *That is, each node must have a degree at most three.*

In order to describe further properties of the yard network, we now formalize some additional definitions.

Definition 1 *If* $|N(i)| = 3$*, then* $i \in N$ *is said to be a* switch *node. Define* N^S *as the set of switch nodes.*

Switch nodes impose some restrictions on the train's movement through the yard, as we now summarize. For a switch node i, the collection of nodes $\{i, N_1(i), N_2(i), N_3(i)\}$ are always oriented (see Figure 3.1) such that, among the three edges $\{e(i,k)\}_{k=1}^{3}$, exactly one pair of edges forms an acute angle. Without loss of generality, suppose the edges $e(i,2)$ and $e(i,3)$ associated with each switch node $i \in N^S$ form an acute angle. We now provide definitions and properties that enable mathematically representing this restriction.

Definition 2 *For* $e, e' \in E,\ e \neq e'$*, we say that edge pair* $\{e, e'\}$ *is* directly traversable *if (1)* e *and* e' *share exactly one node as a common endpoint and (2) the angle formed by* e *and* e' *is non-acute. For example, the edge pairs* $\{[4,5],[5,7]\}$ *and* $\{[4,5],[5,6]\}$ *are directly traversable in Figure 3.2, but* $\{[5,6],[6,7]\}$ *is not.*

By appropriately numbering the neighbors of each switch node, the following properties will be satisfied:

Property 2 *For* $i \in N$ *with* $|N(i)| \geq 2$*, the edge pair* $\{e(i,1), e(i,2)\}$ *is directly traversable.*

Property 3 *For* $i \in N^S$*, the edge pair* $\{e(i,1), e(i,3)\}$ *is directly traversable.*

Property 4 *For $i \in N^S$, the edge pair $\{e(i,2), e(i,3)\}$ is* not *directly traversable.*

Assumption 1 *Because degree-two nodes do not have acute angles (see Property 2), one can transform the yard network to an equivalent yard network with only leaf nodes and switch nodes. This transformation entails contracting the two edges (with lengths l_1 and l_2) adjacent to a degree-two node into a single edge with length $l_1 + l_2$. When convenient, we may therefore assume without loss of generality that all nodes in $N \setminus N^S$ are leaf nodes.*

Definition 3 *A sequence of edges $e(1)$–$e(2)$–$\cdots$–$e(k)$ is said to be a* walk *provided that each successive pair of edges has exactly one endpoint in common. Alternatively, we may refer to the walk $e(1)$–$e(2)$–$\cdots$–$e(k)$ by listing its nodes in sequence, e.g., as $i(0)$–$i(1)$–$\cdots$–$i(k)$. For example, the node sequence* 3–4–12–11–9–3–2 *is a walk in the yard network illustrated in Figure 3.2.*

When a network satisfies Properties 1–4, we say that the network has *yard structure* or is a *yard network*. We now present some additional definitions that will enable us to represent the train's movement through the yard network.

Definition 4 *A walk $e(1)$–$e(2)$–$\cdots$–$e(k)$ is said to be a* directly traversable walk *provided that each successive pair of edges is directly traversable. For instance* 2–3–4–5–7–8–2–3 *is a directly traversable walk in Figure 3.2, but* 2–3–9–11–12–4–3 *is not because it traverses the acute angles $\{[2,3],[3,9]\}$ and $\{[12,4],[4,3]\}$.*

Definition 5 *A directly traversable walk $i(0)$–$i(1)$–$\cdots$–$i(k)$ is said to be a* simple directly traversable walk *provided that (a) there exists a node $j \in \{i(0), i(k)\}$ that the walk visits no more than twice and (b) the remaining $|N| - 1$ nodes are visited by the walk no more than once. (Thus, only one node may be visited twice by a simple directly traversable walk, and that node must be the first or last node on the walk.) For example,* 2–3–4–5–7–8–2 *is a simple directly traversable walk in Figure 3.2, but* 2–3–4–5–7–8–2–5 *is not because more than one node is repeated.*

Definition 6 *A simple directly traversable walk W, with nodes $i(0)$–$i(1)$–$\cdots$–$i(k)$, is said to be* anchored at *node j if $j = i(0)$ and $N_1(j) = i(1)$. In this case, we say W is an "SDT(j) walk."*

For example, 3–4–5–7 *and* 3–4–5–6 *are SDT(3) walks in Figure 3.2, but* 3–2–8 *is not because* $N_1(3) \neq 2$ *(i.e., the edge* $[2,3]$ *is one of the edges on node* 3*'s acute angle).*

Definition 7 *A directly traversable walk W, with nodes* $i(0)$–$i(1)$–$\cdots$–$i(k)$*, is said to be a "reverse SDT(*j*) walk" or "r-SDT(*j*) walk" if* $i(0) = j$ *and* $i(1) \in \{N_2(j), N_3(j)\}$*. For example, the node sequence* 4–12–11–9 *is an r-SDT(4) walk in the yard network illustrated in Figure 3.2.*

Definition 8 *An SDT(*j*) (respectively, r-SDT(*j*)) walk* $(j =)$ $i(0)$–$i(1)$–$\cdots$–$i(k)$ *is said to be* maximal *if there does not exist a node* $j' \in N$ *such that* $i(0)$–$i(1)$–$\cdots$–$i(k)$–j' *is an SDT(*j*) (respectively, r-SDT(*j*)) walk. For example,* 3–4–5–6 *and* 4–12–11–9–3–4 *are respectively maximal SDT(3) and maximal r-SDT(4) walks in Figure 3.2; however,* 3–4–5–7 *is not a maximal SDT(3) walk because it can be extended as* 3–4–5–7–8 *into another SDT(3) walk.*

Definition 9 *Let* MS_j *and* $\overline{MS}_j$ *respectively be the length of the longest SDT(*j*) walk and the length of the longest r-SDT(*j*) walk. For example,* $MS_3 = 18$ *corresponding to the SDT(*3*) walk* 3–4–5–7–8–2–3 *in Figure 3.2, and* $\overline{MS}_3 = 24$ *corresponding to the r-SDT(*3*) walk* 3–9–11–12–4–5–7–8–2–3–4.

Definition 10 *A simple directly traversable walk* $i(0)$–$i(1)$–$\cdots$–$i(k)$ *is said to be a* directly traversable cycle *if* $i(0) = i(k)$*. For instance,* 2–3–4–5–7–8–2 *is both a maximal SDT(2) walk and a directly traversable cycle in Figure 3.2.*

Given the above definitions, we are now in a position to formally characterize the set of feasible train positions inside the railyard.

Definition 11 *A simple directly traversable walk* $e(1)$–$e(2)$–$\cdots$–$e(k)$ *(with node representation* $i(0)$–$i(1)$–$\cdots$–$i(k)$*) is said to be a* feasible position *for the train of length L if* $c_{e(1)} + c_{e(2)} + \cdots + c_{e(k)} \geq L$*. If* $j = i(0) \in N^S$ *and* $N_1(j) = i(1)$*, then we say the feasible position is anchored at* j *or that the feasible position is an "SDT(*j*) feasible position." For example, the node sequence* 3–4–5–6 *is an SDT(*3*) feasible position for a train of length* 3 *in Figure 3.2.*

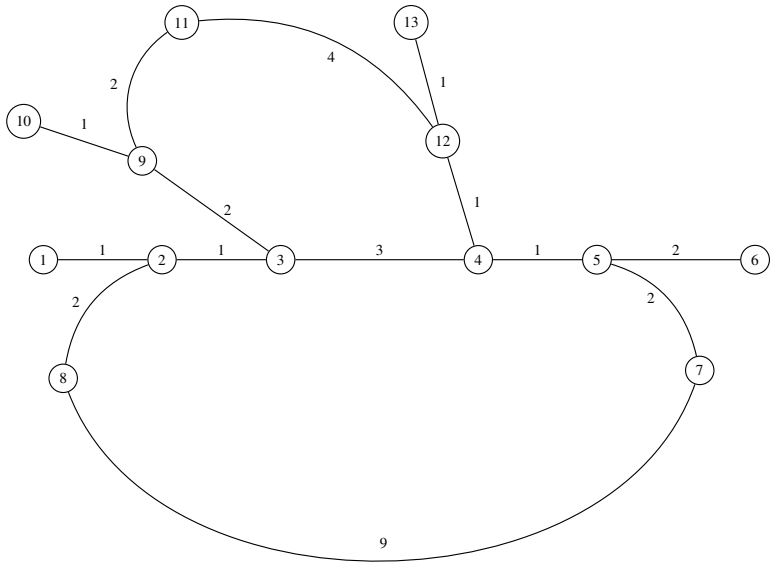


Figure 3.2: A yard network in which each edge $e \in E$ is labeled with length c_e.

There are restrictions on the train's movement through the yard, as we now summarize. To traverse the edge pair $\{e(j,2), e(j,3)\},\ j \in N^S$, in sequence, the train's entire length must first pass through node j before it reverses its direction to continue with the intended walk. Therefore, a necessary condition to traverse the edge pair $\{e(j,2), e(j,3)\},\ i \in N^S$, in sequence is the existence of an SDT(j) feasible position. In this case, we say the switch node is *indirectly traversable*. This definition is formalized below.

Definition 12 *Define N^L as the subset of nodes $j \in N^S$ such that the edge pair $\{e(j,2), e(j,3)\}$ is* indirectly traversable *(that is, there exists a feasible SDT(j) position) for a train of length L.*

The routing problem is intended to find a shortest route for the train from one feasible position to another feasible position in the yard network. By contrast, we are assuming the origin and destination are nodes in the network. One could transform a problem with origin/destination feasible positions into a set of problems with origin/destination nodes by extracting as many as four pairs of origin/destination nodes, with one being an endpoint of the origin feasible position and the other an endpoint of the destination feasible position.

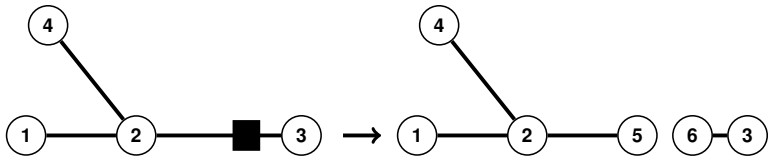


Figure 3.3: Modifying network to account for idle cars

The destination node of the train may also constrain its movement. If the destination node is a leaf, the locomotive must push the cars in backward onto the track, so as not to trap itself on the track behind the cars. This means that the train must reverse its direction an odd and an even number of times before it reaches its leaf node destination assuming that it pulls and pushes the cars from its origin, respectively. The methodology we describe in this chapter will provide a means for restricting the direction of arrival at the destination.

The last restriction on the train's movement stems from the state of the yard which is constantly changing. Idle cars may be occupying track space through the yards which have effects on train movement. In the network shown in Figure 3.3 that there is an idle car on arc $(2,3)$, we modify the network $G = (N,E)$ to $G' = (N \cup \{5,6\}, E \cup [2,5] \cup [6,3] \setminus [2,3])$ to be used as a railyard network in a routing problem.

Given all of the restrictions expressed above, the resulting optimization problem is to identify a shortest route for the train through a railyard from one location to another. We term this optimization problem as the Yard Routing Problem (YRP).

3.3 Problem Formulation

The proposed formulation in this section models the YRP in two stages: (1) the preprocessing stage which checks whether or not the walk $[i,N_2(i)]$–$[i,N_3(i)]$ at every switch node $i \in N^S$ is indirectly traversable for the train of length L; and (2) the routing stage which finds the shortest route possible for the train given the information obtained from the first stage. Note that the yard network used to solve the YRP is designed after considering the idle cars in the yard. As explained in the previous section, $MS_i \geq L$ will be a necessary condition to imply that node i is

indirectly traversable. We need to ensure that a shortest route obtained in the routing stage is also feasible for the train and does not cause the train to have a collision with itself while traveling toward its destination node. This concern will not exist if the yard network is acyclic or has a lower bound on the length of directly traversable cycles, therefore, discovering that a switch node $i \in N^S$ is indirectly traversable a necessary and sufficient condition to imply that the train can reverse directions at node i (e.g., arriving on edge $e(i,2)$, passing through node i in the direction of node $N_1(i)$, reversing direction, and then continuing on edge $e(i,3)$). It should be noted, however, that this condition is only necessary (and not sufficient) in the case where there is a directly traversable cycle of length less than L units. For example, in the yard network illustrated in Figure 3.2, node 3 is indirectly traversable if L equals 18 units. In this yard, the train cannot traverse the walk 2–3–9–11–12–4 without colliding with itself but it is a valid walk if the YRP is divided to two stages. Although there is some loss of generality in assuming all directly traversable cycles are longer than the train itself, we focus on this case for three reasons: (i) In practice, the routing problem is commonly solved to support the operation of combining cars within cut of cars—in this process, there will be a number of cars attached, possibly from different locations within the yard, before the cut of cars becomes too lengthy for our model to handle; (ii) although we cannot guarantee it, using solutions to the case of "well-conditioned" cycles may provide solutions that are useful if there exist ill-conditioned cycles in the network; and (iii), we have addressed the case of well-conditioned cycles in significant detail, providing ideas that may prove to be useful in analyzing the case of ill-conditioned cycles at a later time. We now proceed to explain each proposed stage to solve the YRP.

3.3.1 Preprocessing Stage

In this stage, we check each switch node $j \in N^S$ to determine whether or not edge pair $\{e(j,2), e(j,3)\}$ is indirectly traversable. The preprocessing problem for a given switch node is defined below.

PREPROCESS(j)

Instance: A yard network $G = (N, E)$ with numbered adjacency lists $\{N_k(i)\}_{k=1}^{|N(i)|}$ for all $i \in N$ and edge costs $c_e \geq 0,\ e \in E$; a train length $L \geq 0$; a switch node $j \in N^S$.

Question: Does there exist an SDT(j) feasible position? (Equivalently: Does there exist an SDT(j) walk of length at least L?)

We prove that PREPROCESS($\cdot$) is NP-complete in Section 3.3.1.1 but show in Sections 3.3.1.2–3.3.1.3 that it is polynomially solvable in the special case where a minimum length is imposed on all directly traversable cycles.

3.3.1.1 Complexity

We establish the NP-completeness of PREPROCESS($\cdot$) based on a reduction from the longest path problem, which is known to be NP-complete (Garey and Johnson, 1979).

LONGEST-PATH

Instance: An undirected network $\tilde{G} = (\tilde{N}, \tilde{E})$, a positive length $\tilde{L}$ and two nodes $\tilde{s}, \tilde{t} \in \tilde{N}$.

Question: Does there exist a simple path (i.e., in which each node is visited at most once) of at least $\tilde{L}$ edges between the nodes $\tilde{s}$ and $\tilde{t}$ in network $\tilde{G}$?

We now establish the complexity of PREPROCESS($\cdot$).

Theorem 3 *PREPROCESS($\cdot$) is NP-complete.*

Proof. Inclusion in NP is clear due to the fact that the length of a walk can be computed in polynomial time. We prove NP-hardness of PREPROCESS($\cdot$) via reduction from LONGEST-PATH. Towards proving NP-hardness, we first note that LONGEST-PATH remains NP-complete under the assumption that every node in $\tilde{N} \setminus \{\tilde{s}\}$ remains connected to $\tilde{t}$ when $\tilde{s}$ and its adjacent edges are removed. This assumption does not change the complexity of longest path because a simple path that begins at node $\tilde{s}$ may not re-enter node $\tilde{s}$.

Given an arbitrary instance $\tilde{I} = \{\tilde{N}, \tilde{E}, \tilde{L}, \tilde{s}, \tilde{t}\}$ of LONGEST-PATH, we construct an instance $I = \{N, E, L, c, j\}$ of PREPROCESS(j) as described below. We first expand $\tilde{G} = (\tilde{N}, \tilde{E})$ by attaching a chain of $|\tilde{N}| + 1$ nodes $\{t^0, t^1, t^2, \cdots, t^{|\tilde{N}|}\}$ to node $\tilde{t}$. Let $\bar{N}$ denote the modified node set.

Thus far, yard structure (i.e., Properties 1–4 as defined in Section 3.2) has not been imparted on the network. (Specifically, we may currently have that $d(i) > 3$ for some nodes $i \in \bar{N}$, where $d(i)$ is the degree of node i in the expanded network; furthermore, nodes will need to be designated as switch or non-switch nodes with an appropriate numbering provided for each switch nodes' neighbors.) We now explain an additional expansion that will provide the required yard structure.

Initially, we construct a new node s and add the edge $[s, \tilde{s}]$, increasing $d(\tilde{s})$ by one. We also construct nodes $\{s_2, s_3\}$ as leaf-node neighbors of s. Now that $d(s) = 3$, we may define s as a switch node (as is required in order for PREPROCESS(s) to be well-defined) with $N_1(s) = \tilde{s}$, $N_2(s) = s_2$, and $N_3(s) = s_3$ and set $j = s$. (In what follows, it is possible that $\tilde{s}$ will be replaced as the first neighbor of s.) The transformation up to this point is shown in Figure 3.4.

Now, for each node $i \in \bar{N}$ (which does not include $\{s, s_2, s_3\}$), we perform the following: (a) if $d(i) = 1$, no action is required because Properties 1–4 impose no restriction on leaf nodes; (b) if $d(i) = 2$, no action is required—simply set the pair of edges adjacent to i to be directly traversable so that Properties 1–4 are satisfied for this node; (c) if $d(i) \geq 3$, perform the expansion that is depicted in Figure 3.5 and described in the following paragraph.

For $i \in \bar{N}$ with $d(i) \geq 3$, we let $\{u(1), u(2), \ldots, u(d(i))\}$ denote the neighbors of i and replace node i with $3d(i)$ new switch nodes defined by the sets $\{v(i,\ell)\}_{\ell=1}^{d(i)}$, $\{q(i,\ell)\}_{\ell=1}^{d(i)}$, and $\{r(i,\ell)\}_{\ell=1}^{d(i)}$. Edges are added between these nodes as follows:

- For $\ell = 1, \ldots, d(i)$, define $N_1(v(i,\ell)) = u(\ell)$, $N_2(v(i,\ell)) = q(i,\ell)$, and $N_3(v(i,\ell)) = r(i,\ell)$.
- For $\ell = 2, \ldots, d(i) - 1$, define $N_1(q(i,\ell)) = q(i,\ell+1)$, $N_2(q(i,\ell)) = v(i,\ell)$, and $N_3(q(i,\ell)) = q(i,\ell-1)$. Perform the same definitions for $\ell = 1$ and $\ell = d(i)$ with the following exceptions:

- For $\ell = d(i)$, define $N_1(q(i,\ell)) = r(i,1)$.
- For $\ell = 1$, $N_3(q(i,\ell)$ is undefined. Thus, $q(i,1)$ will have degree two in the transformed network.

- For $\ell = 2,\ldots,d(i)-1$, define $N_1(r(i,\ell)) = r(i,\ell-1)$, $N_2(r(i,\ell)) = v(i,\ell)$, and $N_3(r(i,\ell)) = r(i,\ell+1)$. Perform the same definitions for $\ell = 1$ and $\ell = d(i)$ with the following exceptions:
  - For $\ell = 1$, define $N_1(r(i,\ell)) = q(i,d(i))$.
  - For $\ell = d(i)$, $N_3(r(i,\ell))$ is undefined. Thus, $r(i,d(i))$ will have degree two in the transformed network.

Because node i—which was previously a neighbor of nodes $\{u(1),u(2),\ldots,u(d(i))\}$–has been removed, node $v(i,\ell)$ will replace the position of node i in the numbered adjacency list for node $u(\ell),\ \ell = 1,\ldots,d(i)$. We define $V(i) = \{v(i,\ell)\}_{\ell=1}^{d(i)} \cup \{q(i,\ell)\}_{\ell=1}^{d(i)} \cup \{r(i,\ell)\}_{\ell=1}^{d(i)}$ as the set of new nodes that resulted from expanding node $i \in \bar{N}$ (where $V(i) = \{i\}$ if $d(i) \leq 2$) and $e^\star(i)$ as the new edge between $q(i,d(i))$ and $r(i,1)$. For $d(i) \geq 3$, observe that a simple directly traversable walk between any pair of nodes in $\{v(i,\ell)\}_{\ell=1}^{d(i)}$ on the subgraph induced by $V(i)$ must traverse the edge $e^\star(i)$.

Let G denote the resulting network, and observe that the nodes of the resulting network G have maximum degree three with all degree-three nodes having designated first, second, and third neighbors—therefore, G constitutes a yard network. Let N and E respectively denote the nodes and edges of this network, and define N^S as the set of degree-three nodes. The node sets $V(i),\ i \in \bar{N}$ are disjoint, and only s, s_2, and s_3 are contained within N but not $\bigcup_{i\in\bar{N}} V(i)$.

We assign the length of edge $e \in E$ as $c_e = 0$ if (a) there exists a node $i \in \bar{N}$ such that both of the endpoints of E are elements of $V(i)$ or (b) one of the endpoints of e is the node s; otherwise, $c_e = 1$. In particular, note that for each $t^\ell,\ \ell = 1,\ldots,|\tilde{N}|$, the edge $[t^{\ell-1},t^\ell]$ is contained in E with unit length.

