The Dissertation Committee for Yutian Yang Certifies that this is the approved version of the following dissertation:

Solving Transportation Scheduling Problems: Models and Algorithms

Committee:

Jonathan F. Bard, Supervisor

Anantaram Balakrishnan, Co-Supervisor

Ned Dimitrov

John J. Hasenbein

Leon Lasdon

Solving Transportation Scheduling Problems: Models and Algorithms

by

Yutian Yang, B.E.; M.S.E.

Dissertation

Presented to the Faculty of the Graduate School
of the University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

The University of Texas at Austin
May 2016

Dedication

To my dear wife.

Acknowledgements

First and foremost, I would like to thank my supervisors, Dr. Anant Balakrishnan and Dr. Jonathan Bard, for the unwavering support and mentorship throughout the past five years. Discussion with Dr. Balakrishnan not only guides my research but also enhances my professionalism and critical thinking. I also must thank Dr. Bard for his commitment and dedication to helping and advising me. I am thankful to Dr. John Hasenbein, Dr. Ned Dimitrov, and Dr. Leon Lasdon for taking the time to serve on my committee and making comments on this work. I would like to thank Brian Roth for being my mentor and providing many valuable suggestions on my research and personal growth.

I owe my deepest gratitude to my parents for all their love and support and putting me through the best education possible. I wouldn't have been able to get this stage without them. Support from my wife, Ying Chen, can never been overstated. I am grateful for her unending encouragement and love. I wouldn't have gotten through this doctorate if it wasn't for her.

Solving Transportation Scheduling Problems: Models and Algorithms

Yutian Yang, Ph.D.
The University of Texas at Austin, 2016

Supervisor: Jonathan F. Bard
Co-Supervisor: Anantaram Balakrishnan

Given the increasing load on current logistics and transportation systems, it is crucial to improve resource utilization and reduce operational costs. This can be achieved by developing better models and algorithms for transportation planning and scheduling. The main challenges include the mathematical modeling of operational rules, uncertainties in operations, and large-scale problem size. This dissertation addresses crew scheduling in freight railways and vehicle routing problems (VRP) for mail processing and distribution centers (P&DCs). Our goal is to develop models and algorithms that improve efficiency and reduce operating costs. In Chapter 2, we propose an optimization model to support real-time freight railway crew assignment decisions. Due to workload balance requirements and operating regulations, the optimization model is difficult to solve for realistic instances. Hence, we propose model improvements and develop effective solution techniques to find optimal or near-optimal solutions very quickly. Chapter 3 extends the freight rail crew scheduling problem by incorporating uncertainty in train arrival and departure times. We propose a stochastic programming model, but this model is solvable only the number of scenarios is small. As a consequence, we develop heuristics that use an analytical model to calculate the expected total cost of a given choice of crew deadheads. Using this cost evaluator, we develop four local search based heuristic algorithms to sequentially improve crew scheduling decisions under uncertainty. In Chapter 4, we first cluster the pickup and drop off points in mail P&DCs

into zones and then minimize the number of vehicles required and the total distance traveled to meet daily transport demand. The clustering is performed with a greedy randomized adaptive search procedure, and two heuristics are developed to find solutions to the VRP, which proved intractable for realistic instances. The heuristics are optimization-based within a rolling horizon framework. An extensive analysis is undertaken to evaluate the relative performance of the two heuristics. The contributions of this dissertation include modeling, algorithmic development, computational testing, and validation using real and randomly generated data.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1. Background

As home to the largest consumer market in the world, the United States has an enormous logistics and transportation system. In the U.S., spending on logistics and transportation services totaled $1.45 trillion in 2014, representing 8.3% of annual gross domestic product. However, many of the planning and scheduling problems encountered in the logistics and transportation industry continue to be solved manually. This dissertation addresses the freight railway crew scheduling problem and the vehicle routing problem (VRP) in mail processing and distribution centers (P&DCs). Our goal is to develop decision tools that improve efficiency and save operating costs for large-scale planning and scheduling in transportation systems.

America's freight railroads serve nearly every industry and resource-based sector of the economy. According to the National Rail Plan Progress Report (Federal Railroad Administration 2010), the U.S. freight rail network accounts for approximately 40% of U.S. freight movement in ton-miles (one ton of freight carried one mile). According to AAR's freight rail overview, between 2010 and 2035, the system is expected to experience a 22% increase in the total amount of tonnage. In 2013, the amount spent on building and maintaining American freight railroads was $64.1 billion (AAR, 2013). Apart from the cost of maintaining, expanding, and modernizing the railroad network, the cost of running the network accounted for 66% of the total spending. More than $10 billion was spent on wages and employee benefits, in addition to the cost of services such as hotel stays for train crews when they are away from home. Given the rapidly increasing demand for freight railroad services and the high operating cost, improving the efficiency of railroad operations is very important.

Although freight crew assignments problems are large scale and complex, solving them requires frequent and quick decision making. Unlike airline crew scheduling, where pilot schedules are planned months in advance and solution times of hours are acceptable, the planning horizon in freight railway crew assignments is days and the

problem must be solve in minutes. In a busy district, a two-day planning horizon involves more than 200 trains and 50 crews. Moreover, crew dispatchers need to decide whether and when to *deadhead* crew members from one station to the other with additional costs. For example, a taxi trip with up to five crews can be added at any time during the planning horizon. Although deadheading crews does not fill any crew requirement of scheduled trains, it can relieve crew shortage, reduce waiting time and cost, and so on.

Owing to the lack of systematic solution techniques, railroad companies mostly rely on the knowledge, experience, and judgment of experts when solving freight railway crew scheduling problems. The solution process may not only take a long time but also yield suboptimal solutions. For example, it is very difficult for crew dispatchers to consistently balance cost and operational requirements/preferences. An automated solution approach can result significant cost savings and operational benefits in the long term. Our goal is to develop optimization tools that can assist crew dispatchers with developing crew assignment solutions in real time.

In Chapter 2, we focus on railway crew assignment problems with complex operating rules, assuming that trains operate as scheduled. In Chapter 3, we focus on uncertainties in train schedules for crew assignment problems in single-ended districts. Dealing with operational uncertainty is one of the biggest challenges facing railroad companies. For example, trains may arrive at a terminal later than the scheduled time or may depart a terminal later than the scheduled time. Schafer and Barkan (2008) proposed a model to estimate train delay cost per hour, including car cost, locomotive cost, fuel cost, and crew labor cost. They concluded that delay costs alone can be as high as $200 to $300 per hour per train, and shipment or passenger delay costs can increase the actual costs considerably. Moreover, they pointed out that delaying a train on a track line may delay future trains due to track outage. This cascading effect leads to additional train delay cost. Instead of aiming at minimizing deviations from the scheduled timetable, we study how to model uncertainties and minimize expected crew assignment cost by considering schedule uncertainties.