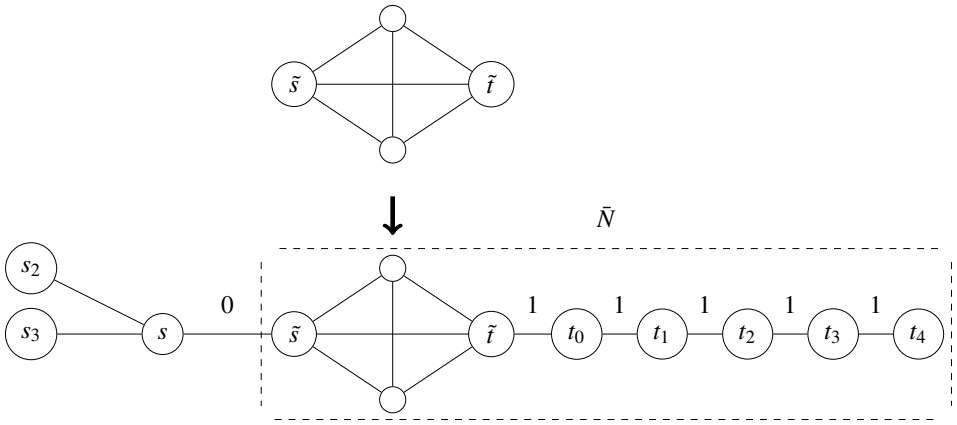


Figure 3.4: First step in constructing the instance $I = \{N, E, L, c, j\}$ of PREPROCESS(j)

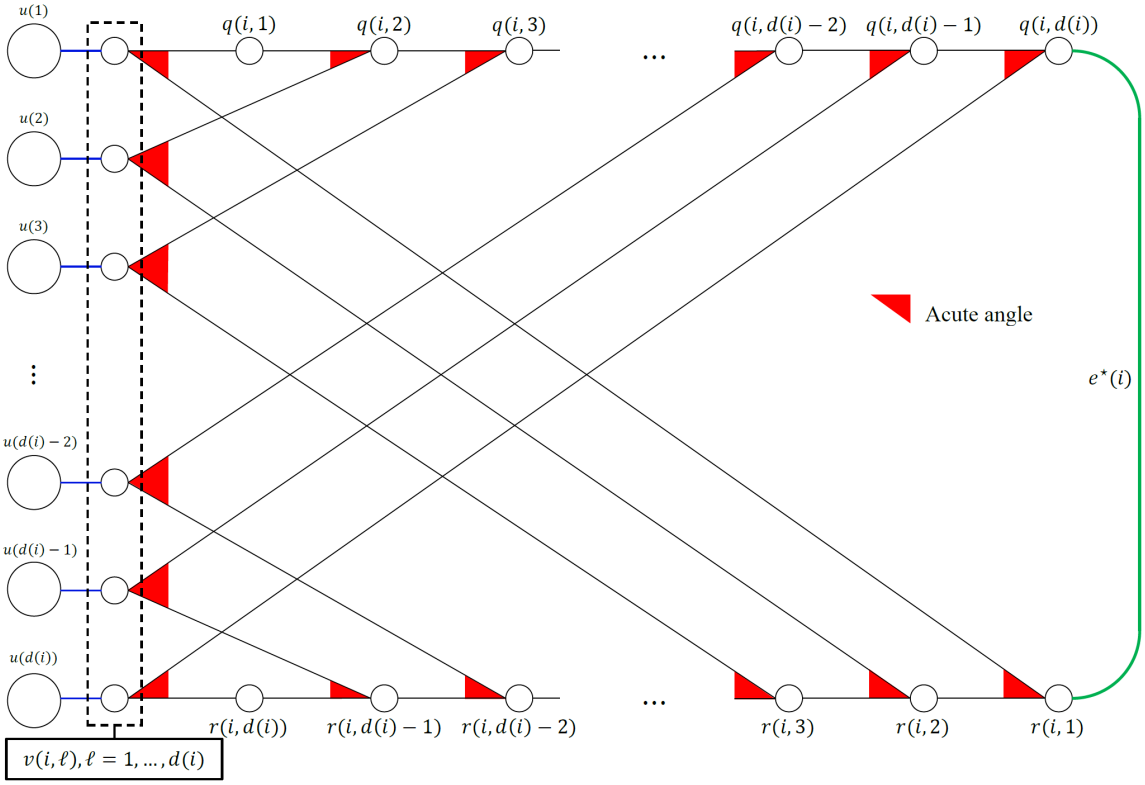


Figure 3.5: Expansion performed on node $i \in \bar{N}$ with $d(i) \geq 3$

We now prove that the answer to PREPROCESS(s) for a train of length $L = \tilde{L} + |\tilde{N}| + 1$ is "yes" if and only if the answer to LONGEST-PATH is "yes." Suppose there exists an SDT(s) feasible position. Note that this position intersects a sequence of node sets $V(i(0))$–$V(i(1))$–$\cdots$–$V(i(m))$, where $i(h) \in \bar{N},\ \forall h \in \{0,1,\ldots,m\}$ and $[i(h-1),i(h)] \in \tilde{E},\ \forall k \in \{1,\ldots,h\}$, and $i(0) = \tilde{s}$ based on the construction of G. The length of this walk from the switch node s until it intersects $V(i(0)) = V(\tilde{s})$ is zero. For a given $h \in \{0,1,\ldots,m\}$, the walk may pass through several edges in $V(i(h))$; however, the combined length of all such edges is zero. The walk also traverses m unit-length edges in moving from $V(i(h-1))$ to $V(i(h))$ for all $h = 1,\ldots,m$, and the total length of the walk is therefore

$$m \geq L\ (= \tilde{L} + |\tilde{N}| + 1). \tag{3.1}$$

Because each simple directly traversable walk that enters and exits $V(i),\ i \in \bar{N}$, must traverse the edge $e^{\star}(i)$, it is therefore impossible to enter and exit $V(i)$ a second time. Because each $V(i),\ i \in \bar{N}$, may be exited at most once, the nodes $\{i(0),i(1),\ldots,i(m)\}$ are unique. Combining this observation with Equation (3.1), the maximum length of any SDT(s) walk would be upper-bounded by $|\tilde{N}|$ if the chain t^0–t^1–$\cdots$–$t^{|\tilde{N}|}$ were excluded from E. Because $m > |\tilde{N}|$, the set $\{i(0),i(1),\ldots,i(m)\}$ must include nodes in the chain. Furthermore, a simple directly traversable walk that enters the chain cannot leave the chain, and therefore $i(m) \in \{t^0,t^1,\ldots,t^{|\tilde{N}|}\}$. Without loss of generality, we assume that $i(m) = t^{|\tilde{N}|}$, extending the feasible position if necessary so that it includes all nodes and edges in the chain. Summarizing the above, we have identified a feasible position that satisfies $i(h) = t^{|\tilde{N}|-m+h},\ h \in \{m-|\tilde{N}|,m-|\tilde{N}|+1,\ldots,m\}$. Because $[i(h-1),i(h)] \in \tilde{E},\ \forall h \in \{1,\ldots,m\}$, we have that $(\tilde{s} =)\ i(0)$–$i(1)$–$\cdots$–$i(m-|\tilde{N}|-1)$ is an $\tilde{s}$-$\tilde{t}$ path in $\tilde{G}$ of length $m-|\tilde{N}|-1$, which is at least $\tilde{L}$ by Equation (3.1). The answer to LONGEST-PATH is therefore "yes."

Conversely, suppose there exists an $\tilde{s}$-$\tilde{t}$ path $(\tilde{s} =)\ i(0)$–$i(1)$–$\cdots$–$i(h)\ (= \tilde{t})$ of $h \geq \tilde{L}$ edges in $\tilde{G}$. Based on construction of $\tilde{G}$, there is an SDT(s) walk in G with length h that traverses the node sets $V(i(0))$–$V(i(1))$–$\cdots$–$V(i(h))$. Noting that $i(h) = \tilde{t}$ and the SDT(s) walk has not yet traversed any nodes in the chain, we may extend this walk to obtain the SDT(s) walk $V(i(0))$–

$V(i(1))$–$\cdots$–$V(i(h))$–$V(t^0)$–$V(t^1)$–$\cdots$–$V(t^{|\tilde{N}|})$, which has length $h+\tilde{N}+1 \geq \tilde{L}+\tilde{N}+1 = L$. Therefore, the answer to PREPROCESS(s) is "yes" and the proof of this theorem is complete. □

This proof establishes NP-completeness for the general case of PREPROCESS($\cdot$). However, its complexity changes when network G has special structures as we summarize in Sections 3.3.1.2–3.3.1.3. Towards analyzing PREPROCESS($\cdot$), it will be useful to define two subroutines that contract the network in a way that preserves both yard structure and Assumption 1.

CONTRACT-1(i)

Given: A leaf node i that is the *first neighbor* of switch node $i' \in N^S$, i.e., $i = N_1(i')$ and $e(i',1) = [i,i']$.

Procedure: Remove the leaf node i, the switch node i', and all edges adjacent to i'. For $k \in \{2,3\}$, add a new leaf node i_k and a new edge $[N_k(i'), i_k]$ with length $c_{e(i',1)} + c_{e(i',k)}$. Let i_k replace i' in the numbered adjacency list of node $N_k(i')$.

CONTRACT-2(i)

Given: A leaf node i that is the *second or third neighbor* of switch node $i' \in N^S$, i.e., $i = N_k(i')$ and $e(i',k) = [i,i']$ for some $k \in \{2,3\}$. Let k' denote the (unique) element of $\{2,3\} \setminus \{k\}$.

Procedure: Remove the leaf node i, the switch node i', and all edges adjacent to i'. Add a new edge $[N_1(i'), N_{k'}(i')]$ with length $c_{e(i',1)} + c_{e(i',k')}$. Let $N_1(i')$ and $N_{k'}(i')$ respectively replace i' in the numbered adjacency list of nodes $N_{k'}(i')$ and $N_1(i')$.

Figure 3.6 illustrates the yard network that results upon executing CONTRACT-2(1) and CONTRACT-1(4) in sequence. We now prove some fundamental properties of these subroutines.

Lemma 1 *Let i be any leaf node such that $N_1(i') = i$ for some switch node $i' \in N^S$. Let $j \in N^S \setminus \{i'\}$, and let G' denote the network that results upon executing CONTRACT-1(i). Then, (a) the maximum length of any SDT(j) walk is the same in G and G', and (b) the maximum length of any r-SDT(j) walk is the same in G and G'.*

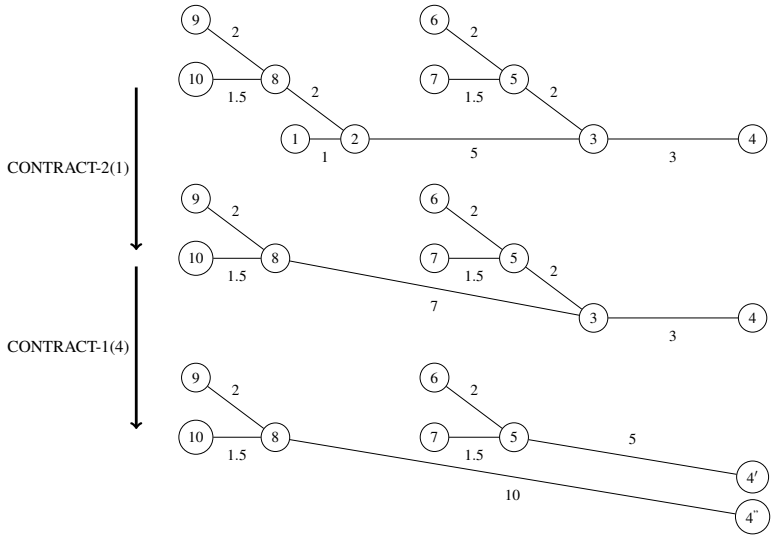


Figure 3.6: Process of executing CONTRACT-2(1) and CONTRACT-1(4) in sequence

Proof. We prove part (a) by establishing that there exists a maximal SDT(j) walk of length l in G if ($\leftarrow$) and only if ($\rightarrow$) there exists a maximal SDT(j) walk of length l in G'. Because $c_e \geq 0$, $\forall e \in E$, there is always a maximal SDT(j) walk among the set of maximum-length SDT(j) walks, and this therefore establishes the result.

Proof of ($\rightarrow$): Let W, with node representation $(j =)\ i(0)$–$i(1)$–$\cdots$–$i(m)$, be a maximal SDT(j) walk of length l in G. If W includes node i', then (because W is maximal) W must also include i, and we therefore have the following two cases: (I) W includes both node i' and node i; and (II) W includes neither node i' nor node i.

Proof of ($\rightarrow$; Case I): Suppose W includes both node i and i'. Because i is a leaf node, we may assume $i(m) = i$, $i(m-1) = i'$, and $i(m-2) = N_k(i')$ for some $k \in \{2,3\}$. Because W is a directly traversable walk, only its first or last node may be repeated; therefore, i' must be distinct from $i(1), i(2), \ldots, i(m-2)$. By assumption, i' must also be distinct from $i(0) = j$. Therefore, after CONTRACT-1(i) has been performed, the sub-walk $(j =)\ i(0)$–$i(1)$–$\cdots$–$i(m-2)$ $(= N_k(i'))$ remains an SDT(j) walk in G' and its length is $l - c_{e(i',k)} - c_{e(i',1)}$ in both G and G'. Observe that CONTRACT-1(i) has inserted in G' the new leaf node i_k and the new edge $[N_k(i'), i_k]$ with length $c_{e(i',k)} + c_{e(i',1)}$; thus, $i(0)$–$i(1)$–$\cdots$–$i(m-2)$ can be extended in G' to $i(0)$–$i(1)$–$\cdots$–$i(m-2)$–i_k with length $l - c_{e(i',k)} - c_{e(i',1)} + (c_{e(i',k)} + c_{e(i',1)}) = l$. Because i_k is a leaf node, this walk is

maximal in G', thereby completing the proof of this case.

Proof of ($\rightarrow$; Case II): Suppose W includes neither node i nor node i'. Note that all edges removed by CONTRACT-1(i) were incident to node i'. Therefore, we must have that $i(0)$–$i(1)$–$\cdots$–$i(m)$ remains an SDT(j) walk in G', and its length remains l. Furthermore, if i' is adjacent to $i(m)$, it must be the case that $[i(m-1), i(m)]$ and $[i(m), i']$ form an acute angle. Therefore, if CONTRACT-1(i) added any new edges that are adjacent to $i(m)$, that edge will also form an acute angle with $[i(m-1), i(m)]$. Consequently, W remains maximal in G', thus completing the proof of this direction.

Proof of ($\leftarrow$): Let W, with node representation $(j =)$ $i(0)$–$i(1)$–$\cdots$–$i(m)$, be a maximal SDT(j) walk of length l in G'. We prove that G contains a maximal SDT(j) walk of length l in two cases: (I) W includes either node i_2 or node i_3; and (II) W includes neither node i_2 nor i_3.

Proof of ($\leftarrow$; Case I): Suppose W includes node $N_k(i')$. If $N_k(i') = i(m)$, it must be the case that $[i(m-1), i(m)]$ and $[i(m), i']$ form an acute angle in G. Therefore, we must have that $(j =)$ $i(0)$–$i(1)$–$\cdots$–$i(m)$ remains an SDT(j) walk in G, and its length remains l. In another case, suppose W includes node i_k as well. Because i_k is a leaf node, we may assume $i(m) = i_k$, and $i(m-1) = N_k(i')$ for some $k \in \{2, 3\}$. Before CONTRACT-1(i) has been performed, the sub-walk $(j =)$ $i(0)$–$i(1)$–$\cdots$–$i(m-1)$ $(= N_k(i'))$ was an SDT(j) walk in G and its length was $l - c_{e(i',k)} - c_{e(i',1)}$. Observe that CONTRACT-1(i) has removed in G the switch node i', the leaf node i and the edges $[N_k(i'), i']$ and $[i', i]$ with length $c_{e(i',k)}$ and $c_{e(i',1)}$, respectively; thus, $i(0)$–$i(1)$–$\cdots$–$i(m-1)$ can be extended in G to $i(0)$–$i(1)$–$\cdots$–$i(m-1)$–i'–i with length $l - c_{e(i',k)} - c_{e(i',1)} + (c_{e(i',k)} + c_{e(i',1)}) = l$. Because i is a leaf node, this walk is maximal in G, thereby completing the proof of this case.

Proof of ($\leftarrow$; Case II): Suppose W includes neither node i_2 nor i_3. Note that all edges removed by CONTRACT-1(i) were incident to node i'. Therefore, we must have that $i(0)$–$i(1)$–$\cdots$–$i(m)$ remains an SDT(j) walk in G, and its length remains l.

The proof of part (b) is identical to the proof of part (a) after replacing "SDT(j)" throughout the proof with "r-SDT(j)." □

In the absence of directly traversable cycles, we have the following additional characterization of maximal SDT(j) and maximal r-SDT(j) walks. This result is useful because there will always exist a maximal SDT(j) walk among the set of maximum-length SDT(j) walks, and likewise for r-SDT(j).

Lemma 2 *If G contains no directly traversable cycles, then the following statement is true for every switch node $j \in N^S$: every maximal SDT(j) walk and every maximal r-SDT(j) walk ends at a leaf node.*

Proof. We prove the statement only for maximal SDT(j) walks because the proof is analogous for maximal r-SDT(u) walks after replacing "SDT" with "r-SDT." Let $(j =)\ i(0)$–$i(1)$–$\cdots$–$i(m)$ denote a maximal SDT(j) walk. If $i(m)$ is not a leaf node, then by Assumption 1, $i(m)$ is a switch node; thus, there exists a node $j' \in N$ such that the edge pair $\{[i(m-1), i(m)], [i(m), j']\}$ is directly traversable—if $i(m-1) = N_1(i(m))$, then $j' = N_2(i(m))$; else $j' = N_1(i(m))$. If $j' = i(h)$ for some $h \in \{0, 1, \ldots, m-1\}$, then we have identified the directly traversable cycle $i(h)$–$i(h+1)$–$\cdots$–$i(m)$–j' (a contradiction with the assumption that there are no directly traversable cycles); else, $i(0)$–$i(1)$–$\cdots$–$i(m)$–j' is an SDT(j) walk (a contradiction because $i(0)$–$i(1)$–$\cdots$–$i(m)$ is a maximal SDT(j) walk). □

Using Lemma 2, we may prove (in the vein of Lemma 1 for CONTRACT-1($\cdot$)) conditions under which CONTRACT-2($\cdot$) preserves MS_i and $\overline{MS}_i$.

Lemma 3 *Let i be a leaf node with the smallest value of $c_{e(i,1)}$. (That is, i is a leaf node with the shortest-length adjacent edge.) If $i \in \{N_2(i'), N_3(i')\}$ for some $i' \in N^S$, then let G' denote the network that results upon executing CONTRACT-2(i). For any $j \in N^S \setminus \{i'\}$ (a) the maximum length of any SDT(j) walk is the same in G and G', and (b) the maximum length of any r-SDT(j) walk is the same in G and G'.*

Proof. We will assume without loss of generality that $i = N_3(i')$ (i.e., $k = 3$ and $k' = 2$ in the definition of CONTRACT-2(i)) as the proof for $i = N_2(i')$ is symmetric. We prove part (a) by establishing that ($\rightarrow$) for every maximum-length, maximal SDT(j) walk in G, there exists an SDT(j) walk of the same length in G', and ($\leftarrow$) for every maximum-length, maximal SDT(j) walk in G', there exists an SDT(j) walk of the same length in G.

Proof of ($\rightarrow$): Let W, with node representation $(j =)\ i(0)$–$i(1)$–$\cdots$–$i(m)$, denote a maximum-length, maximal SDT(j) walk in G. We prove this result in three cases: (I) W includes neither i' nor i, (II) W includes i' but not i, and (III) W includes both i' and i.

Proof of ($\rightarrow$, Case I): If W includes neither i' nor i, then W is an SDT(j) walk with the same length in G', and there is nothing to prove.

Proof of ($\rightarrow$, Case II): If W includes i' but not i, then let $h\{1,2,\ldots,m-1\}$ denote the index such that $i(h) = i'$. (Note $h \neq 0$ because j is distinct from i' by assumption in the lemma's statement, and $h \neq m$ because we know that maximal SDT(j) walks end at a leaf nodes but $i' \in N^S$.) Because $i = N_3(i')$ is not traversed, we know that either $i(h-1) = N_1(i')$ and $i(h+1) = N_2(i')$ or $i(h-1) = N_2(i')$ and $i(h+1) = N_1(i')$. Therefore, the total length of the edges $[i(h-1), i(h)]$ and $[i(h), i(h+1)]$ is $c_{e(i',1)} + c_{e(i',2)}$. As a consequence of CONTRACT-2(i), nodes $i(h-1)$ and $i(h+1)$ are neighbors in G' such that $(j =)\ i(0)$–$i(1)$–$\cdots$–$i(h-1)$–$i(h+1)$–$i(h+2)$–$\cdots$–$i(m)$ is an SDT(j) walk in G'. Because the new edge $[i(h-1), i(h+1)]$ in G' has length $c_{e(i',1)} + c_{e(i',2)}$, this SDT($j$) walk has the same length as W.

Proof of ($\rightarrow$, Case III): If W includes both i' and i, then we must have that $i(m-1) = i'$ and $i(m) = i$—in this case we could construct another maximal SDT(j) walk W' by replacing $[i(m-1), i(m)] = e(i', 3)$ in W with $e(i', 2)$ and then iteratively extending the SDT(j) walk (as in the proof of Lemma 2 until it is no longer possible to add additional edges. By Lemma 2, W' ends at a leaf node; furthermore, because i is a leaf node with the smallest value of $c_{e(i,1)}$, the last edge of W' must have length at least $c_{e(i,1)}$. Because the edge $e(i, 1)$ is the only edge in W that is not in W', we have that W' is also a maximum-length, maximal SDT(j) walk. Case III is therefore proven as a consequence of Case II, thereby completing the proof of ($\rightarrow$).

Proof of ($\leftarrow$): Let W, with node representation $(j=)\ i(0)$–$i(1)$–$\cdots$–$i(m)$, denote a maximum-length, maximal SDT(j) walk in G'. Let e' denote the unique edge added to G' by CONTRACT-2(i). If e' is not traversed by W, then W is also an SDT(j) walk in G, and its length remains the same. We therefore assume e' is traversed by W, i.e., there exists $h \in \{1,\ldots,m\}$ such that $e' = [i(h-1), i(h)]$, where $c_{e'} = c_{e(i',1)} + c_{e(i',2)}$. In this case, $(j=)\ i(0)$–$i(1)$–$\cdots$–$i(h-1)$–i'–$i(h)$–$i(h+1)$–$\cdots$–$i(m)$ is an SDT(j) walk in G, and its length is the same in G' as in G. This completes the proof of ($\leftarrow$) as well as part (a).

The proof of part (b) is identical to the proof of part (a) after replacing "SDT(j)" throughout the proof with "r-SDT(j)." □

Given the above results, we are now in position to analyze PREPROCESS($\cdot$) for special cases of the network G.

3.3.1.2 Case 1: G contains no directly traversable cycles

We now demonstrate that if G contains no directly traversable cycles, we can solve PREPROCESS(j) in polynomial time. Although this assumption is not realistic in practice, it provides considerable simplifications that will extend naturally (see Section 3.3.1.3) to the more realistic case where all directly traversable cycles have length at least L.

Under the assumption that G contains no directly traversable cycles, we demonstrate that PREPROCESS(j) can be solved for all $j \in N^S$ by solving a single linear program (LP) that simultaneously determines the values MS_j and $\overline{MS}_j$ for every switch node $j \in N^S$. Naturally, $MS_j \geq L$ will be necessary and sufficient to imply the answer to PREPROCESS(j) is "yes."

Let y_i and $x_i,\ i \in N^S$, respectively be variables that represent MS_i and $\overline{MS}_i$. The LP is given as

$$\min \sum_{i \in N^S} y_i, \tag{3.2a}$$

$$\text{s.t. } y_i \geq c_{e(i,1)}, \qquad \forall i \in N^S : |N(N_1(i))| = 1, \tag{3.2b}$$

$$y_i \geq x_{N_1(i)} + c_{e(i,1)}, \qquad \forall i, N_1(i) \in N^S, N_1(N_1(i)) = i, \tag{3.2c}$$

$$y_i \geq y_{N_1(i)} + c_{e(i,1)}, \qquad \forall i, N_1(i) \in N^S, N_1(N_1(i)) \neq i, \tag{3.2d}$$

$$x_i \geq c_{e(i,k)}, \qquad \forall i \in N^S,\ k \in \{2,3\} : |N(N_k(i))| = 1, \tag{3.2e}$$

$$x_i \geq x_{N_k(i)} + c_{e(i,k)}, \qquad \forall i \in N^S,\ k \in \{2,3\} :\ N_k(i) \in N^S,\ N_1(N_k(i)) = i, \tag{3.2f}$$

$$x_i \geq y_{N_k(i)} + c_{e(i,k)}, \qquad \forall i \in N^S,\ k \in \{2,3\} :\ N_k(i) \in N^S,\ N_1(N_k(i)) \neq i, \tag{3.2g}$$

invoking Assumption 1 to eliminate degree-two nodes. The maximum space from a switch node i in the direction of its leaf node neighbor $N_k(i)$ is specified by Constraint (3.2b) for $k = 1$ and Constraint (3.2e) for $k \in \{2,3\}$. Constraints (3.2f) and (3.2g) establish lower bounds on the maximum space from a switch node i in the direction of its second or third neighbor, provided that the neighbor is a switch node. For a switch node i, the maximum space in the direction of its first neighbor, provided that the neighbor is a switch node, is specified in Constraints (3.2c) and (3.2d). The objective function (3.2a) ensures that we do not overestimate the space variables.

Before establishing the relevant properties of LP (3.2), we first illustrate its application to the notional yard network depicted in Figure 3.7. The LP for this example network is given as

$$\min\ y_3 + y_4 + y_6 + y_8, \tag{3.3a}$$

$$\text{s.t. } x_3 \geq 3, \tag{3.3b}$$

$$x_3 \geq 2, \tag{3.3c}$$

$$y_3 \geq x_4 + 2, \tag{3.3d}$$

$$x_4 \geq 2, \tag{3.3e}$$

$$x_4 \geq x_6 + 3, \tag{3.3f}$$

$$y_4 \geq x_3 + 2, \tag{3.3g}$$

$$x_6 \geq 3, \tag{3.3h}$$

$$x_6 \geq y_8 + 2, \tag{3.3i}$$

$$y_6 \geq y_4 + 3, \tag{3.3j}$$

$$x_8 \geq 2, \tag{3.3k}$$

$$x_8 \geq y_6 + 2, \tag{3.3l}$$

$$y_8 \geq 2. \tag{3.3m}$$

Constraints (3.3b), (3.3c), (3.3e), (3.3h), and (3.3k) specify lower bounds on the maximum space from nodes 3, 5, 6 and 8 in the direction of their leaf node neighbors. Note that x-variables are used to represent these quantities because the leaf nodes correspond to either the second or third neighbor of nodes 3, 5, 6 and 8. Constraint (3.3m) performs a similar function for node 8, ensuring $y_8 \geq 2$ because the adjacent leaf node is the first neighbor of node 8. The remaining constraints ensure that variables x and y of the switch nodes are in a correct relationship with their neighbors' variables based on how they are connected. For example, Constraint (3.3f) ensures that x_4 (the space from node 4 in the direction of nodes 5 and 6) should be at least 3 (the length edge $[4,6]$) plus x_6 (the space from node 6 in the direction of nodes 7 and 8). An optimal solution to this LP is $(x_3, x_4, x_6, x_8) = (3,7,4,10)$ and $(y_3, y_4, y_6, y_8) = (9,5,8,2)$. If $L = 6$, PREPROCESS(j)

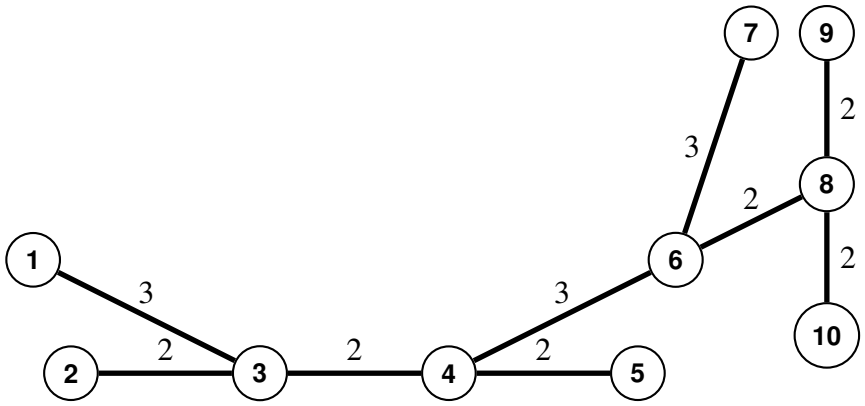


Figure 3.7: An acyclic network

would return "yes" for $j = 3$ and $j = 6$ (because $y_j \geq L$) and "no" otherwise.

The combination of Lemmas 1–3 yields a procedure for decomposing the yard network, one switch node at a time, in such a way that preserves MS_j and $\overline{MS}_j$. This will yield not only a proof (see Theorem 4) of correctness for LP (3.2), but specialized procedure to solve the LP (and PREPROCESS(j), $\forall j \in N^S$) that exploits network structure.

Lemma 4 *If G contains no directly traversable cycles, then G contains a leaf node i (that is connected to a switch node, say $i' \in N^S$) such that executing either CONTRACT-1(i) or CONTRACT-2(i) yields a new network G' that satisfies the following property: For every $j \in N^S \setminus \{i'\}$, (a) the maximum length SDT(j) walk is the same in G and G', and (b) the maximum length r-SDT(j) walk is the same in G and G'.*

Proof. This result follows as a direct consequence of Lemmas 1 and 3 upon selecting i as a leaf node that has the shortest adjacent edge. Based on Lemma 2, the leaf node i exists in G since it contains no directly traversable cycles. If $i = N_1(i')$, Lemma 1 provides the result upon executing CONTRACT-1(i); otherwise, Lemma 3 provides the result upon executing CONTRACT-2(i). □

As another consequence of the assumption that there are no directly traversable cycles, removing a leading sequence of edges from a maximum-length SDT(·) or r-SDT(·) walk will yield another maximum-length SDT(·) or r-SDT(·) walk.

Lemma 5 *Suppose* $(j =)\ i(0)$–$i(1)$–$\cdots$–$i(m)$ *is a maximum-length SDT(*j*) walk in a yard network* G *that contains no directly traversable cycles. Then for every* $h = 1, \ldots, m-1$*, the sub-walk* $i(h)$–$i(h+1)$–$\cdots$–$i(m)$ *is a maximum-length SDT(*$i(h)$*) walk (if* $i(h+1) = N_1(i(h))$*) and a maximum-length r-SDT(*$i(h)$*) walk otherwise.*

Proof. Suppose that $i(h+1) = N_1(i(h))$. By contradiction, let $(i(h) =)\ i'(0)$–$i'(1)$–$\cdots$–$i'(m')$ be an SDT($i(h)$) walk that is strictly longer than the SDT($i(h)$) walk $i(h)$–$i(h+1)$–$\cdots$–$i(m)$. We have either (I) $i(0)$–$i(1)$–$\cdots$–$i(h)$–$i'(1)$–$i'(2)$–$\cdots$–$i'(m')$ is an SDT(j) walk or (II) the node sets $\{i(0), i(1), \ldots, i(h)\}$ and $\{i'(1), i'(2), \ldots, i'(m')\}$ overlap. In case (I), the new walk must be longer than $i(0)$–$i(1)$–$\cdots$–$i(m)$ because $i'(0)$–$i'(1)$–$\cdots$–$i'(m')$ is longer than $i(h)$–$i(h+1)$–$\cdots$–$i(m)$ (a contradiction). Case (II) is also a contradiction because it implies existence of a directly traversable cycle, $i(\ell)$–$i(\ell+1)$–$\cdots$–$i(h)$–$i'(1)$–$i'(2)$–$\cdots$–$i'(\ell')$, where $\ell' \in \{1, \ldots, m'\}$ is the smallest index such that $i'(\ell') \in \{i(1), i(2), \ldots, i(h)\}$. □

The following definitions will aid in proving the correctness of LP (3.2). For $j \in N^S$, define

$$Q(j) \equiv \{i \in N^S :\ \text{there exists an SDT}(i)\ \text{walk that includes } j\}, \tag{3.4a}$$

$$\bar{Q}(j) \equiv \{i \in N^S :\ \text{there exists an r-SDT}(i)\ \text{walk that includes } j\}. \tag{3.4b}$$

Note that $Q(j) \cap \bar{Q}(j) \neq \emptyset$ implies existence of a closed directly traversable walk (by merging the SDT(i) and r-SDT(i) walk), which therefore implies existence of a directly traversable cycle. We may therefore assume $Q(j) \cap \bar{Q}(j) = \emptyset,\ \forall j \in N^S$. Further, because j is reachable along an SDT(i) or r-SDT(i) walk if and only if i is reachable along and SDT(j) or r-SDT(j) walk, $Q(j) \cup \bar{Q}(j)$ must equal the set of switch nodes that are reachable from node j.

We now build towards a proof that an optimal solution of the LP (3.2) establishes the value of MS_i. Towards this end, we first establish the impact of the subroutines CONTRACT-1($\cdot$) and CONTRACT-2($\cdot$) on LP (3.2). Both CONTRACT-1($\cdot$) and CONTRACT-2($\cdot$) will remove a single switch node from the network. With reference to the removed switch node i', it will simplify

exposition throughout the remainder of this section to rename variables $(x_i, y_i),\ i \in N^S \setminus \{i'\}$ as $(w_i^{(i')}, z_i^{(i')}),\ i \in N^S \setminus \{i'\}$ as follows:

$$\text{For } i' \in N^S,\ i \in Q(i')\text{: } w_i^{(i')} \equiv y_i \text{ and } z_i^{(i')} = x_i, \tag{3.5a}$$

$$\text{For } i' \in N^S,\ i \in \bar{Q}(i')\text{: } w_i^{(i')} \equiv x_i \text{ and } z_i^{(i')} = y_i. \tag{3.5b}$$

Lemma 6 *Let i be a leaf node that is the first neighbor of a switch node i', i.e., $i = N_1(i')$. Let G denote the original network, and let G' denote the network after CONTRACT-1(i) has been executed. Similarly, let LP(G) and LP(G') denote the LP* (3.2) *under each network, and let $\Gamma(G)$ and $\Gamma(G')$ denote the respective sets of variables associated with these LPs. Then LP(G') is a restriction of LP(G) in the sense that (a) $\Gamma(G') \subseteq \Gamma(G)$ and (b) any feasible solution to LP(G') can be extended into a feasible solution for LP(G) by assigning values to the variables in $\Gamma(G) \setminus \Gamma(G')$.*

Proof. Let i' denote the switch node that was removed by CONTRACT-1(i). The set of variables associated with LP(G') includes $(x_j, y_j),\ j \in N^S \setminus \{i'\}$, while the set of variables associated with LP(G) are $(x_j, y_j),\ j \in N^S$. This proves part (a) as $\Gamma(G') \cup \{x_{i'}, y_{i'}\} = \Gamma(G)$. Consider a solution $(x_j, y_j),\ j \in N^S \setminus \{i'\}$ to $LP(G')$. We now establish that the solution to $LP(G')$ can be extended to a solution $(\bar{x}_j, \bar{y}_j),\ j \in N^S$ to $LP(G)$ by assigning $\bar{y}_{i'} = c_{e(i',1)}$ and $\bar{x}_{i'} = \max\{\bar{z}_{N_2(i')}^{(i')} + c_{e(i',2)}, \bar{z}_{N_3(i')}^{(i')} + c_{e(i',3)}\}$. Assuming neither $N_2(i')$ nor $N_3(i')$ is a leaf node (with a description of simplification in these special cases to follow), the constraints that involve variables $y_{i'}$ and $x_{i'}$ in

the $LP(G)$ are:

$$y_{i'} \geq c_{e(i',1)}, \tag{3.6a}$$

$$x_{i'} \geq c_{e(i',2)} + \bar{z}^{(i')}_{N_2(i')}, \tag{3.6b}$$

$$\bar{w}^{(i')}_{N_2(i')} \geq c_{e(i',2)} + y_{i'}, \tag{3.6c}$$

$$x_{i'} \geq c_{e(i',3)} + \bar{z}^{(i')}_{N_3(i')}, \tag{3.6d}$$

$$\bar{w}^{(i')}_{N_3(i')} \geq c_{e(i',3)} + y_{i'}, \tag{3.6e}$$

Constraints (3.6a) is satisfied at equality by the given value for $\bar{y}_{i'}$. Constraints (3.6b) and (3.6d) must be satisfied because $\bar{x}_{i'}$ is set equal to $\max\{\bar{z}^{(i')}_{N_2(i')} + c_{e(i',2)}, \bar{z}^{(i')}_{N_3(i')} + c_{e(i',3)}\}$. On the other hand, LP($G'$) contains the constraints $\bar{w}^{(i')}_{N_2(i')} \geq c_{e(i',2)} + c_{e(i',1)}$ and $\bar{w}^{(i')}_{N_3(i')} \geq c_{e(i',3)} + c_{e(i',1)}$, respectively corresponding to the switch node $N_2(i')$ and $N_3(i')$ in the direction of the new leaf nodes constructed by CONTRACT-1(i). Because $y_{i'}$ has been set equal to $c_{e(i',1)}$, constraints (3.6c) and (3.6e) must be satisfied. If $N_2(i')$ is a leaf node, the proof is modified by setting $\bar{z}^{(i')}_{N_2(i')} = 0$ and removing constraint (3.6c); likewise, if $N_3(i')$ is a leaf node, set $\bar{z}^{(i')}_{N_3(i')} = 0$ and remove constraint (3.6e). Therefore, all constraints involving the two variables $y_{i'}$ and $x_{i'}$ are satisfied by the chosen setting, completing the proof of part (b). □

Lemma 7 *Let i be a leaf node with the smallest value of $c_{e(i,1)}$. (That is, i is a leaf node with the shortest-length adjacent edge.) If $i \in \{N_2(i'), N_3(i')\}$ for some $i' \in N^S$, then let G' denote the network that results upon executing CONTRACT-2(i). Let LP(G) and LP(G') denote LP* (3.2) *under the original and contracted network, and let $\Gamma(G)$ and $\Gamma(G')$ denote their respective sets of variables. Then LP(G') is a restriction of LP(G) in the sense that (a) $\Gamma(G') \subseteq \Gamma(G)$ and (b) any feasible solution to LP(G') can be extended into a feasible solution for LP(G) by assigning values to the variables in $\Gamma(G) \setminus \Gamma(G')$.*

Proof. With the exception of node i', all switch nodes from G remain switch nodes in G'; therefore, $\Gamma(G') \cup \{x_{i'}, y_{i'}\} = \Gamma(G)$, proving part (a).

We now prove part (b). Towards this end, we assume without loss of generality that $i = N_3(i')$ as the proof for $i = N_2(i')$ is symmetric. Let $(\bar{x}_j, \bar{y}_j),\ j \in N^S \setminus \{i'\}$, denote any feasible solution to LP(G'). In accordance with Equations (3.5), let $(\bar{w}_j^{(i')}, \bar{z}_j^{(i')}),\ j \in N^S \setminus \{i'\}$ denote the (fixed) values of $(w_j^{(i')}, z_j^{(i')})$ under solution $(\bar{x}_j, \bar{y}_j),\ j \in N^S \setminus \{i'\}$. Observe for $k \in \{1,2\}$ that $\bar{z}^{(i')}_{N_k(i')} = \bar{x}_{N_k(i')}$ and $\bar{w}^{(i')}_{N_k(i')} = \bar{y}_{N_k(i')}$ if $N_1(N_k(i')) = i'$; otherwise, $\bar{z}^{(i')}_{N_k(i')} = \bar{y}^{(i')}_{N_k(i')}$ and $\bar{w}^{(i')}_{N_k(i')} = \bar{x}_{N_k(i')}$.

To prove part (b), it suffices to prove that $x_{i'}$ and $y_{i'}$ can be assigned values such that the collection $\{\bar{x}_j : j \in N^S \setminus \{i'\}\} \cup \{\bar{y}_j : j \in N^S \setminus \{j\}\} \cup \{x_{i'}, y_{i'}\}$ satisfies the (exhaustive) set of constraints

$$x_{i'} \geq c_{e(i',3)}, \tag{3.7a}$$

$$y_{i'} \geq \bar{z}^{(i')}_{N_1(i')} + c_{e(i',1)}, \tag{3.7b}$$

$$\bar{w}^{(i')}_{N_1(i')} \geq x_{i'} + c_{e(i',1)}, \tag{3.7c}$$

$$x_{i'} \geq \bar{z}^{(i')}_{N_2(i')} + c_{e(i',2)}, \tag{3.7d}$$

$$\bar{w}^{(i')}_{N_2(i')} \geq y_{i'} + c_{e(i',2)}, \tag{3.7e}$$

that are included in LP(G) but not in LP(G'). (Note: In the above we have assumed $N_1(i')$ and $N_2(i')$ are not leaf nodes, but will summarize at the end of this proof the modifications that are necessary if this is not the case.)

We claim that the assignment $x_{i'} = \bar{z}^{(i')}_{N_2(i')} + c_{e(i',2)}$ and $y_{i'} = \bar{z}^{(i')}_{N_1(i')} + c_{e(i',1)}$ satisfies the system (3.7) and therefore proves the result. This assignment immediately satisfies Constraints (3.7b) and (3.7d) at equality. Additionally, due to Constraints (3.2c), (3.2d), (3.2f), and (3.2g) of LP(G') corresponding to node $i = N_1(i')$ in the direction of node $N_2(i')$, we have that

$$\bar{w}^{(i')}_{N_1(i')} \geq \bar{z}^{(i')}_{N_2(i')} + c_{e(i',1)} + c_{e(i',2)}, \tag{3.8}$$

and corresponding to node $i = N_2(i')$ in the direction of node $N_1(i')$, we have that

$$\bar{w}^{(i')}_{N_2(i')} \geq \bar{z}^{(i')}_{N_1(i')} + c_{e(i',1)} + c_{e(i',2)}. \tag{3.9}$$

Therefore, upon substituting $x_{i'}$ into (3.8) and $y_{i'}$ into (3.9), Constraints (3.7c) and (3.7e) are satisfied with respect to this assignment.

To complete the proof, Constraint (3.7a) must be dominated by Constraint (3.7d) as a consequence of our selection of i. To see this, let $(i' =)\ i(0)$–$i(1)$–$\cdots$—$i(m)$ (with edge representation $e(1)$–$e(2)$–$\cdots$–$e(m)$) denote any maximal SDT(i') walk in G such that $i(1) = N_2(i')$. Then, LP(G) includes the constraint set

$$x_{i'} \geq \bar{z}^{(i')}_{i(1)} + c_{e(1)}, \tag{3.10a}$$

$$\bar{z}^{(i')}_{i(h)} \geq \bar{z}^{(i')}_{i(h+1)} + c_{e(h+1)},\ \forall h = 1, \ldots, m-2, \tag{3.10b}$$

$$\bar{z}^{(i')}_{i(m-1)} \geq c_{e(m)}, \tag{3.10c}$$

where $\bar{z}^{(i')}_{i(m)}$ does not appear on the right-hand side of Constraint (3.10c) because (via Lemma 2), maximal SDT(i') walks must end at a leaf node. Summing Constraints (3.10) yields $x_{i'} \geq \sum_{h=1}^{m} c_{e(h)} \geq c_{e(m)}$. By assumption in this lemma's statement, $i(m)$ is either the second or third neighbor of $i(m-1)$, and by selecting i as a leaf node with the minimum-cost adjacent edge, we must have $c_{e(m)} \geq c_{e(j,3)}$. Thus, we have that $x_{i'} \geq c_{e(j,3)}$ and Constraint (3.7a) is satisfied.