United States Postal Service (USPS) is another example of a large transportation network in America. As of February 2015, USPS employed 617,254 workers and operated 31,000 post offices and locations in the U.S. It delivered 155 billion pieces of mail annually as of 2014. In 2006, USPS operated 673 mail processing and distribution centers (P&DCs). Owing to the widespread use of email and competition from FedEx and UPS, the mail volume at USPS has been declining since 2001. In response, USPS has enhanced productivity through increased automation, route re-optimization, and facility consolidation. In 2011, USPS announced the Network Rationalization Initiative plan for consolidating P&DCs and reducing mail processing infrastructure costs. This consolidation had the effect of increasing what was and still is a staggering amount of mail passing through these facilities. To handle such a large volume, highly sophisticated optical character readers, barcode sorters, and other advanced equipment have been designed to automate as much of the mail stream as possible. Material handling, too, plays a key role. Bulk parcels of mail trays need to be transferred from and to the docks and between various work centers. P&DCs operate 24 hours a day, 7 days a week. During a busy weekday, there may be over 4000 transfer requests in a typical P&DC. To meet these requests, most facilities rely on forklifts and tugs, which are collectively called *powered industrial vehicles* (PIVs).

In Chapter 4 of this dissertation, we study this pickup and delivery problem within mail P&DCs. Our goal is to develop a methodology for determining the minimal number of PIVs needed to transfer mail among P&DC workstations.

1.2. Objectives and Contributions

The general freight railway crew scheduling problem involves assigning crews to trains so that crew requirements on scheduled trains are met during a planning horizon. In current crew management practice, freight railways divide their railway networks into districts, each of which consists of a section of railway track and terminals in both ends. A typical train passes through multiple districts from its origin terminal to its destination terminal, and its crew usually boards the train at one ending terminal and gets off the

train at the other ending terminal so that the crew members travel within the district to which they belong.

At the operational level of railway crew scheduling, crew dispatchers are given the time schedule of trains in a planning horizon and the current status of crews and trains in the district. The dispatchers assign the required number of crew members of each occupation (engineer or conductor) to every scheduled train. In most cases, a train needs one engineer to operate the locomotive(s) and one conductor to drive the train based on signals. Moreover, crew dispatchers need to decide whether and when to deadhead crew members from one station to the other. There are three modes of deadheading trips: taxi, train deadhead (i.e., empty seats in a freight train), and passenger train (e.g., Amtrak). While deadheading crews results additional costs, it can relieve crew shortage, reduce waiting time and cost, and so on. In Chapters 2 and 3, our goal is to develop optimization tools that can support crew assignment decisions in real time.

In Chapter 2, we address a freight railway crew scheduling problem in double-ended districts, where both terminals have crew bases. In these districts, the workloads assigned to the two crew bases should be balanced within a pre-specified range. The main decisions include assigning duties/workloads to crew bases and determining deadhead trips while satisfying workload balance requirements and operating regulations. Assuming trains operate as scheduled, our goal is to find the minimum-cost crew assignment for a given planning horizon. We propose an optimization model and then improve it by providing an alternative formulation that requires considerably fewer variables and constraints. Moreover, we strengthen the model using valid inequalities. To support real-time crew scheduling decisions, we develop effective solution techniques to find feasible solutions in seconds and then solve the optimization model with warm starts (i.e., initializing CPLEX with integer-feasible solutions before branch-and-bound begins). We test our algorithms using real-life data from a major North American railway company. Compared with the original model, the model reduction reduces 95% of constraints in the formulation. Furthermore, the valid inequalities strengthen the lower

bound by 0.47% on average. After warm starts, all instances are solved to optimality within 200 seconds.

In Chapter 3, we discuss the freight railway crew scheduling problem considering uncertain train arrival/departure times. We focus on single-ended districts, where crews are based at only one ending terminal and the only scheduling decision is to select a set of deadhead trips. Robust optimization and stochastic programming are the most common approaches in related literature and practice. Robust optimization involves building slack into schedule solutions to survive an expected level of uncertainty in operations and then to hoping the solution can be executed in spite of the occurrence of unforeseen events. Stochastic programming models approximate a real problem based on a set of proposed scenarios. As the number of proposed scenarios increases, the approximation error decreases. However, when the uncertainty is high dimensional, the required number of scenarios significantly increases model size and complexity. We first propose a stochastic programming model and find that the model is solvable only when it includes a very small number of scenarios and approximates the real problem poorly. We developed an analytical model that calculates exactly the expected cost for a given set of train and deadhead trips. Instead of sampling, the analytical model is based on assumed distributions of train arrival/departure times. Then, we use four local search-based heuristic algorithms, greedy local search, Tabu search, simulated annealing, and genetic algorithms, to search for the optimal solution. The analytical model calculates the expected total cost of solutions in the heuristic algorithms. The initial solution in heuristics is the optimal solution for the stochastic programming model with few scenarios.

In Chapter 4, we discuss a variant of the classical VRP: a pickup and delivery problem with time windows (PDPTW). The problem arises from operations in mail processing centers, where mails are requested to be transferred between docks, operations, and work centers within specific time windows. Unlike classical pickup and delivery problems, our case requires grouping physical locations into zones and assigning vehicles to each zone before they are routed. The vehicles can cross zones by following

operational rules if necessary. Owing to large numbers of requests and a limited number of vehicles, routing vehicles efficiently is substantial for smooth operation of mail processing centers. We first cluster mail pickup and drop off points into zones using the algorithm developed by Deng and Bard (2011). Then, we determine the minimum number of PIVs required to meet all the demand for moving mail while minimizing the total vehicle travel times. That is, we solve a VRP with time windows (VRPTW) subject to cross-zone movement restrictions. We first try the column generation algorithm to solve the problem. For the smaller instances with 30 and 60 requests, the CG algorithm can converge and provide good lower bound within 4–13 hours of CPU time. However, when problem size increases to up to 120 requests, CG cannot solve the subproblems or even find a feasible solution with a negative objective function value within the given time limit (300 s). To solve the joint capacity planning and routing problems for the busiest shift in a week, which contains 1450 transfer requests, we propose two optimization-based, rolling horizon heuristics. Finally, we present an extensive analysis of the Chicago P&DC to demonstrate our computational scheme and the results of our parametric study.

We organize the rest of this dissertation as follows. Chapter 2 discusses the deterministic crew scheduling problem for freight railroads. We extend the analysis to a stochastic environment in Chapter 3. In Chapter 4, we address the pickup and delivery problem within mail P&DCs.

Chapter 2: Balanced Crew Scheduling in Freight Railways

2.1. Introduction

The general freight railway crew scheduling problem involves assigning crews to trips in a district so that crew requirements on scheduled trains are met during a planning horizon. In this chapter, we study railway crew scheduling in double-ended districts with workload balance requirements in a major North American railway company.

At the operational level of railway crew scheduling, crew planners assign the required number of crews from each occupation (engineer, conductor, and, possibly, brakeman) to every scheduled train. In *double-ended* districts, each ending terminal has home crew pools and away-from-home (away) crew pools for every occupation. For each outbound trip, crew planners must assign the required crew from either the home crew pool or the away crew pool. After choosing a crew pool, crew planners need to activate (or call on duty) crews from the chosen crew pool in the order governed by the crew activation rule (called *Rotation Key* rule). Moreover, crew planners need to decide whether and when to *deadhead* crew members from one station to the other with deadhead cost. Although deadheading crews does not fulfil the crew requirements of scheduled trains, it can relieve crew shortage, reduce waiting time and cost (*layover cost*) at the away crew pools, and balance workload across crew pools.