Finally, we address the case where either $N_1(i')$ or $N_2(i')$ is a leaf node. If $N_1(i')$ is a leaf node, the proof proceeds exactly as above after assigning the value $\bar{z}^{(i')}_{N_1(i')} = 0$ and removing constraint (3.7c). Similarly, if $N_2(i')$ is a leaf node, apply the above proof after assigning $\bar{z}^{i'}_{N_2(i')} = 0$ and removing constraint (3.7e). □

Lemmas 6–7 lead to the following proof of correctness for LP (3.2).

Theorem 4 *If G contains no directly traversable cycles, then (a) LP* (3.2) *is feasible and (b) $y_j = MS_j,\ \forall j \in N^S$ in every optimal solution.*

Proof. By iteratively selecting a leaf node i with the shortest adjacent edge, we can execute either CONTRACT-1(i) or CONTRACT-2(i), depending on whether or not i is the first neighbor of some switch node. By selecting i in this way, Lemma 4 guarantees the resulting network will preserve $MS_j,\ j \in N^S$, in each iteration, with the exception that $MS_{i'}$ (where i' is the neighbor of the leaf node i) is no longer defined. Because each iteration removes a switch node, the number of iterations will be $P \equiv |N^S|$. Therefore, for $p = 0, 1, \ldots, P$, let G^p denote the network that results after p iterations and let LP(G^p) denote the corresponding instance of LP (3.2). Define $N^S(p)$ as the set of switch nodes in G^p such that

$$N^S = N^S(0) \supseteq N^S(1) \supseteq \cdots \supseteq N^S(P) = \emptyset. \tag{3.11}$$

We now complete the proof in two parts: We first establish the existence of a feasible solution for which $y_j = MS_j$ for every switch node $j \in N^S$, thus establishing result (a); we then establish result (b) by proving that MS_j is a lower bound on feasible y_j, implying that $y_j = MS_j,\ \forall j \in N^S$ for any optimal solution under Objective (3.2a).

To prove result (a), we prove the following statement inductively: For all $p = 0, 1, \ldots, P$, there exists a feasible solution to LP(G^p) in which $y_j = MS_j$ and $x_j = \overline{MS}_j,\ \forall j \in N^S(p)$. The base case, $p = P$, is trivial because $N^S(P) = \emptyset$; thus, LP(G^P) has no variables or constraints and is therefore vacuously feasible.

We will now assume the result holds for a given value of p and prove the result must also hold for $p-1$. Let i' denote the (unique) switch node in $N^S(p-1) \setminus N^S(p)$, let i denote the leaf node that was contracted to remove node i'. Let $(\bar{x}_j, \bar{y}_j),\ j \in N^S(p)$ denote a fixed solution that satisfies the induction hypothesis, i.e., $\bar{y}_j = MS_j$ and $\bar{x}_j = \overline{MS}_j,\ \forall j \in N^S(p)$. By applying Lemma 6 (if $i = N_1(i')$) or Lemma 7 (if $i = N_2(i')$), we may assign values to $x_{i'}$ and $y_{i'}$ such that the collection $\{\bar{x}_j, \bar{y}_j :\ j \in N^S(p)\} \cup \{x_{i'}, y_{i'}\}$ is feasible to LP(G^{p-1}). We now prove that this assignment provides the required solution to establish the case $p-1$ in the inductive proof.

Towards proving that $y_{i'} = MS_{i'}$ in case $p-1$ (denoted as "Result I"), let $(i' =)\ i(0)$–

$i(1)$–$\cdots$–$i(m)$ (with edge representation $e(1)$–$e(2)$–$\cdots$–$e(m)$ such that $e(1) = e(i',1)$) denote a maximum-length SDT(i') walk in G^{p-1}. By Lemma 4, we have that

$$MS_{i'} = \sum_{h=1}^{m} c_{e(h)}. \tag{3.12}$$

We have either $i(2) = N_1(i(1))$, which we denote as "Result I, Case A" or succinctly "Case (I:A)," or $i(2) \in \{N_2(i(1)), N_3(i(1))\}$, which we denote as "Case (I:B)". We prove that Lemmas 6–7 set $y_{i'} = MS_{i'}$ in each of these cases.

In case (I:A), suppose $i(2) = N_1(i(1))$. By Lemma 5, $i(1)$–$i(2)$–$\cdots$–$i(m)$ is a maximum-length SDT($i(1)$) walk. We proceed with the proof of this case by assuming that CONTRACT-2(i) was executed in iteration p, and we end the proof of this case by summarizing the simplifications that result if CONTRACT-1(i) was executed instead. Lemma 7 sets

$$\begin{aligned}
y_{i'} &= c_{e(i',1)} + \bar{z}^{(i')}_{N_1(i')}, && \text{(3.13a)}\\
&= c_{e(1)} + \bar{z}^{(i')}_{i(1)}, && \text{(3.13b)}\\
&= c_{e(1)} + \bar{y}_{i(1)}, && \text{(3.13c)}\\
&= c_{e(1)} + MS_{i(1)}, && \text{(3.13d)}\\
&= c_{e(1)} + \sum_{h=2}^{m} c_{e(h)}, && \text{(3.13e)}
\end{aligned}$$

which is equal to $MS_{i'}$ by Equation (3.12), thereby yielding the result. In the above, Equation (3.13a) is taken directly from Lemma 7. Equation (3.13b) holds because $i(1) = N_1(i')$ in order for ($i' = i(0)$)–$i(1)$–$\cdots$–$i(m)$ to be an SDT(i') walk. Equation (3.13c) employs the substitution $\bar{z}^{(i')}_{i(1)} = \bar{y}_{i(1)}$ from Equation (3.5) because i' is reachable from $i(1)$ along the r-SDT($i(1)$) walk $i(1)$–$i(0)$. Equation (3.13d) invokes the induction hypothesis, and Equation (3.13e) employs Lemmas 5–Lemma 4, which state that (Lemma 4) $MS_{i(1)}$ must equal the length of the longest SDT($i(1)$) walk in G^{p-1} and (Lemma 5) the length of this walk is $\sum_{h=2}^{m} c_{e(h)}$. To complete the proof of this case, observe that if CONTRACT-1(i) was executed in iteration p, then $m = 1$ and Equations (3.13)

hold upon setting $\bar{z}^{(i')}_{N_1(i')} = 0$ (as specified by Lemma 6).

In case (I:B), Suppose $i(2) \in \{N_2(i(1)), N_3(i(1))\}$. By Lemma 5, $i(1)$–$i(1)$–$\cdots$–$i(m)$ is a maximum-length r-SDT($i(1)$) walk. In this case, the proof is analogous to case I with the modification that $\bar{y}_{i(1)}$ is replaced by $\bar{x}_{i(1)}$ (because i' is now reachable along the SDT($i(1)$) walk $i(1)$–$i(0)$) and $MS_{i(1)}$ is replaced by $\overline{MS}_{i(1)}$ (upon invoking the induction hypothesis). The analog of Equation (3.13e) holds upon invoking Lemmas 4–5 to guarantee that $\overline{MS}_j = \sum_{h=2}^{m} c_{e(h)}$. This completes the proof of Case (I:B) and Result I.

Towards proving that $x_{i'} = \overline{MS}_{i'}$ in case $p-1$ (denoted as "Result II"), let $(i' =)\ i(0)$–$i(1)$–$\cdots$–$i(m)$ (with edge representation $e(1)$–$e(2)$–$\cdots$–$e(m)$) denote a maximum-length r-SDT(i') walk in G^{p-1}. Assume without loss of generality that $i(1) = N_2(j)$, i.e., such that $e(1) = e(j,2)$. Lemma 4 now implies

$$\overline{MS}_j = \sum_{h=1}^{m} c_{e(h)}. \tag{3.14}$$

Analogous to result I, we have either $i(2) = N_1(i(1))$, which we denote "case (II:A)," or $i(2) \in \{N_2(i(1)), N_3(i(1))\}$ denoted "case (II:B)."

In case (II:A), suppose $i(2) = N_1(i(1))$ such that Lemma 5 guarantees $i(1)$–$i(2)$–$\cdots$–$i(m)$ is a maximum-length SDT($i(1)$) walk. We will first prove this case assuming CONTRACT-1(i) was executed in iteration p and then provide a summary of the simplifications that result if CONTRACT-2(i) was executed instead. Lemma 6 sets $x_{i'} = \max\left\{c_{e(i',2)} + \bar{z}^{(i')}_{N_2(i')}, c_{e(i',3)} + \bar{z}^{(i')}_{N_3(i')}\right\}$,

which yields

$$x_j = \max\left\{c_{e(i',2)} + \bar{z}^{(i')}_{N_2(i')}, c_{e(i',3)} + \bar{z}^{(i')}_{N_3(i')}\right\}, \tag{3.15a}$$

$$= \max\left\{c_{e(1)} + \bar{z}^{(i')}_{i(1)}, c_{e(i',3)} + \bar{z}^{(i')}_{N_3(i')}\right\}, \tag{3.15b}$$

$$= \max\left\{c_{e(1)} + \bar{y}_{i(1)}, c_{e(i',3)} + \bar{z}^{(i')}_{N_3(i')}\right\}, \tag{3.15c}$$

$$= \max\left\{c_{e(1)} + MS_{i(1)}, c_{e(i',3)} + \bar{z}^{(i')}_{N_3(i')}\right\}, \tag{3.15d}$$

$$= \max\left\{c_{e(1)} + \sum_{h=2}^{m}, c_{e(i',3)} + \bar{z}^{(i')}_{N_3(i')}\right\}, \tag{3.15e}$$

$$= \max\left\{\overline{MS}_{i'}, c_{e(i',3)} + \bar{z}^{(i')}_{N_3(i')}\right\}, \tag{3.15f}$$

after applying the logic of (3.13) to the first term in the maximum. We now prove that $\overline{MS}_{i'} \geq c_{e(i',3)} + \bar{z}^{(i')}_{N_3(i')}$ (and therefore $x_{i'} = \overline{MS}_{i'}$. By Lemma 5, $\bar{z}^{(i')}_{N_3(i')}$ equals either $\overline{MS}_{N_3(i')}$ (if $N_1(N_3(i')) = i'$) or $MS_{N_3(i')}$ (otherwise); therefore, there exists a directly traversable walk W of length $\bar{z}_{N_3(i')}$ that begins at node $N_3(i')$. One of these cases yields that W is an r-SDT($N_3(i')$) walk and $N_3(i')$–i' is an SDT($N_3(i')$) walk while the other yields that W is an SDT($N_3(i')$) walk and $N_3(i')$–i' is an r-SDT($N_3(i')$) walk; thus, W cannot include i' (as this would imply $Q(i') \cap \bar{Q}(i') \neq \emptyset$ and the existence of a directly traversable cycle in G^{p-1}). Therefore, we can create an r-SDT(i') walk by merging $[i', N_3(i')] = e(i',3)$ with the edges of W. This walk has length $c_{e(i',3)} + \bar{z}^{(i')}_{N_3(i')}$, and we must therefore have $\overline{MS}_{i'} \geq c_{e(i',3)} + \bar{z}^{(i')}_{N_3(i')}$ because $\overline{MS}_{i'}$ is defined as the maximum length among r-SDT(i') walks. This completes the proof that $x_{i'} = \overline{MS}_{i'}$ if CONTRACT-1(i) is executed in iteration p. If CONTRACT-2(i) is executed instead, then the proof holds upon setting $\bar{z}^{(i')}_{N_3(i')} = 0$ as is specified by Lemma 7.

In case (II:B), suppose $i(2) \in \{N_2(i(1)), N_3(i(1))\}$. The proof of this case is identical to case (II:A) with the exception that $\bar{y}_{i(1)}$ is replaced by $\bar{x}_{i(1)}$ and $MS_{i(1)}$ is replaced by $\overline{MS}_{i(1)}$. The extension from (II:A) to (II:B) is analogous to the extension from (I:A) to (I:B).

With results I and II established, we have now completed inductive proof for case $p-1$, which therefore yields that LP(G^p) contains a feasible solution in which $y_j = MS_j$ and $x_j =$

$\overline{MS}_j,\ \forall j \in N^S(p)$, for all cases $p = 0, 1, \ldots, p$. In particular, case $p = 0$ yields that $y_j = MS_j$ and $x_j = \overline{MS}_j,\ \forall j \in N^S$, is feasible for LP (3.2), thus completing the proof of result (a).

To prove result (b), we establish that $y_j \geq MS_j,\ \forall j \in N^S$, for every feasible solution to LP (3.2). For any switch node $j \in N^S$, let $(j =)\ i(0)$–$i(1)$–$\cdots$–$i(m)$ (with edge representation $e(1)$–$e(2)$–$\cdots$–$e(m)$) be any maximum-length, maximal SDT(j) walk such that $MS_j = \sum_{h=1}^{m} c_{e(h)}$. Then LP (3.2) contains the constraints

$$y_u \geq z^{(j)}_{i(1)} + c_{e(1)}, \tag{3.16a}$$

$$z^{(j)}_{i(h)} \geq z^{(j)}_{i(h+1)} + c_{e(h+1)},\ \forall h = 1, \ldots, m-2, \tag{3.16b}$$

$$z^{(j)}_{i(m-1)} \geq c_{e(m)}, \tag{3.16c}$$

where $z^{(j)}_{i(m)}$ does not appear on the right-hand side of (3.16c) because Lemma 2 guarantees that $i(m)$ is a leaf node. Summing inequalities (3.16) yields

$$y_j \geq \sum_{h=1}^{m} c_{e(h)} = MS_j. \tag{3.17}$$

Note that the feasible solution with $y_j = MS_j,\ \forall j \in N^S$, as established in result (a), simultaneously achieves the lower bound provided for each $y_j,\ j \in N^S$. Because all y-variables appear with a positive coefficient of Objective (3.2a), this solution must be optimal for (3.2). For the same reason, any solution with $y_j > MS_j$ for some $j \in N^S$ would be sub-optimal for LP (3.2), thereby proving result (b). □

Theorem 4 states that the preprocessing problem can simultaneously be solved for all the switch nodes in the yard with one LP formulation, LP (3.2). The theorem will also help us to prove that the preprocessing problem is polynomially solvable for the next case.

3.3.1.3 Case 2: Yard network has minimum directly traversable cycle length at least L

In this subsection, we demonstrate that PREPROCESS(i) remains polynomially solvable provided that the yard network does not contain directly traversable cycles of length less than L. For this case, we propose a modified version of the LP (3.2), using the same variables defined in the LP (3.2), to solve PREPROCESS(i). As opposed to the acyclic case, the procedure for this case will require solving a different LP associated with each switch node $i \in N_S$. We again invoke Assumption 1 to remove nodes of degree two, and we then modify network $G = (N, E)$ to obtain a new network $G' = (N', E')$ as follows: (a) Let $j_2 \equiv N_2(i)$ and $j_3 \equiv N_3(i)$ denote the second and third neighbors of i in the original network; (b) create four new nodes to define $N' = N \cup \{j_2', j_2'', j_3', j_3''\}$; (c) set $E' = E \cup \{[i, j_2'], [j_2'', j_2], [i, j_3'], [j_3'', j_3]\} \setminus \{[i, j_2], [i, j_3]\}$; (d) respectively define the second and third neighbors of i as j_2' and j_3'; (e) in a similar fashion, define j_2'' and j_3'' as the appropriately numbered neighbors of j_2 and j_3, replacing i's position in their previous adjacency list; and (f) respectively assign lengths 0, c_{ij_2}, 0, and c_{ij_3} to edges $[i, j_2']$, $[j_2'', j_2]$, $[i, j_3']$, and $[j_3'', j_3]$. This modification is intended to disconnect the node i from its second and third neighbors as i should not be adjacent to $N_2(i)$ or $N_3(i)$ in any SDT(i) feasible position. Note that i is a switch node in the transformed network, so the variable y_i is included in the LP (3.2).

With some minor modifications, the LP (3.2) can be applied to G' to solve PREPROCESS(i), where the optimal value of y_i will (under ideal circumstances) indicate that i is indirectly traversable if $y_i \geq L$—we will refer to this LP as LP$_i$. Having $y_i \geq L$ in an optimal solution to LP$_i$ turns out to be a sufficient condition to prove that i is indirectly traversable, but it is not a necessary condition. In fact, an indirectly traversable node i may lead to an infeasible LP$_i$ in certain cases. We will demonstrate that infeasibility of LP$_i$ results when LP$_i$ includes constraints that correspond to edges that form a directly traversable cycle in G'. Provided that such a directly traversable cycle is reachable from i, the minimum-cycle-length condition will allow us to use infeasibility of LP$_i$ to conclude that i is indirectly traversable. Towards solving PREPROCESS(i) in this manner, we must first remove from G' any edges that are not contained in any SDT(i) walk. We accomplish this by performing a breadth first search starting from node i to identify which edges' constraints

should remain in LP_i.

We now prove the necessary results that allow us to use the infeasibility of LP_i or having $y_i \geq L$ in an optimal solution to LP_i to conclude that the answer to PREPROCESS(i) is "yes." Before explaining these results, we note that a feasible LP_i must have an optimal solution since all the x variables are nonnegative and zero is a lower bound on the objective value.

Lemma 8 *Suppose LP_i is feasible. In any optimal solution to LP_i, the value of y_i is no less than MS_i.*

Proof. Let $e(1)$–$e(2)$–$\cdots$–$e(m)$ (with node representation $i(0)$–$i(1)$–$\cdots$–$i(m)$) denote a SDT(i) walk of length MS_i. We prove that $y_i \geq MS_i$ in any solution to LP_i. We prove the result for the case where $N_1(i(h-1)) = i(h)$ for all $h \in \{1,2,\ldots,m\}$ and $i(k)$ is a switch node. (For $h = 1,\ldots,m$, if $i(h) \in \{N_2(i(h-1)), N_3(i(h-1))\}$ the proof can be modified by replacing $y_{i(h)}$ throughout the proof with $x_{i(h)}$.) With this simple directly traversable walk, we can extract from LP_i the constraints

$$y_{i(h-1)} \geq c_{e(h)} + y_{i(h)}, \forall h = 1,\ldots,k. \tag{3.18}$$

Summing these equations yields

$$y_i = y_{i(0)} \geq c_{e(1)} + c_{e(2)} + ... + c_{e(k)} + y_{i(k)}. \tag{3.19}$$

Because $c_{e(1)} + c_{e(2)} + \cdots + c_{e(k)} = MS_i$ and $y_{i(k)} \geq 0$, this establishes $y_i \geq MS_i$. □

Lemma 9 *Suppose LP_i is feasible. In any optimal solution to LP_i, the value of y_i exceeds the value MS_i only if it is possible to reach a node j in a cycle W via an SDT(i) walk such that W is indirectly traversable when oriented such that j is its first and last node.*

Proof. We prove that if it is not possible to reach such a node j via an SDT(i) walk, then the value y_i in an optimal solution of LP_i is equal to the value MS_i. Because we have pruned off

edges in G' that are not reachable via an SDT(i) walk, this assumption implies that LP_i is an instance of the acyclic LP (3.2). Based on Lemma 4, the value of y_i in an optimal solution equals (and therefore does not exceed) MS_i. □

Lemma 10 *LP_i is infeasible only if it is possible to reach a node j in a cycle W via an SDT(i) walk such that W is directly traversable when oriented such that j is its first and last node.*

Proof. Analogous to the proof of Lemma 9, LP_i corresponds to an instance of the acyclic LP (3.2) if no such node j exists, and Theorem 4 guarantee feasibility of this LP. □

Theorem 5 *Provided all cycles have length at least L, switch node $i \in N_S$ is indirectly traversable if and only if (a) LP_i is infeasible or (b) $y_i \geq L$ in an optimal solution of LP_i.*

Proof. This result follows from Lemmas 8–10. Lemma 8 tells us that finding $y_i < L$ in the solution of the LP_i is sufficient to establish that switch node i is not indirectly traversable. Based on Lemma 9, finding $y_i \geq L$ in the solution of the LP_i means either $MS_i \geq L$ or there is a cycle of length at least L that is reachable (and directly traversable) via an SDT(i) walk. In both cases, the switch node i is indirectly traversable. Lemma 10 also states that an infeasible LP_i means that there is there is a cycle of length at least L that is reachable (and traversable) via an SDT(i) walk, and i is therefore indirectly traversable. □

Theorem 6 *The preprocessing problem is polynomially solvable when all cycles are at least L in length.*

Proof. Based upon Theorem 5, solving PREPROCESS(i) corresponds to solving LP_i, whose formulation is polynomial in the size of the PREPROCESS(i) input. Linear programs are polynomially solvable (Khachiyan, 1980). □

3.3.1.4 Case 3: G contains a directly traversable cycle of length less than L

As explained in Section 3.3, PREPROCESS(·) is NP-complete in general. In this subsection, we propose two modeling approaches for this case of PREPROCESS(·). The first one is an algorithm that, for a specific switch node $i \in N^S$, searches the yard network to determine if there is a feasible position for the train anchored at node i. There are two constraints involved in this search algorithm: (1) it does not search the routes that are not directly traversable; (2) it ensures that the routes do not visit any edges twice.