The workload balance requirements consist of the *Maximum Calling Ratio* (MCR) constraint and the *Workload Imbalance Control* (WIC) constraint. Both aim to allocate crew workloads to crew pools without overusing one crew pool. While the WIC constraints control the workload balance periodically during the planning horizon, the MCR constraints are applied to consecutive outbound trips. We study two crew Rotation Key (RK) rules: First-in-First-out (FIFO) and First-out-First-out (FOFO). Our objective is to minimize the total cost, which includes costs at away terminals, deadheading fixed cost, and trip cost. It is challenging to formulate an appropriate optimization model and solve it optimally (or with performance guarantees) and quickly.

The main contributions of this chapter are summarized as follows:

- We propose an optimization model for the freight railway CSP in double-ended districts with workload balance constraints. We reduce the model size by proposing an alternative formulation that requires considerably fewer variables and constraints. We strengthen the model by adding valid inequalities.
- To effectively and quickly solve the problem, we develop a heuristic algorithm that solves our optimization model without RK constraints and then repairs RK violations. Moreover, we use the heuristic solution to warm start the optimization model.
- We apply our algorithms to real-life data from a major North American railway company. Our approach generates high-quality heuristic solutions in seconds and optimally solves most instances within 3 minutes. Computational experiments show that our optimization model remains effective when critical input features vary.

This chapter is organized as follows. Section 2.2 presents a brief review of the literature on the railway CSP. We describe our problem in Section 2.3. Section 2.4 provides a basic optimization model, and Section 2.5 summarizes model improvements. We present our solution techniques in Section 2.6. The computational results and our conclusions are given in Sections 2.7 and 2.8, respectively.

2.2. Literature review

Crew scheduling problems have been studied extensively in operations research literature. Arabeyre et al. (1969) surveyed airline crew scheduling in early years, and Gopalakrishnan and Johnson (2005) summarized different approaches for airline CSPs in recent times. Railway crew scheduling problems in passenger and freight railway are different. For instance, while the CSP in freight railway assigns duties to crew members in real time, the CSP in passenger railway aims to assign periodic timetables to crew members. Caprara et al. (2007) overviewed the main optimization problems in passenger railway scheduling, including creating routes and frequencies of trains and constructing crew rosters. Ahuja et al. (2005) reviewed freight railway scheduling problems. In

general, mathematical models for CSPs can be categorized into set-covering/partitioning formulations and network flow formulations.

A set-covering/partitioning formulation of CSPs is usually solved by column generation and decomposition–based solution methods. For example, Caprara et al. (1997, 1999) studied railway crew scheduling for an Italian Railway company. They followed the approach used in airline crew scheduling and decomposed the problem into three subproblems: (1) pairing generation; (2) pairing optimization; and (3) rostering optimization. Pairing generation is the process of constructing all or a large number of feasible pairings, which is a sequence of trips based on a given train's schedule. Then, pairing optimization is employed to select a subset of pairings so that all trips are covered at the minimum cost. Finally, the selected pairings are sequenced into rosters to be assigned to each individual crew member. Overall, they modeled the CSP as a set-covering problem and used column generation to solve the model. A similar approach was employed by Vance et al. (1997) for airline crew scheduling. To obtain integer-feasible solutions, the column generation algorithm is usually embedded with a branch and bound/price method.

In previous studies, one reason for employing the set-covering approach was the difficulty of modeling complex regulations concerning flight/trip time and required rest in crew pairings. Vance et al. (1997) also presented airline restrictions on the construction of a legal pairing, which consists of more than two flights. The feasibility and total cost of a pairing depends not only on each flight and connections between flights but also on the overall features of a pairing. However, in our case involving freight and double-ended railway districts, travel is between two ending terminals and regulations restrict only the rest hours between an inbound trip i and an outbound trip j, which transfers the crew on trip i. Therefore, instead of generating a pairing/variable for each sequence of trips, we only create connection variables for valid connections from every inbound trip to every outbound trip at each terminal. While a connection from an inbound trip to an outbound trip represents a short pairing, which consists of only two trips, this formulation idea follows the network flow approach. Clearly, the advantage of

formulating the problem as a network flow–based mixed-integer programming problem is model reduction, especially in terms of the number of decision variables.

Cappanera and Gallo (2004) formulated the airline crew rostering problem as a 0-1 multi-commodity flow problem satisfying a set of collective agreements and security rules. Moreover, they provided computational results obtained with a commercial integer programming solver (CPLEX). Sahin et al. (2011) solved the freight railway crew planning problem at the tactical level by using a time-space network model. Their tactical crew planning problem involves finding the minimum number of crews required to operate a set of scheduled trains while satisfying the strict day-off requirement for crew members.

Crew workload balance has been discussed in airline and railway crew scheduling. A few workload balance requirements focus on averaging workload for individual labor. For example, Yu et al. (2003) attempted to minimize the variance in flight hours among the pairings created for a crew base. Jütte et al. (2011) presented sample requirements from more than 100 contractual and legal requirements that restrict crew scheduling in terms of a crew's work times, break times, and driving times in European railways. Moreover, *crew base constraints* in previous studies aim to allocate workload to crew bases within predetermined ranges. Ernst et al.'s (2001) method requires that workloads assigned to each crew pool be within the specified bounds at the end of the planning horizon. Caprara et al. (1997, 1999) formulated crew base constraints in a set-covering formulation. Crew base constraints are similar to our WIC constraints, and WIC constraints are applied to intermediate time points during the planning horizon. However, the MCR rule is unique, and no such requirement is found in literature.

The FIFO rule is not a new topic in crew scheduling. However, few researchers have solved an optimization model with the FIFO rule to optimality because modeling this rule explicitly makes large instances computationally intractable. Vaidyanathan et al. (2007) proposed an optimization model that formulates the FIFO rule, but their model is computationally intractable for large-scale instances because of the substantial number of FIFO constraints required. Therefore, they formulated a FIFO-relaxed model that drops

FIFO constraints. Then, they sequentially added FIFO constraints, which were violated by intermediate solutions while solving the FIFO-relaxed problem. However, the high runtime of this approach motivated them to develop other algorithms. In the end, they perturbed connection costs to automatically satisfy FIFO requirements in the solution for the FIFO-relaxed model. Kliewer et al. (2006) presented a time-space network for multi-depot bus scheduling. They first aggregated connections, solved the simplified problem by using an optimization model, and then used the FIFO rule to decompose crew flows to crew rotations without losing optimality. Instead of formulating the FIFO rule, they used it to decompose crew flows and extract a schedule that satisfies the FIFO rule from multiple optimal solutions. Their approach is valid because they assume the waiting cost at away stations for each vehicle is a linear function of the waiting time starting at zero, which is different from the waiting cost function in our problem.

Very few researchers have studied the FOFO rule. Gorman and Sarrafzadeh (2000) studied railway crew scheduling in single-ended districts, where the activation order for home crew pools follows the FIFO rule and that for away crew pools follows the FOFO rule. They did not formulate a problem but used dynamic programming to optimally solve a restricted version of the problem; then, they used a heuristic algorithm to improve the solution. They repeated this procedure until the heuristic algorithm could not improve the solution anymore. Balakrishnan et al. (2014) discussed railway crew scheduling in primary-secondary-queue districts. They developed push-pull and RK constraints to formulate their activation rules. Although they reduced the number of RK constraints by strengthening RK constraints for trips with different capacities, an instance with 61 regular trips requires 62,000 RK constraints in their model.

To the best of our knowledge, thus far, no study has simultaneously considered workload balance requirements and RK rules in a CSP.

2.3. Problem ingredients

A freight railroad network is geographically segmented into districts, and crew planners manage crew scheduling within each district. We summarize the features of balanced crew scheduling in double-ended districts as follows:

2.3.1 Crew pools and trip rate

In double-ended districts, crew members are homed at one of the terminals at both ends. Each terminal has home crew pools for regular and extra home crews, and away crew pools for regular and extra away crews. Moreover, crew pools vary by occupation. For example, a regular engineer who is homed at terminal *A* departs from terminal *A* and arrives at terminal *B*; he/she leaves the regular home engineer pool in terminal *A* and enters the regular away engineer pool at terminal *B*. Extra crews are always available when no regular crew is fully rested to operate scheduled trains. The company needs to pay a large penalty for calling a new extra crew who will leave the district after returning to his/her extra home pool. The *trip rate* is the cost of assigning a crew member to either a scheduled train or a deadhead trip. Trip rate varies by occupation and crew pool.

2.3.2 Minimum rest hours, maximum layover time, and layover cost

Crew members need at least 11.5 hours (*Minimum rest hours*) to be fully rested (available) for the next trip, as required by FRA regulations. Resting/waiting at the home terminal is free, but the company pays a daily lodging cost for crews waiting at away terminals. In double-ended districts, the *maximum layover time* forbids crews from waiting more than 24 hours at an away terminal. Moreover, if the waiting time at an away terminal exceeds the layover threshold (e.g., 16 hours), a per-hour *layover cost* is charged for the amount of time exceeding the threshold. We define the sum of layover cost and lodging cost as the *layover cost*.

2.3.3 Deadheading modes and costs

Deadheading is unproductively repositioning crew from one terminal to the other. However, deadheading can cover crew shortages, reduce layover costs, and balance workload across crew pools. There are three modes of deadhead trips: passenger train, extra seats in scheduled trains (i.e., train deadhead), and taxi. These transportation modes vary in speed, capacity, and cost. The company pays a ticket fare for each crew on a passenger train. We assume there is no capacity limit for a passenger train. Using train deadheads is free, but the capacity is limited. The company has to pay a fixed cost for each taxi. Taxi capacity is five crew members. Note that each crew on deadhead trips

also charges a trip rate, which is the same as the cost of assigning a crew member to operate a scheduled train.

Available train deadheads and passenger trains are given in the input data. Taxi deadhead trips are available at any time during the planning horizon. For each away crew departing before the planning horizon, we add an outbound taxi candidate that departs at their fully rested time to send the away crews back to their home station. For each scheduled train, we add an inbound taxi candidate to cover potential crew shortages on the train and an outbound taxi candidate to send potential away crews back to the home station.

2.3.4 Workload balance requirements

The workload balance requirements include two sets of constraints that are applied with different frequencies. First, the MCR requirement balances workloads across crew pools for consecutive departing trains at each terminal. By this rule, at each terminal *t* (A or B), crew planners cannot consecutively assign more than a given number of crews homed at station *s* (A or B) to regular trains departing from station *t*. For instance, at terminal *A*, if the maximum allowed number is five, we cannot consecutively assign six crews from the same crew pool to outbound trips from this terminal. Although a small proportion of double-ended districts apply this requirement to both regular and deadhead trips, our model focuses on regular trips only. Secondly, in a district, the WIC constraint requires that the ratio of crew members from each station assigned to trips must equal to a target value or remain within a given range. Crew planners need to not only balance workload at the end of the planning horizon but also control workload imbalance along the planning horizon (e.g. every 16 hours). Past assignments are also counted in this constraint. Both the MCR and WIC constraints are soft constraints. We penalize per unit violation.

2.3.5 Rotation Key rules

The RK rules govern the order of activating crews from a crew pool. According to the FIFO rule, crews from each crew pool are activated in a first-in first-out order. According to the FOFO rule, crews are assigned to trains in the same order they were

activated in the last assignment before arriving into the current crew pool. Both FIFO and FOFO aim to fairly assign workloads to crews within each crew pool. In real-life railway operations, scheduled trains with different priorities vary in speed and traversing time. Moreover, compared to freight trains, taxis usually require less time. Therefore, a crew member may arrive later than other crews who have departed after him/her. We call this situation as *crossing*. Figure 2.1 shows an example of crossing. The timeline at each station in the figure indicates the departing time (start of a trip arc) and the fully rested time (end of a trip arc) for each trip. The planning time horizon starts at time 0 and ends at time *T*. In Figure 2.1 (a), the crew in trip 1 has higher FOFO-priority than the crew in trip 2 because trip 1 departs earlier than trip 2.

The FOFO rule makes an exception for the crossing. If a higher FOFO-priority crew is not fully rested when an outbound train departs, crew planners can assign a lower FOFO-priority crew to the outbound train. Figure 2.1 (b) shows an example of the FOFO exception. In the example, the crew in trip 2 has lower FOFO priority than the crew in trip 1, but the crew is activated when trip 3 departs. This exception is made because the crew in trip 1 is not fully rested when trip 3 departs.

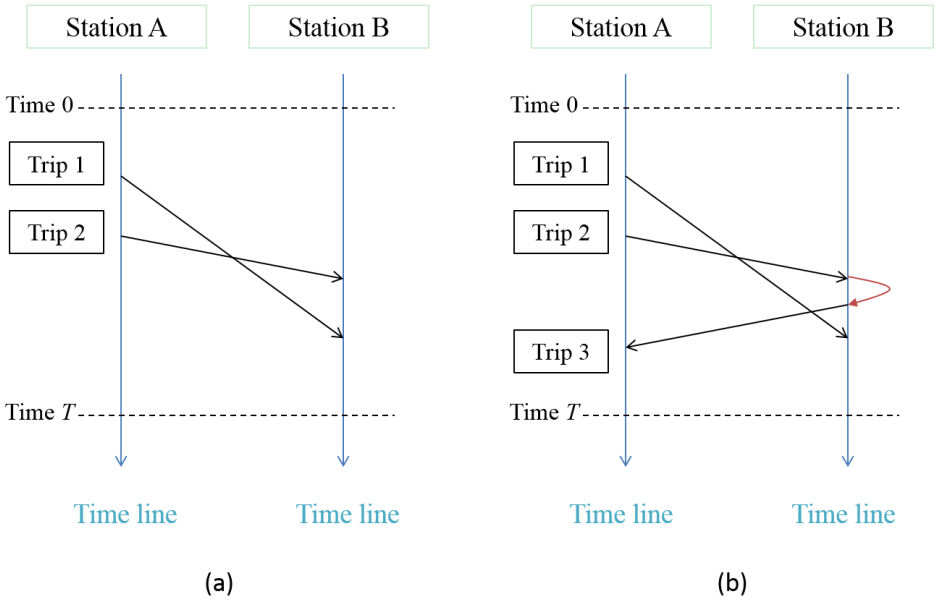


Figure 2.1. Crossing situation and FOFO rule exception

We first present a basic optimization model for the railway CSP in Section 2.4, which is denoted as Model 1. This basic model can solve small instances by standard optimization software (i.e., CPLEX). In Section 2.5, model improvement helps us solve medium and large-scale instances.