A dynamic programming algorithm (Algorithm 3) is proposed to solve the preprocessing problem. This algorithm invokes an auxiliary function represented in Algorithm 4 to compare MS_i to L for a given switch node $i \in N^S$. In Algorithm 3, we define a set V to ensure we do not visit any edges twice. In line 3 of Algorithm 3, we loop over all the switch nodes in the network G to determine whether they are traversable or not. For a switch node i, we check the length of the first edge from node i in the direction of node $N_1(i)$ ($[i, N_1(i)]$). If the length is greater than L units, then that specific switch node is traversable. Otherwise, we check whether there is a space of at least length $L - c_{iN_1(i)}$ at node $N_1(i)$ in the direction of its neighbors except node i. We continue to do this until we finish the search process for the switch node i. We keep a record of the nodes that we visit during this search with the set V. Whenever we reach a node that is already in the set V we stop that branch of the search. There exist four cases when we continue the search from a node i in the direction of node j: (1) node j is a leaf node (2) node j has two outgoing edges; (3) node j is a switch node and its first neighbor is node i; (4) node j is a switch node and its first neighbor is not node i. These four cases are considered in lines 1, 3, 14, and 27 of Algorithm 4, respectively. The complexity of the proposed algorithm greatly depends on the proportion of switch nodes in the network and how the switch nodes are connected to each other, besides the input parameter L. The greater the density of switch nodes in the network, the more time will be needed to solve the preprocessing problem and the worst case complexity is exponential in the size of the node set. However, in practical cases where the proportion of switch nodes in the network is very small, the proposed algorithm would be efficient.

Algorithm 3: Preprocessing

input : $L, N^S, G = (N, A)$
output: N^L

Set $N^L = \emptyset$
Set $V = \emptyset$
for $i \in N^S$ **do**
  if $c_{iN_1(i)} > L$ **then**
    $N^L = N^L \cup \{i\}$
  else
    $V = V \cup \{i, N_1(i)\}$
    if $f(i, N_1(i), L - c_{iN_1(i)}, V)$ **then**
      $N^L = N^L \cup \{i\}$
    end
  end
  Set $V = \emptyset$
end

3.3.2 Routing Stage

The information on whether or not a train of specific length can traverse the walk $[N_2(i), i]$–$[i, N_3(i)]$ indirectly at each switch node $i \in N^S$ (i.e., the set N^L) is obtained by the proprocessing stage. The routing stage finds a shortest route using the information obtained from the preprocessing stage. The routing problem for a given origin-destination pair and a train of length L is defined below.

ROUTING(O, D, L)

Instance: A yard network $G = (N, E)$ with edge costs $c_e \geq 0,\ e \in E$; a train length $L \geq 0$; a set of switch nodes N^L; the origin node i_O; the destination node i_D.

Problem: Find a walk $i(0)$–$i(1)$–$\cdots$–$i(m)$ that minimizes the value $Ln_1 + \sum_{j=1}^{m} c_{[i(j-1),i(j)]}$ where $i(0) = i_O$, $i(m) = i_D$, n_1 is the number of nodes in the walk that are indirectly traversed, and the edge pair $[i(j), i(j+1)]$–$[i(j+1), i(j+2)]$ is either directly or indirectly traversable for all $j \in \{0, \ldots, m-2\}$.

There are a set of assumptions in the routing problem. The train starts its route from the

Algorithm 4: Auxiliary function (f)

input : $i, N_1(i), L, V$
output: true\false
if $|N(N_1(i))| = 1$ **then**
  return false
else if $|N(N_1(i))| = 2$ *and* $N(N_1(i)) = \{i, j\}$ **then**
  if $L < c_{N_1(i)j}$ **then**
    return true
  else
    if $j \notin V$ **then**
      $V = V \cup \{j\}$
      if $f(N_1(i), j, L - c_{N_1(i)j}, G, V)$ **then**
        return true
      end
    end
  end
else if $N_1(N_1(i)) = i$ **then**
  for *node* $j(\neq i)$ *such that* j *is a neighbor of* $N_1(i)$ **do**
    if $L < c_{N_1(i)j}$ **then**
      return true
    else
      if $j \notin V$ **then**
        $V = V \cup \{j\}$
        if $f(N_1(i), j, L - c_{N_1(i)j}, G, V)$ **then**
          return true
        end
      end
    end
  end
else
  $j = N_1(N_1(i))$
  if $L < c_{N_1(i)j}$ **then**
    return true
  else
    if $j \notin V$ **then**
      $V = V \cup \{j\}$
      if $f(N_1(i), j, L - c_{N_1(i)j}, G, V)$ **then**
        return true
      end
    end
  end
end
return false

origin by pulling the cars. We consider the middle point of the train as its location in the yard. So when this middle point reaches the destination node, we say that the train has reached the destination location. This assumption impacts the total length of the route from the origin to the destination. When the train passes a switch node i indirectly, it must find a feasible position anchored at node i. Therefore, the middle point traverse an extra L units which should be added to the total length of the route from the origin to the destination. As mentioned in Section 3.2, if the destination node has one outlet, the locomotive must push the cars in backward into the track. So, in this case, the train can pass the switch nodes indirectly an odd number of times. Toward solving ROUTING(O,D,L), we first expand the network $G = (N,E)$ as follows:

- We expand the network to account for the direction of each edge that a train passes. So each non-switch node is expanded to two nodes to account for the direction of the train when passing the node. Let $G' = (N',A)$ be the modified network.

- We make a duplicate copy of network G'. Let G'' denote this network. The reason that we have two identical networks G' (pull layer) and G'' (push layer) is to account for how the locomotive moves the cars behind it, whether it pushes or pulls them.

- We expand each switch node $i \in N^S$ to 6 nodes (2 for each direction, 3 for each adjacent edge) in each layer. Let $V'_k(i)$ and $V''_k(i),\ k \in \{1,2,3\}$ respectively denote the pair of push- and pull-layer nodes corresponding to $N_k(i)$.

- For each pair of nodes in $V'(i)$ that the train can traverse their adjacent edges directly (i.e., between $V'_1(i)$ and either $V'_2(i)$ or $V'_3(i)$), we construct a directed arc of length zero. We do the same for the set $V''(i)$.

- For each switch node i that can be traversed indirectly, create four directed arcs with length L: two between $V'_2(i)$ and $V''_3(i)$, and two between $V'_3(i)$ $V''_2(i)$. The reason for setting L as the length is that the train must find an SDT(i) feasible position to continue with the intended walk which adds L units to total length of the route that the train takes.

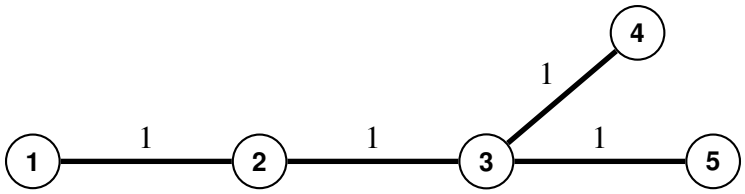


Figure 3.8: Initial Network

For instance, Figure 3.9 illustrates the expanded version of yard shown in Figure 3.8, in which we assume $\{[3,4],[4,5]\}$ is indirectly traversable. The cost of each green arc is zero as all of these arcs correspond to a directly traversable route through Figure 3.8. However, the blue arcs denote the move from one acute edge to the other one, which is not directly traversable. In order to this, the train must move onto an SDT(i) feasible position which increases the total length traveled by L units. Because traversing an indirectly traversable pair of edges requires reversing the train's direction, all of the blue arcs have one endpoint in the pull layer and one endpoint in the push layer.

After expanding the yard network G, we need to specify the origin and the destination. Since we assume that the locomotive starts with pulling the cars, we only consider the nodes in the pull layer of the expanded network G' that were originated from the origin node in G as the origin. So we connect these nodes to a dummy origin node with the right direction. To generate the destination node, we connect all the nodes in push and pull layers of the expanded network that were originated from the destination node in G to a dummy destination node, unless there is a restriction that the destination must be in pushlayer. However, if the destination node is a leaf, we set the node in the push layer as our destination. After expanding the yard network and determining the origin and destination node, we formulate the YRP as a shortest path problem on the expanded network to obtain a shortest route for the train which is an optimal solution to the YRP.

Remark 1 There are two ways to solve the YRP with the proposed two-stage approach for case 2 of the yard network. Since the value of y_i in the optimal solution to LP_i can be computed independently of the L-parameter, one can compute the optimal value of y_i for every switch node i and use this information to solve the routing stage for a specific train of length L. Another way

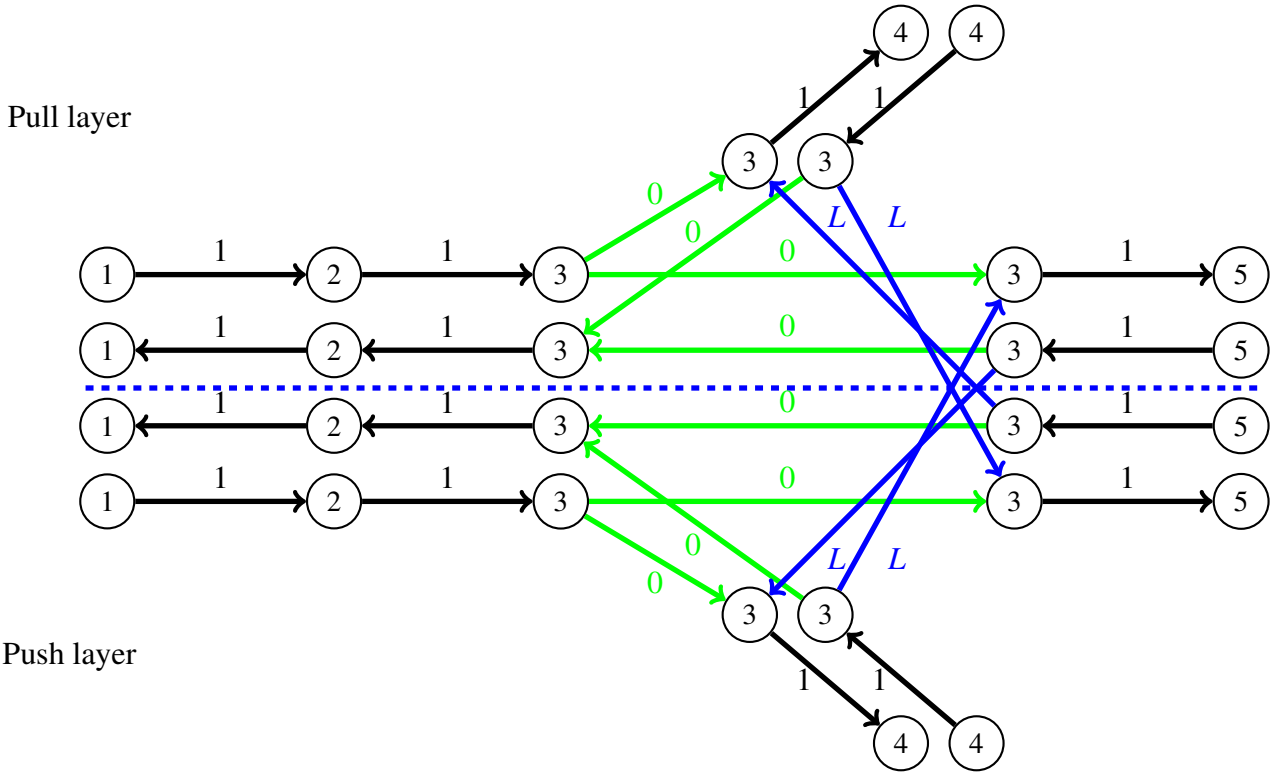


Figure 3.9: Expanded network of the yard instance in Figure 3.8

is to assume that all the switch nodes are indirectly traversable for train of length L and solve the routing problem first. Then preprocess the indirectly traversed switch nodes in the current solution one by one, update and solve the routing problem after each preprocess is solved until the route uses only switch nodes that have already been identified as indirectly traversable. Our preliminary experiments showed that the first way was very fast, and so we did not report results for the second approach.

We now present a YRP instance, the steps to model the problem and how to solve it. The YRP instance is shown in Figure 3.10. Edge lengths are shown in the arc labels. The dashed link represent other parts of the network that are directly traversable with a total length of 10 units. However, the left end point of the dashed link makes an acute angle with the edge $[3,4]$. In this instance, the length of the train is 2 units which makes all the switch nodes indirectly traversable. Since the destination node has one outlet, the destination node is in the push layer of the expanded network and there is no need to add a dummy destination. We assume that locomotive

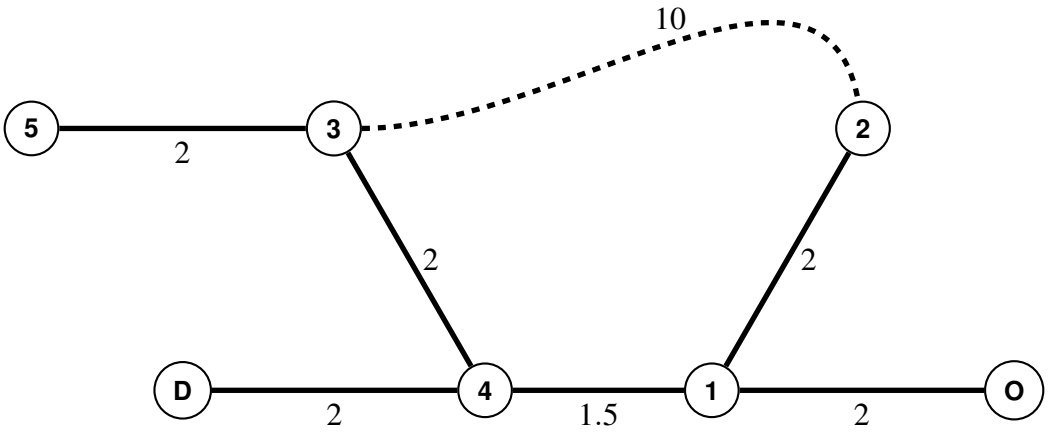


Figure 3.10: A YRP instance ($L = 2$)

starts the trip by pulling the cars, so the origin node is in the pull layer. There exist two possible solutions for this instance which are shown in Figure 3.11 and 3.12. As it can be seen in these two figures, the number of times that locomotive moves from one layer to another layer is an odd number to satisfy the requirement that it should reach the destination node by pushing the cars. We now proceed by proposing an alternative approach to solve the PREPROCESS($\cdot$) using the expanded yard network defined in this section.

3.3.3 An Alternative Preprocessing Approach

Using the expanded network of a yard network defined in Section s:chapter2routing, we propose an alternative approach to solve PREPROCESS(i). This approach is an integer programming formulation that takes, as its input, only one layer of the expanded version of network $G' = (N', E')$ described for switch node i in Section 3.3.1.3. Without loss of generality, we choose the pull layer of the expanded version. Define $\bar{G}'' = (\bar{N}'', \bar{A}'')$ as the pull layer of the expanded version of network G'. Note that there is no arc connecting the two layers, push and pull, in G'' when we use only one layer of the expanded version of network G'. This means that the train cannot be positioned in a way that it occupies the walk $[N_3(j), j]$–$[j, N_2(j)]$ where $j \in N'^S$. We can use the network G'' to find out whether there is feasible position for a train of length L at

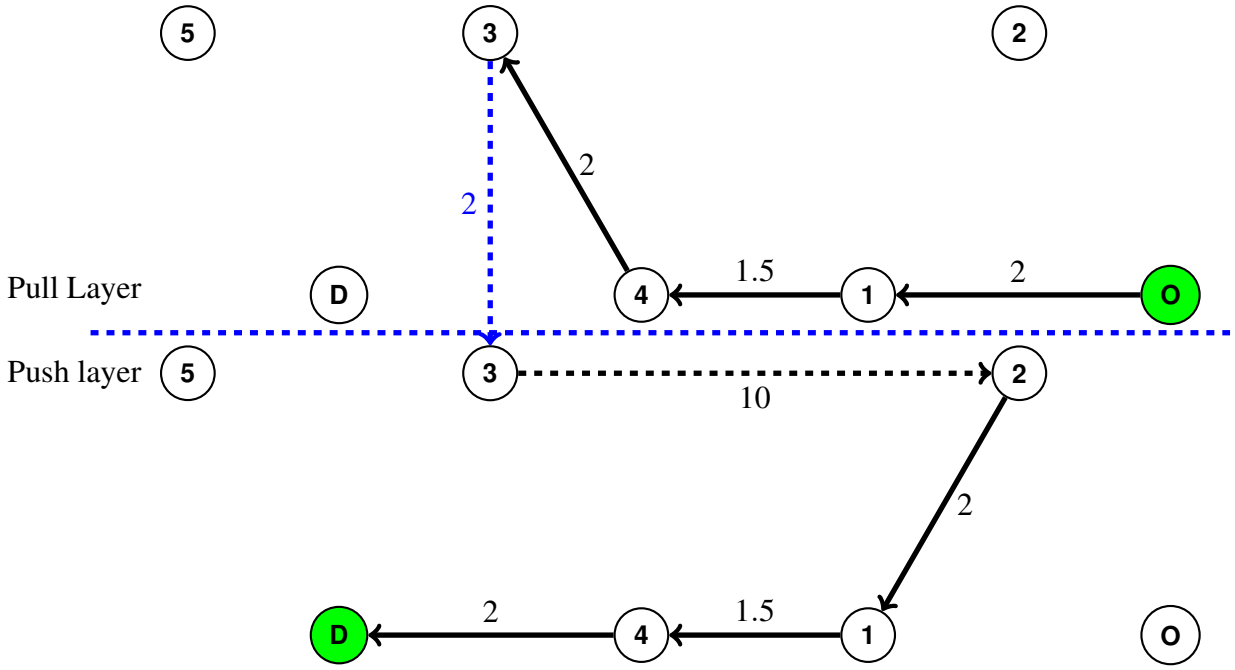


Figure 3.11: Optimal solution to instance of Figure 3.10 (objective value = 23)

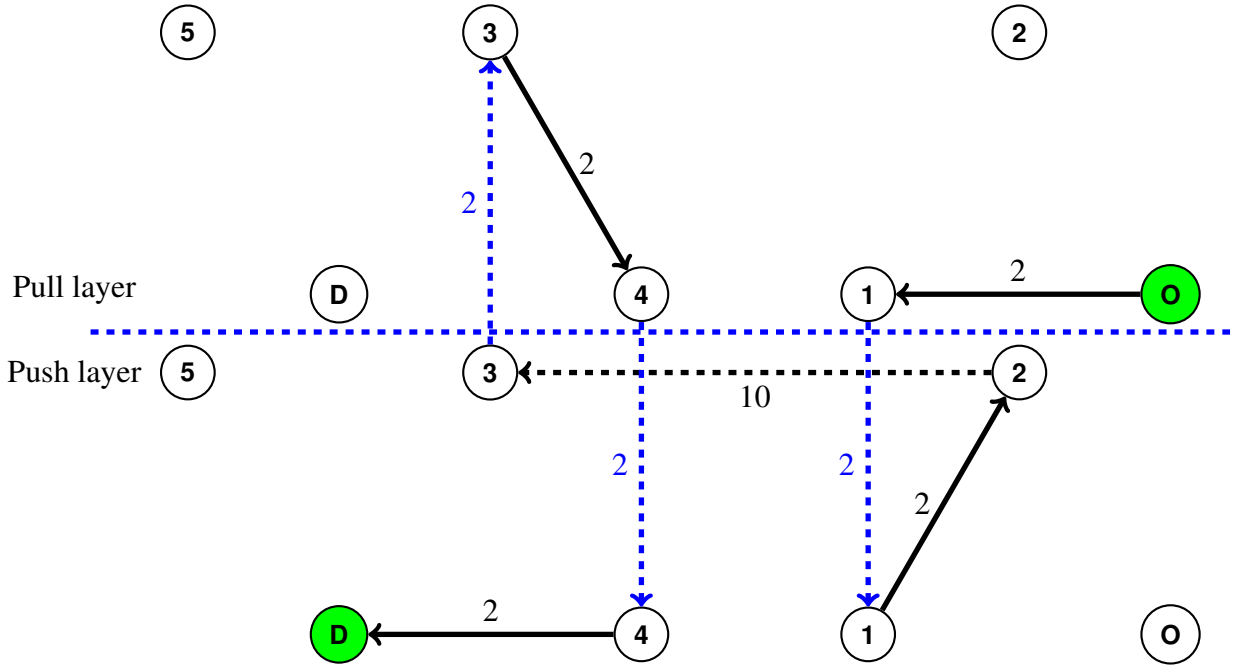


Figure 3.12: A feasible solution to instance of Figure 3.10 (objective value = 24)

node i in the direction of its first neighbor. Let x_{ij} ($(i,j) \in \bar{A}$) equal 1 if node i immediately precedes node j in the feasible position; else, let x_{ij} equal to 0. Let p_i equal 1 if node i is the first node in the position. Let q_i equal 1 if node i is the last node in the position.

$$\min \sum_{(i,j)\in A''} c_{ij}x_{ij} + \varepsilon \sum_{(i,j)\in A''} x_{ij}, \tag{3.20a}$$

$$\text{s.t.} \sum_{j:(i,j)\in A''} x_{ij} - \sum_{j:(j,i)\in A''} x_{ij} = p_i - q_i, \qquad i \in N'' \tag{3.20b}$$

$$\sum_{(i,j)\in A''} c_{ij}x_{ij} \geq L, \tag{3.20c}$$

$$\mathbf{x} \text{ contains no subtours}, \tag{3.20d}$$

$$\sum_{i\in N''} p_i = 1, \tag{3.20e}$$

$$\sum_{i\in N''} q_i = 1, \tag{3.20f}$$

$$p_i, q_i \in \{0,1\}, \qquad i \in N'' \tag{3.20g}$$

$$x_{ij} \in \{0,1\}, \qquad (i,j) \in A'' \tag{3.20h}$$

Constraints (3.20b) and (3.20c) ensure $\mathbf{x}$ defines a feasible position—with subtours excluded by Constraints (3.20d)—that begins at node $i \in N$ if $p_i = 1$ and ends at i if $q_i = 1$. Constraints (3.20e) and (3.20f) respectively specify that the position must start and end at exactly one node. Objective (3.20a) ensures that the formulation gives the smallest position possible and ε is introduced to break symmetry in favor of shorter positions. With this formulation, we can solve the preprocessing for a switch node i by setting $p_i = x_{i,N_1(i)} = 1$ and see whether the formulation is feasible or not. Feasibility of the formulation means that there is feasible position anchored at node i for a train of length L. We justify the necessity of the Constraint (3.20d) as the feasible position generated with the Formulation (3.20) can consist of two disjoint walks in which one of them is a cycle.