2.4. Model formulation

We formulate the problem as a network-flow based mixed integer programming problem. For every fully rested connection from an inbound trip to an outbound trip, we create connection variables to represent whether the outbound trip transfers crew of specific occupations and pools from the inbound trip. If the connection is selected and the outbound trip is a scheduled train, then the train's crew requirement is covered by the crew from the inbound trip. Our goal is to select a minimum cost set of connections such that every scheduled train is covered. This model is defined as Model I.

Notation

K = set of occupation types: $K = \{Engineer, Conductor\}$, indexed by k.

M = set of crew pools: $M = \{AR, BR, AX, BX\}$, indexed by m. Each pool is also identified by homing station $s = A$ or B and if the pool is a regular board (R) or an extra board (X).

$M(s)$ = set of crew pools that are based at station s. $M(s) = \{sX, sR\}$.

IC = set of initial trips (including in-transit trips at time zero), corresponding to initial crews who are at terminals or in-transit at time zero.

TR = set of regular train trips scheduled during the time horizon.

DH = set of deadhead trips including train deadhead (DH_{train}), taxi (DH_{taxi}), and passenger train (DH_{PTrain}).

$DH = DH_{train} \cup DH_{taxi} \cup DH_{PTrain}$

E = set of all trips (including initial, scheduled, and deadhead trips) over the time horizon.

$E = IC \cup TR \cup DH$.

ss_i, se_i = starting station and ending station for trip i.

ts_i, te_i = starting time and fully rested time for trip i.

$P(j)$ = set of immediate *predecessors* that can connect to trip $j \in TR \cup DH$.

$S(i)$ = set of immediate *successors* to which trip $i \in E$ can connect.

Parameters

R_i^{km} = number of crews of occupation type k from pool m who arrived on initial trip i. $R_i^{km} \in \{0,1\}$.

U_h = capacity of deadhead trip $h \in DH$.

Costs

CC_{ij}^{km} = "connection" cost for a crew of occupation type k from pool m to connect from trip i to trip j, for all $i \in E, j \in S(i)$; this cost depends on the rest time between trips i and j and the pool m, and includes the costs for lodging and layover time at the terminal se_i.

TC_i^{km} = cost of each crew member of occupation k and pool m on trip i. Passenger train deadhead trips also need ticket fee for each crew.

FC = fixed setup cost per taxi.

BC^k = new extra crew member cost for an occupation k crew from extra crew pool.

Decision Variables

x_{ij}^{km} = 1 if inbound trip i connects to outbound trip j by a crew of occupation type k from pool m; 0 otherwise, for all $k \in K, m \in M, j \in TR \cup DH$ and $i \in P(j)$.

v_i^{km} = number of crews of occupation type k from pool m transported by trip i.

y_h = number of taxi deadhead trips which are selected for taxi candidate h. In the model the upper bound of y_h is 3.

Objective function

$$\min \sum_{k\in K}\sum_{i\in E}\sum_{j\in S(i)}\sum_{m\in M(ss_i)} CC_{ij}^{km} x_{ij}^{km} + \sum_{k\in K}\sum_{i\in TR\cup DH}\sum_{m\in M} TC_i^{km} v_i^{km} + \sum_{h\in DH_{taxi}} FC y_h + \sum_{k\in K}\sum_{i\in TR} BC^k v_i^{k,ss_i X} \qquad (2.1)$$

Our objective is to minimize a combination of costs, including connection costs, crew trip costs, taxi deadhead fixed cost and new extra crew penalty. We made workload balance requirements soft by introducing slack variables which represent unit violation

and pay penalties. We will present the constraint violation penalties in the formulation of each constraint.

Flow Conservation Constraints

$$\sum_{j \in S(i)} x_{ij}^{km} = R_i^{km} \qquad \text{for all } i \in IC,\ k \in K,\ m \in M \tag{2.2}$$

$$\sum_{j \in S(i)} x_{ij}^{km} = \sum_{l \in P(i)} x_{l,i}^{km} \qquad \text{for all } i \in TR \cup DH,\ k \in K,\ m \in M \tag{2.3}$$

$$\sum_{j \in S(i)} x_{ij}^{km} = v_i^{km} \qquad \text{for all } i \in TR \cup DH,\ k \in K,\ m \in M \tag{2.4}$$

Constraint (2.2) ensures one crew from initial inbound trip is assigned to exactly one outbound trip. Constraint (2.3) states that the sum of crews assigned to trips i equals the sum of crews released from trip i. Therefore, these two constraints ensure crew flow conservation. Constraint (2.4) defines variable v_i^{km} by summing up associated connections from trip i.

Crew Requirement Constraints

$$\sum_{m \in M} v_i^{km} = 1 \qquad \text{for all trip } i \in TR,\ k \in K \tag{2.5}$$

Constraint (2.5) ensures required number of crews in each occupation are assigned to each scheduled train. In particular, every scheduled train requires exact one crew member of each occupation.

Deadhead Capacities Constraints

$$\sum_{k \in K} \sum_{m \in M} v_h^{km} \le U_h y_h \qquad \text{for all } h \in DH_{taxi}, \tag{2.6}$$

$$\sum_{k \in K} \sum_{m \in M} v_h^{km} = U_h \qquad \text{for all } h \in DH_{train}, \tag{2.7}$$

Deadhead trips have different capacities based on trip type. The capacity of each train deadhead is given as input data. Each taxi deadhead trip has capacity of five and charges a fixed taxi setup cost.

Maximum Calling Ratio Constraints

ρ = maximum number of crew members of occupation k homing at the same station that can be consecutively dispatched from a (home or away) station.

$TR^+(\rho)$ = set of all scheduled train trips excluding the first ρ scheduled trains starting from each station.

$DR_i(\rho)$ = set of trips consisting of trip $i \in TR$ and the ρ preceding scheduled trains that depart from the same station as trip i.

s_i^k = nonnegative slack variable which indicates units of violation for MCR constraints as of trip i, for occupation k.

$$\sum_{j \in DR_i(\rho)} \sum_{m \in M(ss_i)} v_j^{km} + s_i^k \geq 1, \qquad \text{for all } i \in TR^+(\rho),\ k \in K. \tag{2.8}$$

$$\sum_{j \in DR_i(\rho)} \sum_{m \in M(se_i)} v_j^{km} + s_i^k \geq 1 \qquad \text{for all } i \in TR^+(\rho),\ k \in K. \tag{2.9}$$

Constraint (2.8) enforces that there is at least one home crew among the ρ+1 crews assigned to trip i and ρ consecutive scheduled trains before trip i. Similarly, constraint (2.9) ensures that there is at least one away crew among those ρ+1 crews. If both constraints (2.8) and (2.9) are satisfied, then the crew assignment as of trip i meets the MCR requirement. Moreover, crew planners allow constraint violation with large penalties. So we introduce non-negative variable s_i^k to represent violations as of trip i and impose a large penalty for every unit of s_i^k in the objective function.

Workload Imbalance Control Constraints

A_0^k = number of trips assigned to crews who are homed at terminal A as of time zero for occupation k.

B_0^k = number of trips assigned to crews who are homed at terminal B as of time zero for occupation k.

p^k = target ratio of crew members from each station assigned to trips for occupation k.