Algorithm 3 and Formulation (3.20) can solve PREPROCESS($\cdot$) in all cases of network G.

We now proceed by presentation the computational results.

3.4 Computational Results

This section presents computational results for the proposed combined preprocess-routing approach to solve the YRP. The computational experiment is performed on a real yard network dataset provided by a major rail company. This yard network has 4601 nodes, 4725 edges (0.04% arc density) and 287 switch nodes (6% switch node density). As the length of the smallest directly traversable cycle in this dataset is 2380 units, we solve the YRP only for values $L \leq 2380$ to ensure that we can solve the preprocessing problems with the LP_i formulation. All tests were conducted on an Intel Core i7 CPU @ 2.93 GHz, 8 GB RAM computer using IBM ILOG CPLEX Optimization Studio 12.6 to implement the modeling approach. To perform the computational experiments for this section, we generated 400 instances of the yard routing problem with different origin and destination. Out of these 400 instances, we have removed 125 of them that the train did not change its direction in the optimal route, even for the smallest L examined. The solution times are less than one second for all the instances as the modeling approach solves the problem in polynomial time. In this section, we show how the L parameter, objective function, and whether or not congestion caused by idle cars affect the optimal solution of the problems.

For the instances generated, four different lengths for the train (L) are examined considering that these lengths should all be less than the length of the shortest cycle in the yard network in order for us to use LP_i to solve PREPROCESS(i). We examined two objective functions for the yard routing problems, minimizing the total length of the route including the L units increase in the objective value for traversing the switch nodes indirectly (OF1) and minimizing the number times that the train traverse the switch nodes indirectly (OF2). We consider two environments for the yard network to perform the experiments: an empty yard network (NC) and a yard network with one percent congestion (C). The one-percent congestion is done randomly for each track segment in the railyard since we do not have a clear information on the probability distribution of congestion on the track segments. We do not increase the congestion percentage to ensure that

the instances do not become infeasible.

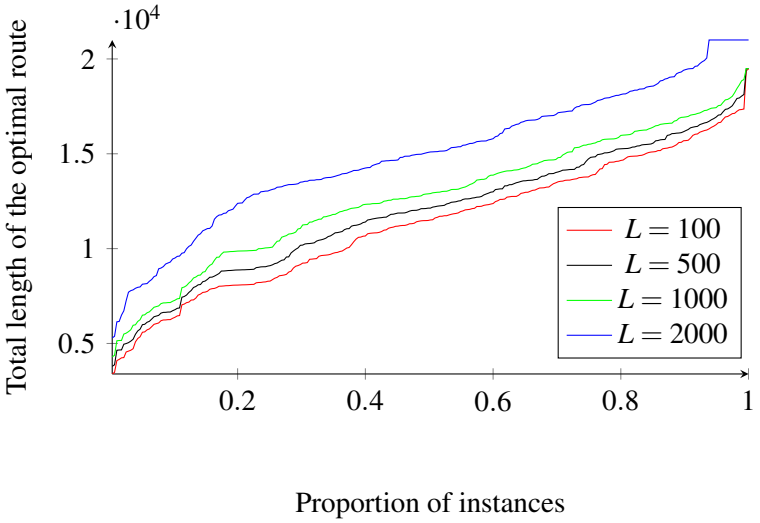
For the first set of experiments, we choose OF1 and NC as the objective function and yard environment, respectively. Figure 3.13(a) and 3.13(b) show the impact of the increase in L on the total and the pure length of the optimal route, respectively. Instances that hit the upper limit of the plot are infeasible. These two figures compare the cumulative proportion of instances within the y-axis value for different parameters of L. The value "pure length of the optimal route" does not include the L units in it for when the train reverses its direction. Those instances that the blue plot hit the highest value are infeasible. As can be seen in Figure 3.13(b) increasing the length of the train up to 1000 units does not change the optimal route. However, when L increases to 2000 units, the previous optimal route is no longer feasible. The reason is that there cannot be a feasible position for the train of length 2000 units at switch nodes where the train reverses its direction in the previous optimal route.

Figure 3.14 shows the number of times that the train uses an indirectly-traversable walk at switch nodes in the yard. Instances with -1 as the number of indirectly traversed switch nodes are infeasible. Even though on average this value decreases as the length of the train increases, there exist some instances where this value increases depending on the origin and destination which is illustrated in Figure 3.15. In Figure 3.15, the two plots at each point in the x-axis corresponds to the same problem. Those instances with negative y-axis are infeasible.

For the second set of experiments, we choose OF2 and NC as the objective function and the yard environment, respectively. Figure 3.16 and 3.17 compare the new results with the previous set. There is a significant trade-off between decreasing the number of indirectly traversed walks in the optimal routes and its total length. In practical situations where indirectly traversing the switch nodes is difficult or expensive, this shows the trade-off in changing the objective function.

For the last set of experiments, we choose OF1 and NC as the objective function and the yard environment, respectively. Figure 3.18 compares the new results with the first set of experiment. Those instances that the blue plot hit the highest value are infeasible. We can approxi-

(a)



(b)

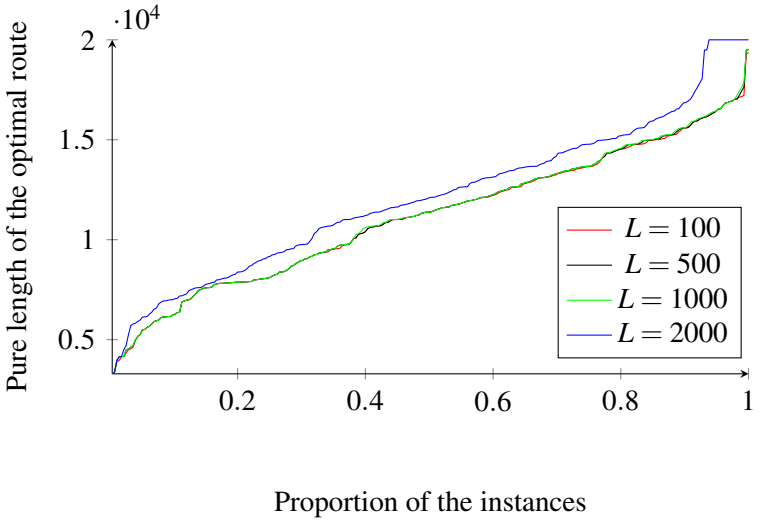


Figure 3.13: Impact of the parameter L on (a) the total length and (b) the pure length of the shortest route in the yard network with no congestion

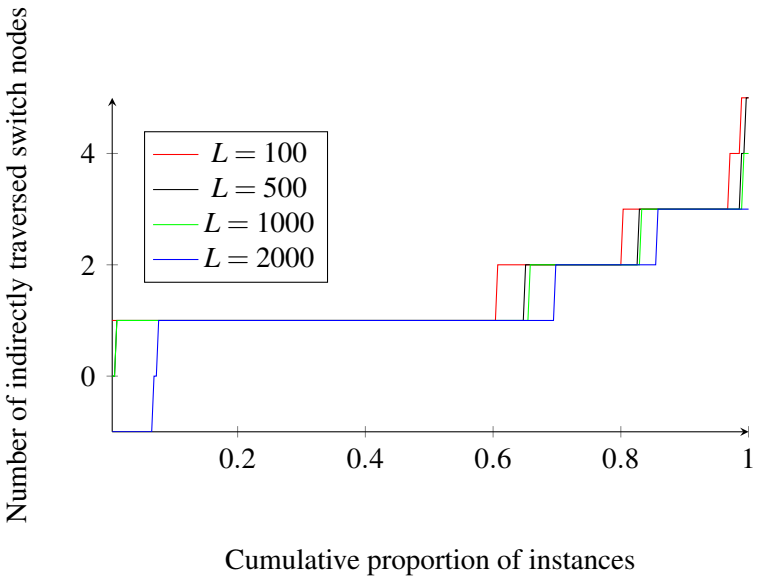


Figure 3.14: Impact of the parameter L on number of indirectly traversed switch nodes in the optimal route

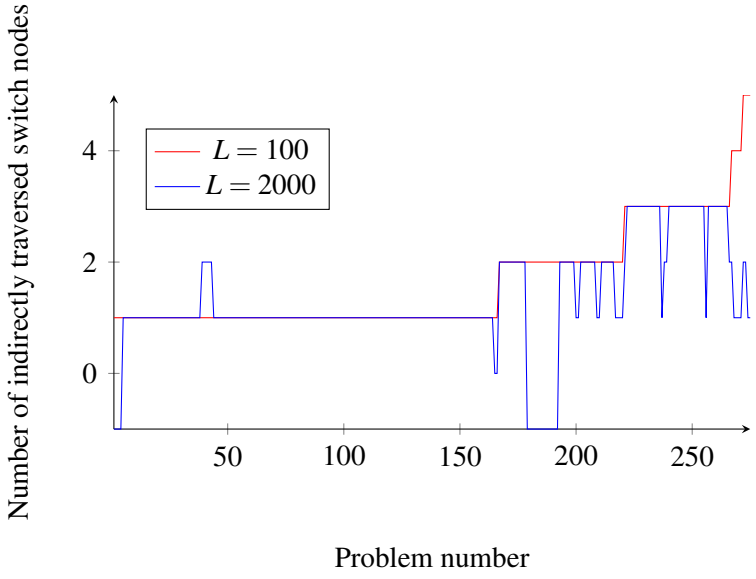


Figure 3.15: Impact of the parameter L on number of indirectly traversed switch nodes in the optimal route

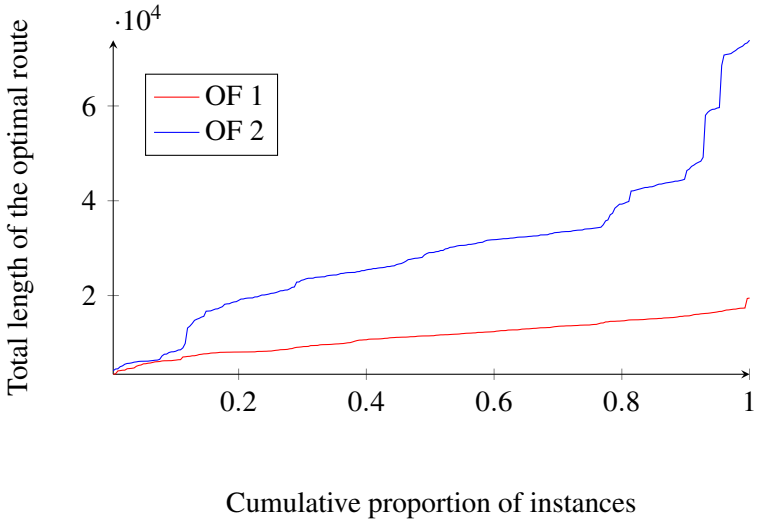


Figure 3.16: Impact of the objective function on the total length of the optimal route for a train with length 100 units

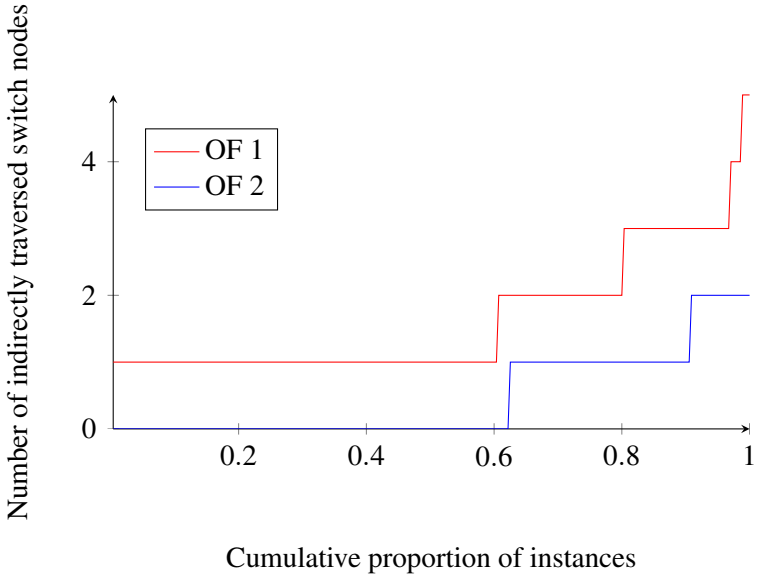


Figure 3.17: Impact of the objective function on number of indirectly traversed switch nodes in the optimal route for a train with length 100 units

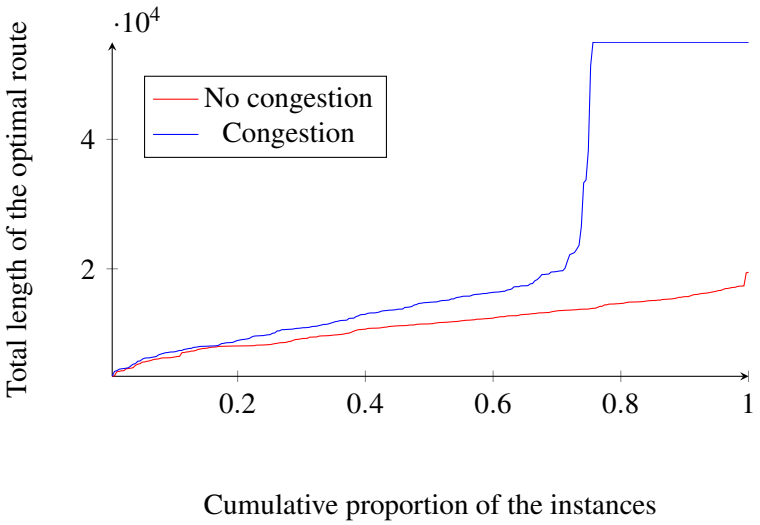


Figure 3.18: Impact of the congestion on the total length of the optimal route for a train with length 100 units

mately say that the congestion in the yard network does not change the optimal solution for 80 percent of the instances but it makes the other origin-destination pair infeasible.

3.5 Conclusion

We have presented an approach that models the single-train yard routing problem subject to the geometry of the yard network and the length of the train. The objective of the model is to minimize the total length of the route that the train takes to reach its destination. Our model is different from previous work as it considers the geometry of the track segments in the yard and that the train route must accommodate the total length of the train. The proposed approach models the problem in two stages, preprocessing and routing. We have shown that the preprocessing is NP-complete for the general case and we could develop an approach that solves an important special case in polynomial time. An algorithm and an integer programming formulation are proposed to solve the general case of the preprocessing.

In routing a single train in a railyard, the solution time is of importance as the problem must be solved repeatedly in a very short time for simulations of train movements in large complex yards. Our computational experiments point out the quick computational time needed for the

approach to identify the optimal solution for the special case that happens in real yard networks.

Bibliography

Akgün, I. and Tansel, B. Ç. (2007). Optimization of transportation requirements in the deployment of military units. *Computers & Operations Research*, 34(4):1158–1176.

Chardaire, P., McKeown, G. P., Verity-Harrison, S., and Richardson, S. (2005). Solving a time-space network formulation for the convoy movement problem. *Operations Research*, 53(2):219–230.

Cordeau, J.-F., Toth, P., and Vigo, D. (1998). A survey of optimization models for train routing and scheduling. *Transportation Science*, 32(4):380–404.

Fu, L. and Dessouky, M. (2018). Algorithms for a special class of state-dependent shortest path problems with an application to the train routing problem. *Journal of Scheduling*, 21(3):367–386.

Fügenschuh, A., Homfeld, H., and Schülldorf, H. (2013). Single-car routing in rail freight transport. *Transportation Science*, 49(1):130–148.

Garey, M. R. and Johnson, D. S. (1979). Computers and intractability: a guide to np-completeness.

Goldstein, D., Shehab, T., Casse, J., and Lin, H.-C. (2010). On the formulation and solution of the convoy routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 46(4):520–533.

Gopalan, R. (2015). Computational complexity of convoy movement planning problems. *Mathematical Methods of Operations Research*, 82(1):31–60.

Haghani, A. E. (1987). Rail freight transportation: a review of recent optimization models for train routing and empty car distribution. *Journal of Advanced Transportation*, 21(2):147–172.

Haghani, A. E. (1989). Formulation and solution of a combined train routing and makeup, and empty car distribution model. *Transportation Research Part B: Methodological*, 23(6):433–452.

Hernando, A., Roanes-Lozano, E., and Garcia-Álvarez, A. (2018). A recommender system for train routing: When concatenating two minimum length paths is not the minimum length path. *Applied Mathematics and Computation*, 319:486 – 498. Recent Advances in Computing.

Khachiyan, L. (1980). Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53 – 72.

Lin, B. (2017). Integrating car path optimization with train formation plan: a non-linear binary programming model and simulated annealing based heuristics. *arXiv preprint arXiv:1707.08326*.

Nagarajan, V. and Ranade, A. G. (2008). Exact train pathing. *Journal of Scheduling*, 11(4):279–297.

Riezebos, J. and Van Wezel, W. (2009). k-shortest routing of trains on shunting yards. *OR Spectrum*, 31(4):745.

Sama, M., Pellegrini, P., D'Ariano, A., Rodriguez, J., and Pacciarelli, D. (2016). Ant colony optimization for the real-time train routing selection problem. *Transportation Research Part B: Methodological*, 85:89–108.

Sama, M., Pellegrini, P., D'Ariano, A., Rodriguez, J., and Pacciarelli, D. (2017). On the tactical and operational train routing selection problem. *Transportation Research Part C: Emerging Technologies*, 76:1–15.

Sun, Y., Lang, M., and Wang, D. (2016). Bi-objective modelling for hazardous materials road–rail multimodal routing problem with railway schedule-based space–time constraints. *International Journal of Environmental Research and Public Health*, 13(8):762.

Yang, L., Gao, Z., Li, X., and Li, K. (2011). A weighted min–max model for balanced freight train routing problem with fuzzy information. *Engineering Optimization*, 43(12):1289–1309.

4. Two-train Yard Routing Problem

4.1 Introduction

With the increase in railway traffic and its growing demand, it is necessary to exploit the capacity of the current railway network as efficiently as possible since extending the rail tracks to enhance the railway infrastructure is very cost-intensive or may not be feasible in some areas. One of the difficult problems in doing so is planning a conflict-free movement of trains through a railway station. In the problem of routing trains in a yard network, there are two important types of decisions: the routing plan assigns trains to routes and the scheduling plan assigns blocking times to the track segments and nodes in each route used by a train. These two components are not independent of each other as conflicts might be avoided by making changes to both of them. In chapter 3, we reviewed the research related to the routing of a single train in a yard network. We now give a literature overview of research that allocates the track capacity of the yard network to trains in a conflict-free manner.

One of the most common optimization in routing multiple trains through a rail yard is to use conflict graph methodology. In this approach, nodes represent a train path and edges correspond to pairwise conflicts, and the routing problem can be explicitly modeled as a node packing problem. Zwaneveld et al. (1996) propose the first node packing model to route trains through railway stations, and this problem was later shown to be NP-hard by Kroon et al. (1997). The model has a timetable as input and determines whether or not this is feasible with respect to the safety rules and the connection requirements at the station. Despite assuming each train's candidate routes are enumerated a priori, the model remains difficult to solve even after adding clique inequalities within a branch-and-cut approach. Zwaneveld et al. (2001) and Caimi et al. (2005) respectively develop new exact and heuristic solution approaches for this problem.

The node packing approach (and all of the papers mentioned in the previous paragraph) for train routing requires the identification of all conflicting train movement prior to formulating the model which makes the approach inflexible in dynamic environments (Lusby et al., 2011b).

In order to address this weakness, Lusby et al. (2011a) examine the problem of maximizing the number of trains that can be routed conflict-free through a junction given the scheduled arrival and departure of trains. They propose a set packing model in which no pair of selected train routes may simultaneously occupy any track section. Although the set packing model does not require a priori identification of route-pair conflicts, it does require enumerating each train's candidate paths. Caimi et al. (2011) show that maximal conflict cliques can be identified efficiently for this problem and used to strengthen the formulation of the set packing model.

Further research has sought to identify route-to-train assignments, under a number of objectives, in which the candidate routes and their scheduled arrival and departure times are fixed a priori. Sun et al. (2014) develop a multi-objective train routing model on a one-way double track rail network that takes into account energy consumption, the user satisfaction, and the average travel time of all trains. Sels et al. (2014) study the train platforming problem with the objective of minimizing the penalty cost of moving assignments from real to fictive platform and of moving assignment from preferred to non-preferred (real) platforms.