BE = the set of time points when WIC constraints are applied.

$BE = \{t \mid t \in \{z \times f, z \in Z^+\}, t \leq T\}$, where f is the time interval between two time points.

bs_t^{ks} = nonnegative slack variable representing the number of additional crews that are homed at station s would have to instantaneously activate to achieve the target ratio as of time t, for $t \in BE$, $k \in K$, $s = A$ or B. Every unit of bs_t^{ks} results a small penalty.

g = upper bound on the nonnegative slack variable bs_t^{ks}. It represents a range of imbalance where only a small penalty is charged.

fs_t^{ks} = nonnegative slack variable which indicates units of violation for WIC constraints as of time $t \in BE$, $k \in K$, $s = A$ or B. The penalty for positive fs_t^{ks} is much larger than the penalty for bs_t^{ks}.

$$\frac{A_0^k + \sum_{i \in E \setminus IC, ts_i \le t} \sum_{m \in M(A)} v_i^{km} + bs_t^{kA} + fs_t^{kA}}{B_0^k + \sum_{i \in E \setminus IC, ts_i \le t} \sum_{m \in M(B)} v_i^{km} + bs_t^{kB} + fs_t^{kB}} = p^k \qquad \text{for } t \in BE,\ k \in K \qquad (2.10)$$

$$bs_t^{kA} + bs_t^{kB} \le g \qquad \text{for } t \in BE,\ k \in K \qquad (2.11)$$

Suppose the target ratio p^k equals to 1. If the number of activated crews homed at terminal A ($A_0^k + \sum_{i \in E \setminus IC, ts_i \le t} \sum_{m \in M(A)} v_i^{km}$) is fewer than the number of activated crews homed at terminal B as of time t, then $bs_t^{kA} + fs_t^{kA}$ becomes positive to achieve the target ratio. Slack variable bs_t^{kA} increases up to upper bound g with small per unit penalty, while fs_t^{kA} stays at zero. If the workload imbalance as of time t is out of preferred range, which means bs_t^{kA} reaches its upper bound and $bs_t^{kA} + fs_t^{kA}$ needs to increase further, then variable fs_t^{kA} becomes positive and large penalties for positive fs_t^{kA} are charged. Let CBS denote the penalty for positive bs_t^{kA} variables and CFS denote the penalty for positive fs_t^{kA}. The cost term for WIC constraints is

$$CFS * \sum_{t \in BE} \sum_{k \in K} (fs_t^{kA} + fs_t^{kB}) + CBS * \sum_{t \in BE} \sum_{k \in K} (bs_t^{kA} + bs_t^{kB}) .$$

Connection-based RK Constraints

$IC(i, j)$ = set of connections $<i', j'>$ that are RK incompatible with connection $< i, j>$.

$$\sum_{m \in M(s)} x_{ij}^{km} + \sum_{m \in M(s)} x_{i'j'}^{km} \leq 1$$

for $i \in E, j \in S(i), <i', j'> \in IC(i, j), k \in K$ and $s = ss_i = ss_{i'}$ (2.12)

Constraint (2.12) eliminates RK violations by enumerating and forbidding all possible RK-violated connection pairs. This formulation generates $O(n^4)$ RK constraints where n is the total number of trains.

If a trip's maximum capacity for one occupation k equals to 1, then it is a *single-job* trip. Therefore, the sum of away connection variables from a single-job trip to its all successors for occupation k is up to 1. We can modify constraint (2.12) for single-job trip i and i' based on this property.

$$\sum_{ts_j < ts_h,\, j \in S(i) \cap S(i')} \sum_{m \in M(s)} x_{i'j}^{km} + \sum_{ts_{j'} \geq ts_h,\, j' \in S(i)} \sum_{m \in M(s)} x_{ij'}^{km} \leq 1$$

for $i, i' \in$ single-job trip, $s = ss_i = ss_{i'}$, trip i has higher RK priority than trip i', trip $h \in S(i) \cap S(i')$ and $k \in K$ (2.13)

Suppose trip i has higher RK priority than trip i'. If single-job trip i' connects with trip j that is a common successor of trip i and i' and single-job trip i connects with trip j' which departs after the trip j, then it is RK-violated connection pair and the left hand side of the corresponding constraint (2.13) equals to 2. This formulation generates $O(n^3)$ constraint (2.13) where n is the total number of trains. We also modify constraint (2.12) when one of trip i and i' is a single-job trip. In Section 2.5, we provide an alternative formulation which requires much fewer constraints.

Balakrishnan et al. (2014) propose a connection-based RK formulation, in which their basic RK constraints are the same as constraint (2.12). They strengthen the basic constraints and reduce the number of constraints according to capacities of trip i, i', j and j' in constraint (2.12). Constraint (2.13) is one of their strengthened RK constraints. Vaidyanathan et al. (2007) also present a connection-based RK formulation. They define A_r as the set of connections which will violate FIFO if connection r is selected. For every connection r, their formulation requires one constraint to forbid the solution selecting any connection in A_r when the connection r is selected. Therefore, their model only requires

$O(n^2)$ RK constraints, where n is the number of trips in the planning horizon. The disadvantage of their formulation is that the big M in the FIFO constraints makes the constraint weaker than constraint (2.12).

2.5. Model enhancements

We first improve the computational performance by model reduction, because the huge model size is the bottleneck for solving large-scale instances. We define the new model formulation as Model II. Moreover, adding valid inequalities that increases the LP bound can also improve the performance of our model. The final model with VIs is called Model III.

2.5.1 Reduced flow conservation constraints and home crew inventory constraints

In the flow conservation constraints (2.2), (2.3) and (2.4), we create connection variables for each pair of inbound trip and outbound trip, and for both home pool and away pool. However, we actually do not need connection variables for home pools, because it is free for crew to stay at home station and we do not need to capture the waiting time for each crew at home station. So we propose a new formulation which removes home connection variables. We modify the flow conservation constraints as follows:

$$\sum_{j \in S(i)} x_{ij}^{km} = R_i^{km} \qquad \text{for all } i \in IC,\ k \in K,\ m \in M(ss_i) \tag{2.14}$$

$$\sum_{j \in S(i)} x_{ij}^{km} = v_i^{km} \qquad \text{for all } i \in TR \cup DH,\ k \in K,\ m \in M(ss_i) \tag{2.15}$$

$$\sum_{l \in P(i)} x_{li}^{km} = v_i^{km} \qquad \text{for all } i \in TR \cup DH,\ k \in K,\ m \in M(se_i) \tag{2.16}$$

Without home connection variables, we create *home crew inventory constraints* to avoid infeasible assignments which assign more home crews than the available amount.

w_i^k = nonnegative number of fully rested regular crews of occupation k, homing at station ss_i, who remain in "crew inventory" at station ss_i immediately after trip i departs.

b_i = index of the outbound trip that departs immediately prior to trip i from station ss_i.

AT_i = set of trips (including initial, regular, and deadhead trips) that arrive at station ss_i and get fully rested between the departure times of trip b_i and trip i.

$AT_i = \{j \mid se_j = ss_i \text{ and } ts_{b_i} < te_j \le ts_i\}$.

$i1^s$ = index of first regular or deadhead trip starting from station s.