There are routing models that are not constrained by the scheduled arrival and departure times for trains. Burggraeve and Vansteenwegen (2017) propose an approach in which the routing problem precedes the timetabling problem. The routing problem, unrestricted by potential impacts on the timetable, first assigns trains to routes in order to minimize maximal node usage. Given the each train's route, a mixed-integer linear programming model generates a "passenger robust" timetable that determines when each train occupies each of the segments on its route. Lu et al. (2004) study the train routing problem in densely populated metropolitan areas, an application that requires accounting for the different trackage configurations in the rail network and the trains' acceleration and deceleration rates. Borndörfer et al. (2016) investigate the problem of routing freight with defined origin and destination in a macroscopic transportation network. Their problem is to minimize the sum of all expected delays and all running times. They formulate the problem as a multi-commodity flow on a time-expanded graph which then reduced to a mixed-integer linear programming by piecewise linear approximation.

In addition to the disaggregated routing and timetabling research described above, there are integrated approaches that solve the two problems simultaneously. Li et al. (2013) formulate a compound model of train routing and scheduling. The objective of their model is to minimize the total delay of all trains in the railway network. A set of routes for each train from the origin to the destination are predefined and trains can only select one route in their feasible route sets. The constraints that they consider in their model are (1) safety constraints that ensure there must be a minimum headway for two trains passing the same railway section in the same direction; (2) conflict constraints for the scenarios where two trains of opposite direction occupy the same rail section at the same time. These constraints ensure a train must dwell on the station such that the other train can meet and cross; (3) capacity constraints that ensure the number of trains at the station does not exceed the capacity of stations at any time. They propose a route adjustment algorithm to obtain good route schemes based on the trains' delay information. Dewilde et al. (2013) works on the robustness of a complex station. Their proposed approach iterates between routing decisions, timetabling and platform assignments to increase robustness. The objective of their methodology is to maximize the minimal time span between any two trains in a station zone. In their routing module, a route is selected from a set of candidate routes.

Looking for a conflict-free schedule requires keeping track of all the trains at all times which is computationally difficult. In this chapter, the routing and scheduling problems are solved separately to decrease the overall computational difficulty. We indirectly deal with the time index in the conflict-free definition by accounting for it as a distance in the routing model. We develop a routing model that using its solution we can generate a scheduling plan with minimum overall time. In order to obtain a conflict-free schedule, the routing component is solved first which is not constrained by the scheduling components. Then the schedule of the conflict-free movements is generated once the routes are defined. We consider the geometry of the yard network when we choose a route for each train with respect to its length. We demonstrate that this approach, in generating a routing plan, has a better total travel time when compared to the case where we generate the train routes individually as a single-train yard routing problem in the yard network.

The main contributions of this chapter are: (i) our proposed yard routing model to generate a routing plan for two origin-destination pairs is the first to consider the geometry of the track segments in the railyard, in a way that scheduling the solution leads to the minimum possible total travel time; (ii) we develop a scheduling model that can generate a non-conflicting timetable given the solution obtained from the routing model; and (iii) we demonstrate, on a realistic railyard dataset, that this two-train routing model yields a schedule with less overall time when compared to the solution of two separate single-train routing models as presented in Chapter 3.

4.2 Problem Definition

We consider the problem of routing two trains of given lengths from their origins to their destination through a railyard defined by the undirected network $G = (N, E)$ with node set N and edge set E. The yard network has the same topological properties and the same restriction on the trains' movements as the single case of this problem, and we will describe the yard network using the notation of Chapter 3. The purpose of determining the routes for the two trains is to generate a conflict-free schedule that requires the least overall time. We indirectly estimate the overall time with the total equivalent distance the two trains travel (i.e., assuming trains move at a constant speed through the network). In addition to the length of the routes that the two trains must travel to reach their destination, the time to complete the routes may be increased if one train needs to wait on the other in order to avoid conflict. This can occur when the two trains traverse the same track segment in the opposite direction. Our proposed model seeks to minimize the estimated overall time or its equivalent distance. The resulting optimization problem is to identify a route for each locomotive to pull a cut of cars across a yard from one location to another with the objective of minimizing the total length of the two routes and their penalty distance, subject to the constraints of the geometry of the yard tracks. We refer to this optimization problem as the Two-train Yard Routing Problem (TYRP).

4.3 Problem Formulation

In this section, we propose a model that routes two trains in a railyard with the objective of avoiding direction-wise conflicts between the trains on yard track segments and then in Section 4.4 we develop a model that optimizes their scheduling assuming the routes are fixed. This model is built given the information obtained from solving PREPROCESS($\cdot$) for all the switch nodes (i.e., the value of y_i in the optimal solution to PREPROCESS(i) for all $i \in N^S$). In order to formulate this model, we first define the necessary sets, parameters and variables.

Sets:

- $G = (N, E)$: the undirected yard network (described in Section 3.2)
- $G' = (N, A)$: the directed version of the network G
- $G^k = (N', A^k)$: the expanded network of G (described in Section 3.3.2) for train $k \in \{1, 2\}$
- $S(i, j)$: the set of all arcs in the expanded network of G that originated from the arc $(i, j) \in A$
- $S^k(i)$: the set of all arcs in the expanded network of G for train $k \in \{1, 2\}$ that connects the nodes originated from the switch node $i \in N^{L^k}$ in layer push with those in layer pull
- T_i^k : the set of all the arcs in the network G' that creates a feasible position for train $k \in \{1, 2\}$ to traverse the switch node $i \in N^{L^k}$ indirectly

Note that G^1 may have different arc set from G^2 since the two trains can have different lengths that affects the set of indirectly traversable switch nodes. As described in Section 3.3.2, each direction of an edge $[i, j]$ in the network G (or each arc in the network G') is represented by two arcs (pull and push) in the expanded network of G which constructs the set $S(i, j)$. In the yard illustrated in Figure 4.1, the set T_5^k includes the arcs $(5, O^2)$, $(O^2, 5)$ if train k has a length of 2 units. The set T_i^k is constructed using the solution of PREPROCESS(i) and the length of train k.

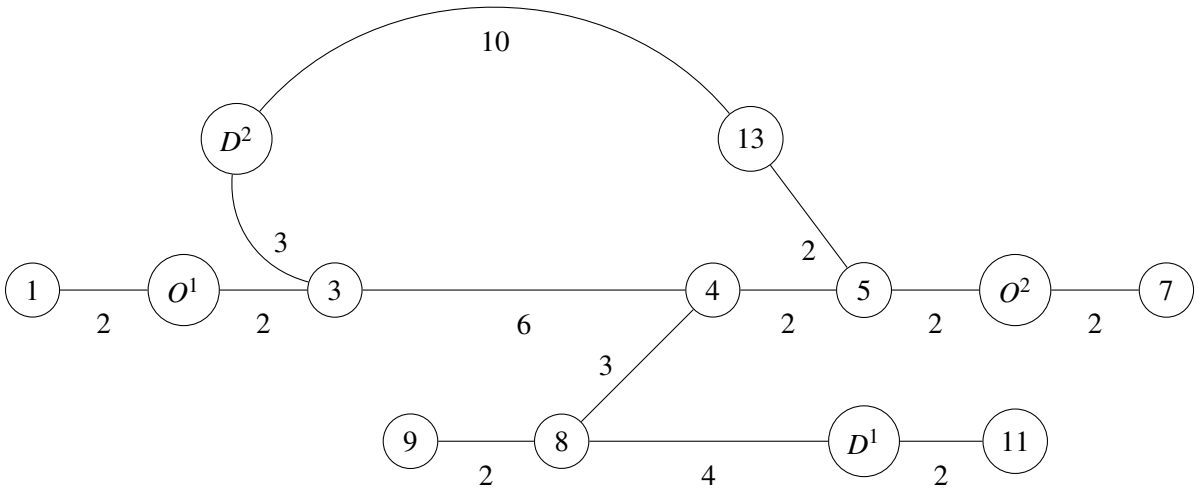


Figure 4.1: A two-train yard routing problem with the parameters L^1, L^2, s^1, s^2, and B set to 2 unit, 2 unit, 1 unit per hour, 1 unit per hour, and 5 hours, respectively. Edge lengths are defined as the arc labels. The yard network continues at node 1, 7, 9, and 11 which is not shown in this figure.

In the expanded network illustrated in Figure 3.9 of Section 3.3.2, the set $S(3)$ includes the blue arcs.

Parameters:

- $c_{(i,j)}$: the length of the directed arc $(i, j) \in A^k$ in the network G^k
- L^k : the length of train $k \in \{1,2\}$
- b_i^k : $\begin{cases} 1 & \text{if node } i \in N' \text{ is the origin for train } k \in \{1,2\} \text{ in the network } G^k \\ -1 & \text{if node } i \in N' \text{ is the destination for train } k \in \{1,2\} \text{ in the network } G^k \\ 0 & \text{otherwise} \end{cases}$

Variables:

- x_{ij}^k : whether (1) or not (0) the arc $(i, j) \in A^k$ is traversed by train $k \in \{1,2\}$ in G^k
- z_{ij}^k : whether (1) or not (0) at least one of the arcs in the set $S(i, j)$ for arc $(i, j) \in A$ is traversed by train $k \in \{1,2\}$ in network G^k
- y_{ij} : whether (1) or not (0) the length of arc $(i, j) \in A$ is counted as a penalty in the objective value

- t_i^k : whether (1) or not (0) train $k \in \{1,2\}$ traverses the switch node $i \in N^{L^k}$ indirectly in network G^k

The integer programming formulation is given by:

$$\min \sum_{k\in\{1,2\}} \sum_{(i,j)\in A^k} c_{ij} x_{ij}^k + \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{4.1a}$$

subject to

$$\sum_{j:(i,j)\in A^k} x_{ij}^k - \sum_{j:(j,i)\in A^k} x_{ji}^k = b_i^k \qquad \forall k \in \{1,2\}, \forall i \in N' \tag{4.1b}$$

$$\sum_{(i',j')\in S(i,j)} x_{i'j'}^k \leq z_{ij}^k \qquad \forall k \in \{1,2\}, \forall (i,j) \in A \tag{4.1c}$$

$$\sum_{(i',j')\in S(i)} x_{i'j'}^k \leq 4 t_i^k \qquad \forall k \in \{1,2\}, \forall i \in N^{L^k} \tag{4.1d}$$

$$z_{ij}^1 + z_{ji}^2 \leq 1 + y_{ij} \qquad \forall (i,j) \in A \tag{4.1e}$$

$$z_{jm}^{k'} + t_i^k \leq 1 + y_{jm} \qquad \forall (j,m) \in A \cap T_i^k, i \in N^{L^k}, \forall k,k' \in \{1,2\}; k \neq k' \tag{4.1f}$$

$$t_i^1 + t_i^2 \leq 1 + y_{jm} \qquad \forall (j,m) \in A \cap T_i^1 \cap T_i^2, \forall i \in N^{L^1} \cap N^{L^2} \tag{4.1g}$$

$$t_i^k \in \{0,1\} \qquad \forall k \in \{1,2\}, \forall i \in N^{L^k} \tag{4.1h}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in \{1,2\}, \forall (i,j) \in A^k \tag{4.1i}$$

$$z_{ij}^k \in \{0,1\} \qquad \forall k \in \{1,2\}, \forall (i,j) \in A \tag{4.1j}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{4.1k}$$

Objective (4.1a) seeks to minimize the total length of the routes for the two trains plus the penalty of the two routes. Constraint (4.1b) ensures the flow balance at each node in the expanded network of G for each train, effectively requiring that the net flow into node i for train k is equal to the flow balance value of that node. Constraint (4.1c) ensures that if train k traverses arc $(i,j) \in A$ in either the push or the pull layer in its expanded network G^k, then the train traverses that arc (i,j). If train k traverses between the two layers push and pull at switch node i, then train k traverses node i indirectly which is stated by Constraint (4.1d). Constraint (4.1e) ensures that a

track segment is traversed in the two routes and counted as a penalty only if both trains traverse that track segment in the opposite direction. If a train traverses a switch node i indirectly and the other train traverses the edge $[j,k]$ in the SDT(i) feasible position of the first train, we count the length of that edge as a penalty in the objective function using Constraint (4.1f). If the two trains traverse the same switch node indirectly, we double count the length of the edges in the feasible position of the shorter train as a penalty in the objective function using Constraint (4.1g). Constraint (4.1h)–(4.1k) define the binary restrictions on the decision variables. We now proceed to the next section in which we describe how to schedule the routes in a conflict-free manner.

4.4 Model Validation

In this section, we compare the routes obtained using the Model (4.1) with the routes obtained by solving the two origin-destination pairs individually as a single-train yard routing problem. In order to perform this comparison, we develop a scheduling formulation that minimizes the overall time for the two trains to reach their destination given the two trains' routes. With this scheduling model, we can compute the overall scheduling time for the two solutions obtained with the two model. The reason to choose the single-train routing model as our reference is that this model runs in polynomial time and consider the track segments configuration. We now provide the following notation necessary to represent the scheduling model.

Notation:

- R^k: the sequence of nodes $\{i^k(h)\}_{h=0}^{n(k)}$ visited by either the front or back of train $k \in \{1,2\}$ in its optimal route.
- $R^{(f,k)}$: the sequence of nodes $\{i^{(f,k)}(h)\}_{h=0}^{n(f,k)}$ visited by the front of train $k \in \{1,2\}$ in its optimal route.
- $R^{(b,k)}$: the sequence of nodes $\{i^{(b,k)}(h)\}_{h=0}^{n(b,k)}$ visited by the back of train $k \in \{1,2\}$ in its optimal route.

- $R^{(fb,k)}$: the sequence of nodes $\{i^{(fb,k)}(h)\}_{h=0}^{n(fb,k)}$ visited by both the front and back of train $k \in \{1,2\}$ in its optimal route.

Sets:

- $H(k) = \{0,\ldots,n(k)\}$ for $k \in \{1,2\}$
- $H(f,k) = \{0,\ldots,n(f,k)\}$ for $k \in \{1,2\}$
- $H(b,k) = \{0,\ldots,n(b,k)\}$ for $k \in \{1,2\}$
- $H(fb,k) = \{0,\ldots,n(fb,k)\}$ for $k \in \{1,2\}$

We use the yard network illustrated in Figure 4.1 to give an example of the above notation. In this figure, suppose we determined that train 1 uses the route O^1–3–4–8–D^1 to reach its destination. Assuming that train 1 starts from its origin by pulling the cars attached to it, the node sequences O^1–3–4–5–4–8–9–8–D^1, O^1–3–4–5–4–8–D^1, O^1–3–4–8–9–8–D^1, and O^1–3–4–8–D^1 denote R^1, $R^{(f,1)}$, $R^{(b,1)}$, and $R^{(fb,1)}$, respectively. Since the train traverses the node 4 indirectly, the front of it travels to the node 5 so its back can continue the walk from node 4 to node 8. The turns at node 4 and 8 imply that the front will visit some nodes that are not visited by the back, and vice versa. The sets $H(k)$, $H(f,k)$, $H(b,k)$, and $H(fb,k)$ respectively have 9, 7, 7, and 5 positions in them.

Parameters:

- M : the summation of the time the two take to traverse their own route without any waiting time
- B : a minimum required headway between the two trains passing the same location
- s^k : the speed of train $k \in \{1,2\}$.
- D_h^k : 1 and -1 for when the train $k \in \{1,2\}$ pulls and pushes the cars attached to it at position $h \in H(k)$, respectively

- $P^k(h)$: whether (f) or not (b) train $k \in \{1,2\}$ is pulling the cars attached to it at position $h \in H(k)$

The minimum required headway is the shortest distance or time between vehicles achievable by a rail system without a reduction in the speed of trains which is determined by the structure of the signaling system. In the yard network illustrated in Figure 4.1 that train 1 takes the route O^1–3–4–8–D^1, $D_0^1 = D_1^1 = D_2^1 = 1$ and $D_3^1 = D_4^1 = -1$ where there are 4 positions in $H(fb,1)$, while $P_0^1 = P_1^1 = P_2^1 = P_3^1 = P_7^1 = P_8^1 = f$ and $P_4^1 = P_5^1 = P_6^1 = b$ where there are 8 positions in $H(1)$. The parameters D_h^k and $P^k(h)$ are defined on the same domain but have different outputs. They respectively output 1 and f if train k pulls the cars attached to it at position h and -1 and b otherwise.

Functions:

- For $h \in H(k)$ such that $i^k(h) \in R^{(f,k)}$, $g^{(f,k)}(h)$ denotes the values $h' \in H(f,k)$ with $i^{(f,k)}(h') = i^k(h)$ for $k \in \{1,2\}$.
- For $h \in H(k)$ such that $i^k(h) \in R^{(b,k)}$, $g^{(b,k)}(h)$ denotes the value $h' \in H(b,k)$ with $i^{(b,k)}(h') = i^k(h)$ for $k \in \{1,2\}$.
- For $h \in H(fb,k)$, $z^{(f,k)}(h)$ denotes the value $h' \in H(f,k)$ with $i^{(f,k)}(h') = i^{(fb,k)}(h)$ for $k \in \{1,2\}$.
- For $h \in H(fb,k)$, $z^{(b,k)}(h)$ denotes the value $h' \in H(b,k)$ with $i^{(b,k)}(h') = i^{(fb,k)}(h)$ for $k \in \{1,2\}$.
- For $h \in H(fb,k)$, $z^{(k)}(h)$ denotes the value $h' \in H(k)$ with $i^{(k)}(h') = i^{(fb,k)}(h)$ for $k \in \{1,2\}$.
- $d^k(h,h')$ denote the distance between position $h \in i^k(h)$ and $h' \in i^k(h)$ in the route of train k (R^k).

Mapping functions $g^{(\cdot,k)}(\cdot)$ and $z^{(\cdot,k)}(\cdot)$ are used between the position indices in the pair of sets $\{H(k), H(\cdot,k)\}$ and $\{H(fb,k), H(\cdot,k)\}$, respectively.

Sets:

- $S^k(h) = \{h' \in H^k : h' \geq h, d^k(h,h') \leq L^k, D_h^k = D_{h'}^k\}, h \in H^k, k \in \{1,2\}$

The set $S^k(h)$ includes the positions within L^k distance units of the position h such that train k does not change its direction.

Variables:

- $t_h^{(f,k)}$: the time that the front of train $k \in \{1,2\}$ reaches position $h \in H(f,k)$
- $t_h^{(b,k)}$: the time that the back of train $k \in \{1,2\}$ reaches position $h \in H(b,k)$
- $y_{hh''}$: whether (1) or not (0) train 2 reaches position $h'' \in H(2)$ before train 1 reaches position $h \in H(1)$

Assuming trains do not traverse one switch node indirectly twice for the notational simplicity, the scheduling model is given by:

$$\text{minimize} \quad \max\{t^{(f,1)}_{n(1)}, t^{(f,2)}_{n(2)}\} \tag{4.2a}$$

subject to

$$t^{(f,k)}_{h+1} - t^{(f,k)}_{h} \geq c_{[i^{(f,k)}(h+1), i^{(f,k)}(h)]}/s^k \qquad \forall k \in \{1,2\}, \forall h \in H(f,k) \setminus n(f,k) \tag{4.2b}$$

$$t^{(b,k)}_{h+1} - t^{(b,k)}_{h} \geq c_{[i^{(b,k)}(h+1), i^{(b,k)}(h)]}/s^k \qquad \forall k \in \{1,2\}, \forall h \in H(b,k) \setminus n(b,k) \tag{4.2c}$$

$$t^{(b,k)}_{z^{(b,k)}(h')} - t^{(f,k)}_{z^{(f,k)}(h')} \geq D^k_{z^{(k)}(h')} L^k/s^k \qquad \forall k \in \{1,2\}, \forall h' \in H(fb,k) \tag{4.2d}$$

$$t^{(f,k)}_0 \geq 0 \qquad \forall k \in \{1,2\} \tag{4.2e}$$

$$t^{(f,2)}_{g^{(f,2)}(h'')} - t^{(P^1(h),1)}_{g^{(P^1(h),1)}(h')} \geq -My_{hh''} + T^1_{hh'}(1-y_{hh'}) \qquad \forall h \in H(1), h'' \in H(2), \; h' \in S^1(h); i^1(h) = i^2(h'') \tag{4.2f}$$

$$t^{(b,2)}_{g^{(b,2)}(h'')} - t^{(P^1(h),1)}_{g^{(P^1(h),1)}(h')} \geq -My_{hh''} + T^1_{hh'}(1-y_{hh'}) \qquad \forall h \in H(1), h'' \in H(2), \; h' \in S^1(h); i^1(h) = i^2(h'') \tag{4.2g}$$

$$t^{(f,1)}_{g^{(f,1)}(h)} - t^{(P^2(h''),2)}_{g^{(P^2(h''),2)}(h')} \geq -M(1-y_{hh''}) + T^2_{h''h'}y_{hh'} \qquad \forall h \in H(1), h'' \in H(2), \; h' \in S^2(h''); i^1(h) = i^2(h'') \tag{4.2h}$$

$$t^{(b,1)}_{g^{(b,1)}(h)} - t^{(P^2(h''),2)}_{g^{(P^2(h''),2)}(h')} \geq -M(1-y_{hh''}) + T^2_{h''h'}y_{hh'} \qquad \forall h \in H(1), h'' \in H(2), \; h' \in S^2(h''); i^1(h) = i^2(h'') \tag{4.2i}$$

$$t^{(f,k)}_h \geq 0 \qquad \forall k \in \{1,2\}, \forall h \in H(f,k) \tag{4.2j}$$

$$t^{(b,k)}_h \geq 0 \qquad \forall k \in \{1,2\}, \forall h \in H(b,k) \tag{4.2k}$$

$$y_{hh'} \in \{0,1\} \qquad \forall h \in H(1), h' \in H(2) \tag{4.2l}$$

where $T^k_{hh'}$ is defined as $B + (L^k - d^k(h,h'))/S^k$ if $i^k(h)$ is not part of a feasible position to traverse a switch node at position h'' indirectly and $B + (2L^k - d^k(h,h') - 2d^k(h,h''))/S^k$ otherwise. The value $T^k_{hh'}$ can be defined as the time a train must wait untill after the other train reach a certain neighbor (at poosition h' of the other train) of a node (at position h of train k) visited by both trains before train k passes that node (at position h of train k).