Note: $w^k_{b_{i1^s}}$ is zero for both stations. $AT_{i1^s} = \{j \mid te_j \le ts_{i1^s}, se_j = s\}$ for station s.

$$w^k_{b_i} + \sum_{j \in AT_i} v^{k,ss_iR}_j = v^{k,ss_iR}_i + w^k_i \quad \text{for all } i \in TR \cup DH,\ k \in K \tag{2.17}$$

Home crew inventory constraint (2.17) captures the flow conservation for regular crews at home crew pool. It guarantees that regular home crews are adequate when the solution assigns regular home crews to an outbound trip.

The benefit of removing home pool connection variables is mainly reducing the model size, especially the number of connection variables. Therefore, we substitute the original flow conservation constraints with the modified flow conservation constraints and home crew inventory constraints.

2.5.2 Remove RK constraints for home crew pool

In each terminal, RK rules are applied to both home crew pool and away crew pool. Since we have removed home crew connection variables, we cannot specify which home crew is assigned to which outbound trip j. To satisfy RK rules at home crew pools, we post-process the optimal solution to specify home connections by RK rules. In the post-processing, we sequentially map the fully rested home crew of the highest RK priority (i.e. first arriving crew in FIFO rule, first departing crew in FOFO rule) to each outbound trip that transports home crews in the solution. In sum, the post-processing ensures RK requirements in home crew pools without loss of optimality, because home connections is always free.

Given that a large amount of RK constraints is a bottleneck for solving this problem, which will be shown in computational results, removing RK constraints at home crew pools from the model is important.

2.5.3 Time-based RK constraints at away terminal

Compared with the connection-based RK constraints in Section 2.4, the time-based formulation is an alternative way of formulating the RK rules. The advantage of this formulation is that the number of constraints it generates is much lower than that in the connection-based formulation. In some cases, the time-based RK constraints can be weaker than the connection-based RK constraints.

Single-job trip means crew requirement for one occupation is one while multi-job trip has more than one crew requirement for single occupation.

t_i^k = nonnegative continuous variable representing the departure time of the last departing time trip j if connection $<i, j>$ for occupation k and crew pools at ss_i is selected in the solution.

$$t_i^k \geq \sum_{j \in S(i)} \sum_{m \in M(ss_i)} ts_j x_{ij}^{km} \quad \text{for all single-job trips } i \in E, \text{occupation } k \in K \tag{2.18}$$

$$t_i^k \geq \sum_{m \in M(ss_i)} ts_j x_{ij}^{km} \quad \text{for all multi-job trips } i \in E, j \in S(i), \text{and occupation } k \in K \tag{2.19}$$

FIFO rule

n_i = index of earliest arriving time trip that arrives after trip i at terminal se_i

$$t_i^k \leq t_{n_i}^k \quad \text{for every trip } i \in E, k \in K \tag{2.20}$$

If trip n_i is a single-job trip:

$$t_i^k \leq \sum_{j \in S(n_i)} \sum_{m \in M(ss_i)} ts_j x_{n_i,j}^{km} + T\left(1 - \sum_{j \in S(n_i)} \sum_{m \in M(ss_i)} x_{n_i,j}^{km}\right) \quad \text{for every trip } i \in E, k \in K \tag{2.21}$$

If trip n_i is a multi-job trip:

$$t_i^k \leq ts_j \sum_{m \in M(ss_i)} x_{n_i,j}^{km} + T\left(1 - \sum_{m \in M(ss_i)} x_{n_i,j}^{km}\right) \quad \text{for every trip } i \in E, j \in S(n_i), k \in K \tag{2.22}$$

According to the FIFO rule, in the away crew pool for occupation k at station se_i, crews from trip i have higher FIFO priority than crews from trip n_i, because trip i arrives earlier than trip n_i. Therefore, away crews of occupation k from trip i should be activated earlier than any away crew of occupation k from trip n_i. For the sake of simplicity in the following two paragraphs, we omit occupation difference.

Constraints (2.18) and (2.19) are forcing constraints for the variable t_i^k that represents the last time of activating crews who are homed at pool $M(ss_i)$ (i.e. away crew at station se_i) and transported by trip i. If trip i does not transport any away crew, t_i^k can be any nonnegative value. No matter trip i or trip n_i transports away crews, constraint (2.20) ensures that t_i^k is not greater than $t_{n_i}^k$.

Suppose trip n_i transports crews that are homed at pool $M(ss_i)$, in which case crew assignment for trip i and n_i may violate the FIFO rule. Constraints (2.21) and (2.22) force t_i^k to be no greater than the activation time of any away crew from trip n_i.

In sum, we only need to enforce FIFO rules between consecutive arriving trips so that we can reduce the total number of FIFO constraints to $O(n)$ if all arriving trips are single-job trips. In the worst case, only $O(n^2)$ FIFO constraints are needed in our formulation where n is the total number of trains. The same idea can be used to formulate the time-based FOFO constraints.

We define the new formulation in which home connections have been removed and RK constraints are time based as Model II.

2.5.4 Strengthen Model II

The time-based RK constraints in Section 2.5.3 reduce model size dramatically, but computational experiments show that they are weaker than the connection-based RK constraints for most instances. Even though the connection-based RK constraints are not strictly tighter than the time-based formulation, adding particular subsets of the connection-based RK constraints to the model with time-based RK constraints can increase the LP bound in most instances.

We only add constraints (2.13) when trip i and i' are both single-job trips. The reason that we only consider single-job trips case is the large number of connection-based constraints. When trip i or i' is not single-job trip, the large amount of constraints requires huge amount of computer memory but only contributes minor improvement on the LP bound. The final model with VIs is called Model III.

2.6. Solution techniques

Given the model improvement in Section 2.5, the optimization model still cannot quickly solve some difficult instances because of the complicated RK constraints. However, computational results show that solving the exact model without RK constraints is a trivial problem. If a simple routine can repair RK violation quickly and satisfy other constraints, then an intuitive idea of a heuristic algorithm is to first solve the exact model without RK constraints and then heuristically repair the violations. Therefore, we develop an RK-violation-repair heuristic method to quickly solve large instances.

The heuristic solutions are used as initial feasible solutions to warm start the optimization model because they are much better than the first feasible solutions while solving the improved exact model without warm-start.

2.6.1 Two stages in the RK-violation-repair heuristic

The RK-violation-repair heuristic algorithm consists of two stages. In stage 1, it optimally solves the exact problem in which most RK constraints are removed. In stage 2, it repairs RK violations in the solution from the stage 1. Which RK constraints are added in the stage 1 problem is critical in this heuristic algorithm. Section 2.6.2 and 2.6.3 discuss this problem in details.

In stage 2, the repairing procedure sequentially maps the fully rested away crew of the highest RK priority in the away crew pool to each outbound trip that was assigned with away crews in the solution to the stage 1 problem. Figure 2.2 presents the execution flow of the repairing procedure. In the repairing procedure, the fully rested crew of the highest RK priority is always activated first so that the final heuristic solution guarantees to satisfy the RK rule. Moreover, repaired away connections may exceed the max layover time. So actions are taken to fix the maximum layover time violations. Section 2.6.4 explains the cause of the max layover time violations and schemes to prevent it.