Objective (4.2a) seeks to minimize the overall time that the fronts of two trains take to

reach their destinations. Constraint (4.2b) and (4.2c) ensures the time between when the front and back of train k traverses two consecutive nodes in its route is at least the time that the train needs to travel the distance between those nodes, respectively. Constraint (4.2d) ensures that the time between when the front and back of train k traverses one specific node is at least the time that the train needs to travel its length. Constraint (4.2e) sets the initial time for the two trains, which is the time that their fronts are placed in their origins, to be zero. Constraints (4.2f)–(4.2i) simultaneously ensures that if the two trains pass the same node in their routes, one must wait until the other pass that node and reach to a certain position before entering that node. Constraints (4.2j)–(4.2l) define the binary and non-negativity restrictions on the decision variables.

We now explain each of these constraints for the same example shown in Figure 4.1. Let train 1 and train 2 take the routes O^1–3–4–8–D^1 and O^2–5–4–3–D^2 to reach their destination, respectively. One instance of the set of Constraints (4.2b) and (4.2c) for train 1 would be for the time its front visits node 4 to be at least 6 $(= 6/s^1)$ units greater than the time its front visits node 3. One instance of the set of constraints (4.2d) for train 1 would be for the time its front visits node 4 to be at least 2 $(= 2/s^1)$ units greater than the time its back visits node 4, assuming that the train moves from its origin by pulling the cars attached to it. One instance of the set of Constraints (4.2f)–(4.2i) is that if train 2 visit node 5 before train 1, then train 1 cannot visit this node until after B $(= B + (L^2 - d^2(\text{node } 5, \text{node } 4))/s^2 = B + (2-2)/1 = B)$ units of time passed when the front of train 2 reached node 4.

4.5 A Problem Instance

In this section, we present how the two models, two-train yard routing and single-train yard routing, solve the instance illustrated in Figure 4.1. We solve the instance using the Model (4.1) and also as two single-train yard routing problems. The solution are:
Model (4.1):

Optimal route for train 1: O^1–3–4–8–D^1 with total length 19$(= c_{(O^1,3)} + c_{(3,4)} + c_{(4,8)} + c(8, D^1) + 2L^1)$ units

Optimal route for train 2: O^2–5–13–D^2 with total length 14($= c_{(O^2,5)} + c_{(5,13)} + c(13,D^2)$) units

Two single-train yard routing problems:

Optimal route for train 1: O^1–3–4–8–D^1 with total length 19($= c_{(O^1,3)} + c_{(3,4)} + c_{(4,8)} + c(8,D^1) + 2L^1$) units

Optimal route for train 2: O^2–5–4–3–D^2 with total length 13($= c_{(O^2,5)} + c_{(5,4)} + c_{(4,3)} + c_{(3,D^2)}$) units

There are two possible routes with lengths 13 and 14 units for train 2 to reach its destination. The reason that Model (4.1) chooses the longest one over the shortest one is that the shortest route traverses certain edges in the opposite direction of the way the first train traverses them, which we referred to as penalty of the route in our two-train yard routing model. This penalty is large enough for our two-train yard routing model to choose the second best route (with total length 14 units) for train 2 that does not have any penalty cost. Now we schedule these two sets of solutions using Model (4.2). The total travel time of the two solutions are:

- (routes are obtained by Model (4.1))

Total travel time for train 1: 19 units of time

Total travel time for train 2: 14 units of time

Optimal objective value $= \max\{14, 19\} = 19$

- (routes are obtained by single-train yard routing model)

Total travel time for train 1: 19 units of time

Total travel time for train 2: 26 (13 (traveling time) plus 13 (waiting time for the front of train 1 to exit node 5)) units of time

Optimal objective value $= \max\{19, 26\} = 26$

This example represents a case where the two-train yard routing model outperform the single-train yard routing model in overall scheduling time because of the direction-wise conflicts between the two trains. We now proceed to the computational experiment performed on a realistic yard data set.

4.6 Computational Results

This section presents computational results to compare the performance of the proposed modeling approach with the single-train yard routing model proposed in Chapter 3 to solve the routing of two trains in a yard network. This comparison is based on the result of the scheduling model developed in Section 4.4 for the two solutions obtained by the two models. The computational experiment is performed on the same real yard network dataset used in Chapter 3. In this section, we show how the length parameter of the two trains and the location of two origins and destinations effect on the performance of the modeling approach.

To generate the instances, first we randomly choose two distinct circular areas with a radius of 500 units in the yard network such that the distance between the center of these two circular areas is 5000, 10000, and 20000 units. Let S1, S2, and S3 denote these three sets, respectively. We generate 100 instances from each set in a way that the origin of one train and the destination of the other are chosen randomly from one of the circular areas, and vice versa from the other circular area in the set. The reason behind this instance generation process is to have interesting instances where the two models give two different solutions. For each set, we run the instances with three different values of L, 500, 1000, and 2000 units. The value of the parameters s^1 and s^2 are both set to 10 units per unit of time. The value of the parameter B is set to double the time that the train can travel its length (i.e., $2L^k/s^k$).

We compare the optimal scheduling objective value ν^2 of the routings obtained by two-train routing approach to the optimal scheduling objective value ν^1 of the routings obtained by single-train routing approach. We use the relative difference in the scheduling objective values of the two routing approaches for comparison which is defined as $100(\nu^1 - \nu^2)/\nu^1$. Figure 4.2,

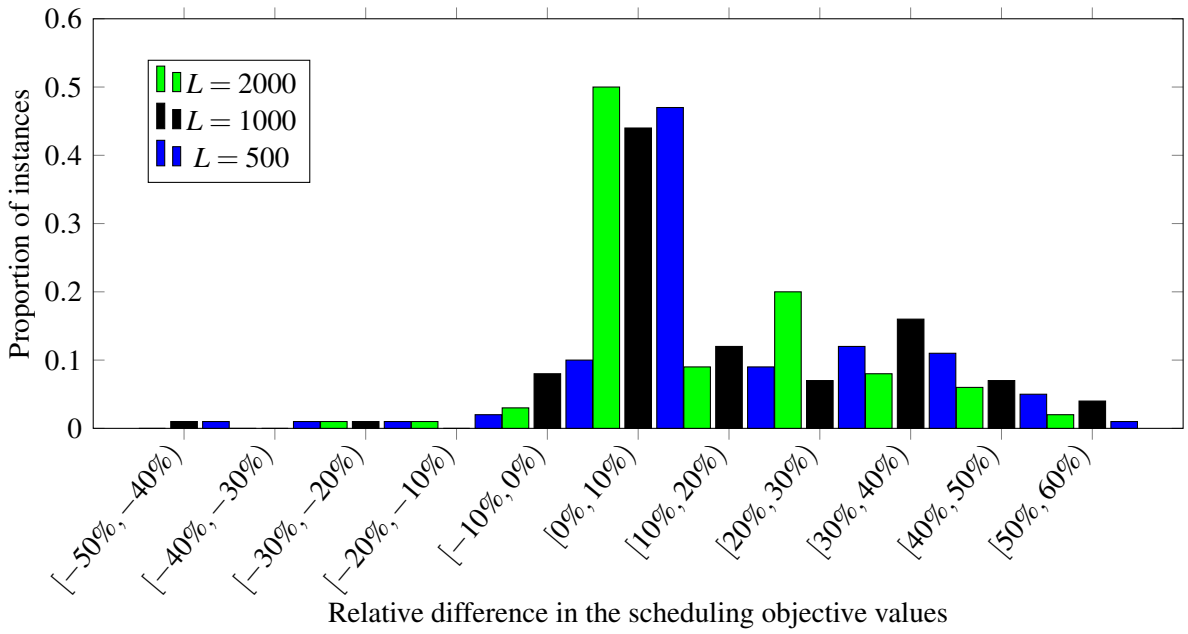


Figure 4.2: Problem set S1

4.3, and 4.4 show the histogram of the relative difference for the the sets S1, S2, and S3, respectively. In these three figures, the lengths of the two trains are the same. The figures indicate the overall advantage in scheduling time obtained by routing the two origin-destination pairs using the proposed routing model. This improvement becomes more significant when the distance between origin and destination of each pair increases. The proportion of the instances in which the two approaches have the same solution decreases from 0.5 to 0.4 as the distance between origin and destination of each pair increases, and this difference shows up in the larger percentage range of the x-axis. Figure 4.5 shows the histogram of the relative difference for the set S2 when the length of train 2 is fixed at 1000 units while the length of train 1 increases from 500 to 1500 units. The result shows no significant change in the performance of the proposed routing model when the lengths of the trains are not equal.

When the penalty for the routes obtained by the single-train routing model increases, the chance of the two approaches giving different solutions increases. In this case, there are two possibilities: (i) the penalty comes from the track segments that the two trains might not reach (and traverse in the opposite direction) at the same time which means the penalty was not effective and

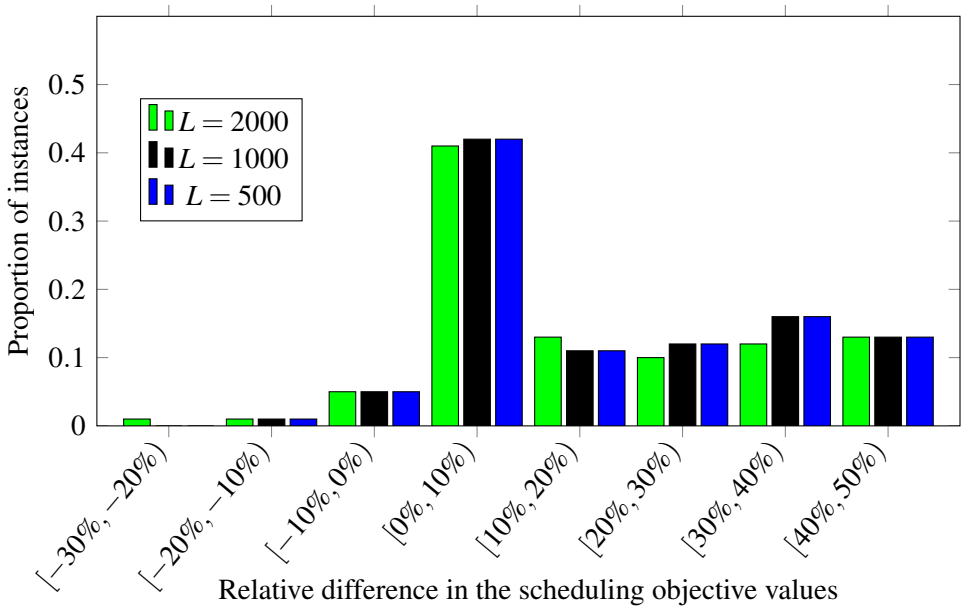


Figure 4.3: Problem set S2

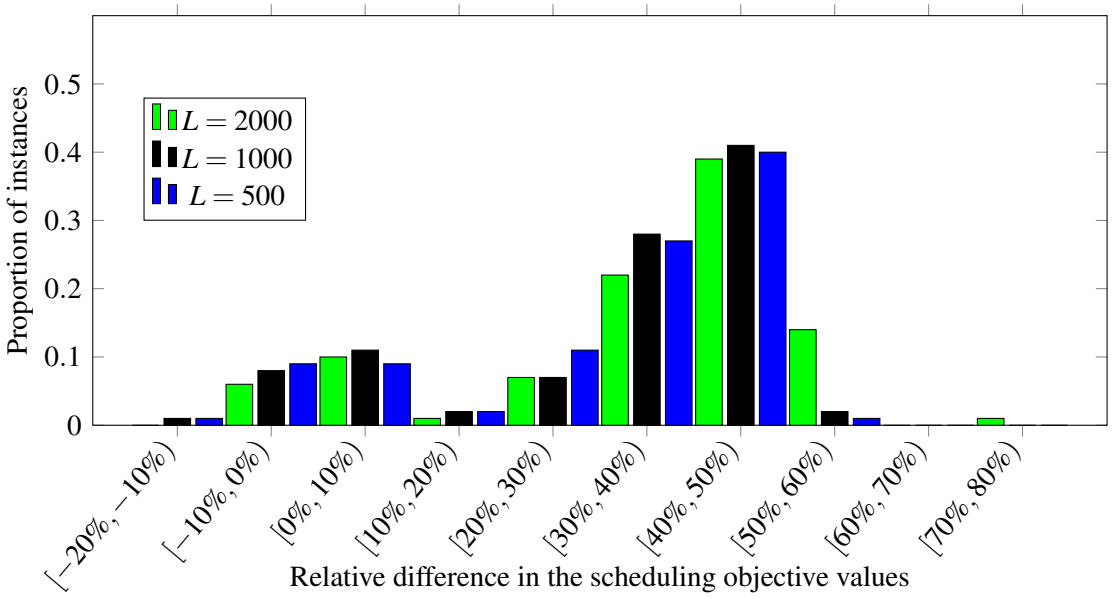


Figure 4.4: Problem set S3

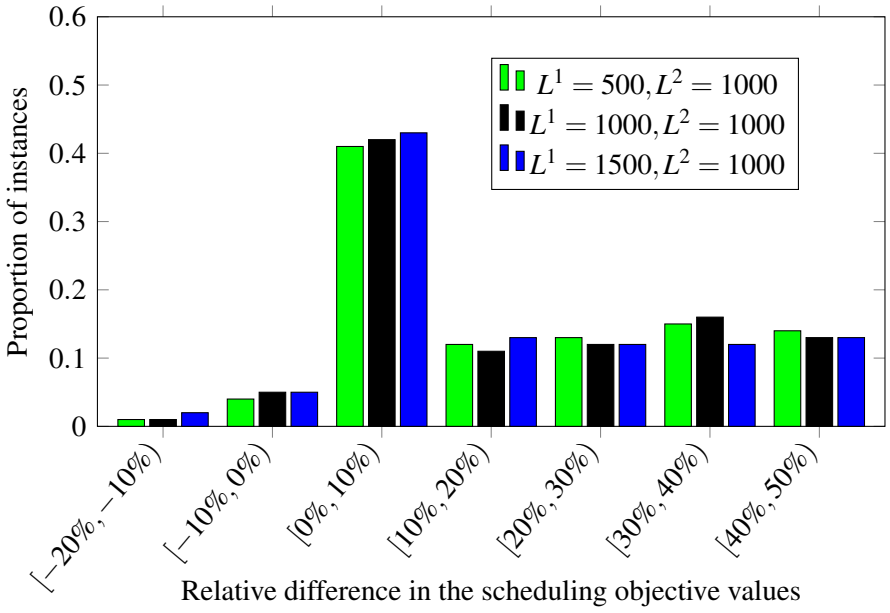


Figure 4.5: Problem set S2

the solution of the single-train model would probably have less overall scheduling time; (ii) the penalty comes from the track segments that the two trains might reach (and traverse in the opposite direction) at the same time which means the penalty was actually effective and the solution of the two-train model would probably have less overall scheduling time. Case two most probably happens when the origin of one pair is located close to the destination of the other pair and vice versa.

4.7 Conclusion

We investigated the routing of two trains in a yard network with the objective of indirectly decreasing the overall time that two trains take to reach their destination. The purpose of developing a routing model for two trains is to generate a solution that can be scheduled in less time in comparison to the solution of two individual single-train routing problems. The proposed two-train routing model in this chapter accounts for the waiting times that the two trains might have in their routes to their destinations. The model generates a routing plan by trading off between the length of the routes and the waiting time that the trains might have. The model interprets the

waiting time of two trains as the total length of the track segments that are used by trains in the opposite direction. Therefore, the objective function of the proposed routing model is to minimize the total length of the two trains' routes and the length of the track segment traversed by the trains in the opposite direction. The proposed two-train routing model generates the same routes as the single-train routing model if the routes do not intersect any edges in the opposite direction. We develop a model that schedules the output of the routing model with the objective of minimizing the overall time for the two trains to reach their destination. Using this scheduling model, the computational experiments presents a comparison of the quality of the solutions obtained by the proposed routing model and the single-yard routing model presented in Chapter 3. Results indicate the overall advantage of the proposed model in routing origin-destination pairs that would travel a set of yard track segments in the opposite direction in the same interval of time. This case has application in the station area of the yard network when there are certain entering and exiting points and the trains can pass the station area in the opposite direction.

Bibliography

Borndörfer, R., Klug, T., Schlechte, T., Fügenschuh, A., Schang, T., and Schülldorf, H. (2016). The freight train routing problem for congested railway networks with mixed traffic. *Transportation Science*, 50(2):408–423.

Burggraeve, S. and Vansteenwegen, P. (2017). Robust routing and timetabling in complex railway stations. *Transportation Research Part B: Methodological*, 101:228–244.

Caimi, G., Burkolter, D., and Herrmann, T. (2005). Finding delay-tolerant train routings through stations. In *Operations Research Proceedings 2004*, pages 136–143. Springer.

Caimi, G., Chudak, F., Fuchsberger, M., Laumanns, M., and Zenklusen, R. (2011). A new resource-constrained multicommodity flow model for conflict-free train routing and scheduling. *Transportation Science*, 45(2):212–227.

Dewilde, T., Sels, P., Cattrysse, D., and Vansteenwegen, P. (2013). Robust railway station planning: An interaction between routing, timetabling and platforming. *Journal of Rail Transport Planning & Management*, 3(3):68–77.

Kroon, L. G., Romeijn, H. E., and Zwaneveld, P. J. (1997). Routing trains through railway stations: complexity issues. *European Journal of Operational Research*, 98(3):485–498.

Li, F., Gao, Z., Li, K., and Wang, D. Z. W. (2013). Train routing model and algorithm combined with train scheduling. *Journal of Transportation Engineering*, 139(1):81–91.

Lu, Q., Dessouky, M., and Leachman, R. C. (2004). Modeling train movements through complex rail networks. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 14(1):48–75.

Lusby, R., Larsen, J., Ryan, D., and Ehrgott, M. (2011a). Routing trains through railway junctions: a new set-packing approach. *Transportation Science*, 45(2):228–245.

Lusby, R. M., Larsen, J., Ehrgott, M., and Ryan, D. (2011b). Railway track allocation: models and methods. *OR Spectrum*, 33(4):843–883.

Sels, P., Vansteenwegen, P., Dewilde, T., Cattrysse, D., Waquet, B., and Joubert, A. (2014). The train platforming problem: The infrastructure management company perspective. *Transportation Research Part B: Methodological*, 61:55–72.

Sun, Y., Cao, C., and Wu, C. (2014). Multi-objective optimization of train routing problem combined with train scheduling on a high-speed railway network. *Transportation Research Part C: Emerging Technologies*, 44:1–20.

Zwaneveld, P. J., Kroon, L. G., Romeijn, H. E., Salomon, M., Dauzere-Peres, S., Van Hoesel, S. P., and Ambergen, H. W. (1996). Routing trains through railway stations: Model formulation and algorithms. *Transportation Science*, 30(3):181–194.

Zwaneveld, P. J., Kroon, L. G., and Van Hoesel, S. P. (2001). Routing trains through a railway station based on a node packing model. *European Journal of Operational Research*, 128(1):14–33.

5. Conclusion and Future Work

In this dissertation, three different network-based optimization problems were studied that contribute to the area of critical interdependent infrastructure systems and rail freight transportation systems.

Chapter 2 presented a model that optimizes the operation of interdependent infrastructure systems. The model has the capability to capture many common interdependence classifications and is able to incorporate the possibility of dependency between components within systems. The model has a compact formulation and allows for non-uniform dependency requirements for systems' component. Computational experiments demonstrated that failing to model nonuniform dependency requirements may drastically reduce the quality of strategic restoration decisions derived to restore a disrupted collection of infrastructure systems. We identified two sets of valid inequalities which were utilized to develop a solution approach that exploits the effect of valid inequalities on the LP relaxation solution to expedite finding feasible solutions to the problem. This chapter contributes a thorough analysis of the operations of interdependent networks in resource-constrained environments. One of the directions to which this work can be extended is to incorporate the proposed operation model in restoration problems to enable better performance.

Chapter 3 presented a model to route a single train in a railyard subject to the geometry of track segments and the fact that train occupies space in the railyard. The model finds a feasible route for an origin-destination pair that minimizes the total length traveled by train to reach its destination. The proposed approach solves the defined yard routing problem in two stages, preprocessing and routing. We presented complexity results for the preprocessing stage and developed a linear programming formulation that can solve the preprocessing in polynomial time for an important special case that happens in practice. As the solution time to solve the problem is of importance in railyard transportation system, our computational experiments pointed out the quick computational time needed for the approach to identify the optimal solution for the special case that happens in real yard networks.

The problem defined in Chapter 3 was extended to routing two trains in Chapter 4. The

objective of the two-train routing model is to indirectly minimize the overall scheduling time of the two routes chosen for the two train. We developed a scheduling model that takes as input the solution of the two-train routing model and minimizes the total time needed for the two trains to reach their destination. This scheduling model helped us to compare the quality of the solution obtained by the two-train routing model and the solution obtained by solving the routing problem as two separate single train yard routing problems. The computational experiments indicate a better performance obtained by the proposed routing model when the routes use track segments in the opposite direction in certain cases. One of the directions to which this work can be extended is to incorporate acceleration and deceleration of train in calculating the overall scheduling time as in practice trains do not have constant speed throughout the routes they traverse. The other possible direction is to extend the problem to generate a routing plan for multiple trains with the objective of minimizing the overall scheduling time.