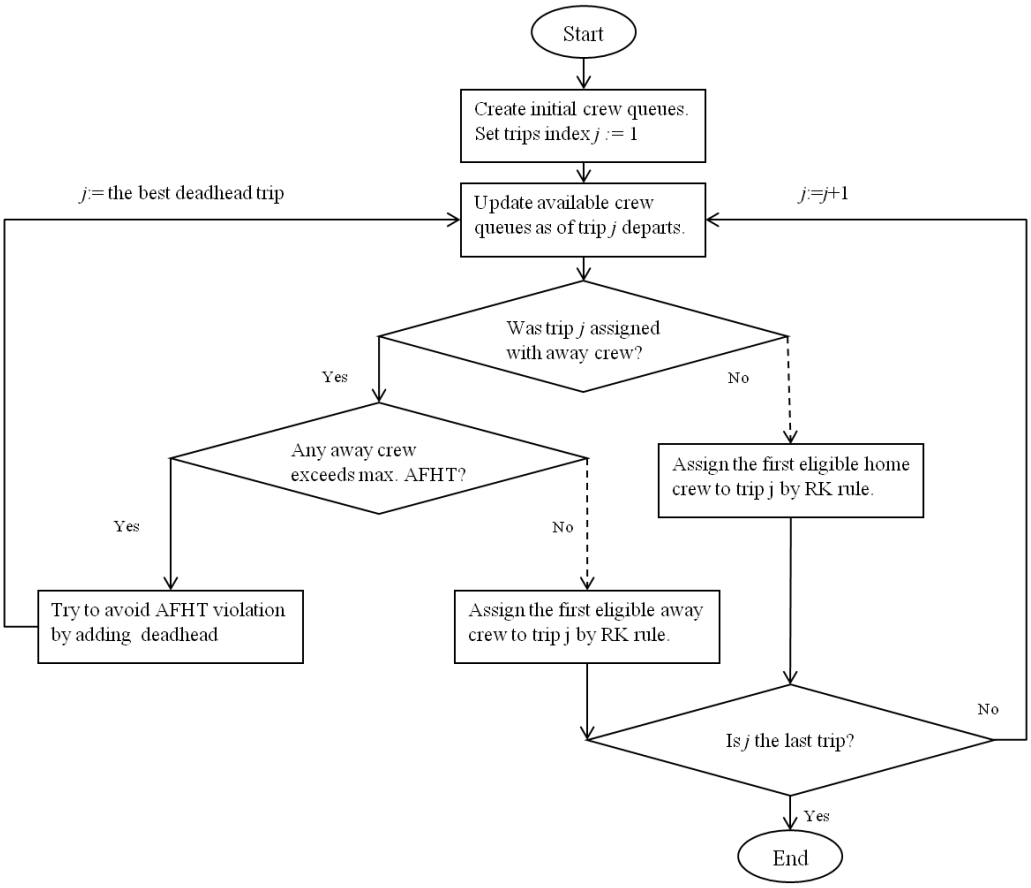


Figure 2.2. Execution flow of the repairing procedure in RK-violation-repair heuristic

2.6.2 Identify and prevent expensive FOFO violations

While relaxing FOFO constraints makes the problem easy to solve, repairing FOFO violations potentially increases the total cost and sacrifices the solution quality. If repairing a FOFO-violated connection pair increases the total layover cost, then it is an *expensive* FOFO violation. In contrast, repairing a FIFO-violated connection pair does not increase the total layover cost.

Figure 2.3 illustrates the total layover cost before and after repairing the FOFO violation. In this example, the total layover cost increases because the repaired away connection <*i*', *j*'> exceeds the layover threshold time (e.g. 16 hours), while the total layover cost of connection pair <*i*, *j*'> and <*i*', *j*> is zero.

Denote the objective values of solutions in stage 1 and 2 as z^1 and z^2 respectively. Let z^* represent the optimal objective value of the exact model, then $z^1 \le z^* \le z^2$. In the best

case, the total layover cost does not increase (no expensive FOFO violation occurs) while repairing FOFO violations in the stage 2. Thus $z^1 = z^2$. Moreover, because of $z^1 \le z^* \le z^2$, the final heuristic solution z^2 is an optimal solution to the exact model. Prohibiting FOFO-violated connection pairs in the solution to stage 1 problem can reduce the gap between z^1 and z^2. This heuristic algorithm tries to add a minimal amount of RK constraints such that the solving time is within ten minutes and the heuristic gap achieves about 0.5% compared with the best solution to Model II.

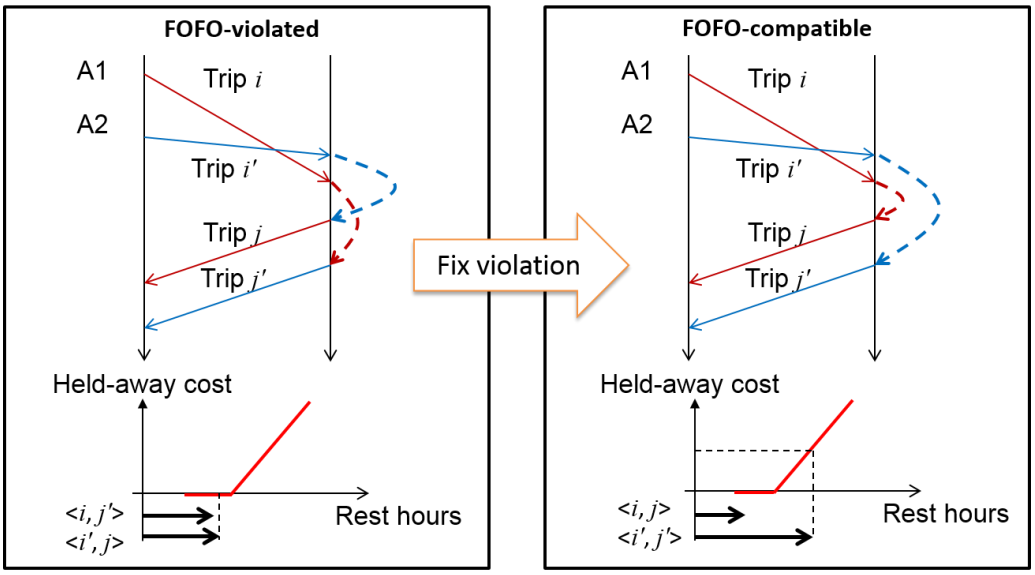


Figure 2.3. An expensive FOFO violation

In sum, if we can identify a small subset of connection-based RK constraints which eliminate the expensive FOFO-violated connection pairs, then we add this subset of RK constraints to the stage 1 problem. Proposition 1 proposes a necessary and sufficient condition for an expensive FOFO-violated connection pair.

Assume that trip *i* and *i*' are assigned with regular away crews homed at terminal ss_i. Trip *i* departs earlier than trip *i*', but arrives later than trip *i*'. At the arrival station of trip *i* and *i*', trip *j* departs earlier than trip *j*'. Let *NHW*(*i*) be the time interval when crews from the trip *i* are fully rested and no layover cost is charged, i.e., the 4-hour window from fully-rested time to layover threshold. Figure 2.4 shows the time intervals for trip *i* and *i*'.

Proposition 1: Repairing FOFO-violated connections <*i*', *j*> and <*i*, *j*'> increases the total layover cost for crews in trip *i* and *i*' if and only if trip *j* departs within *NHW*(*i*) and trip *j*' departs after *NHW*(*i*').

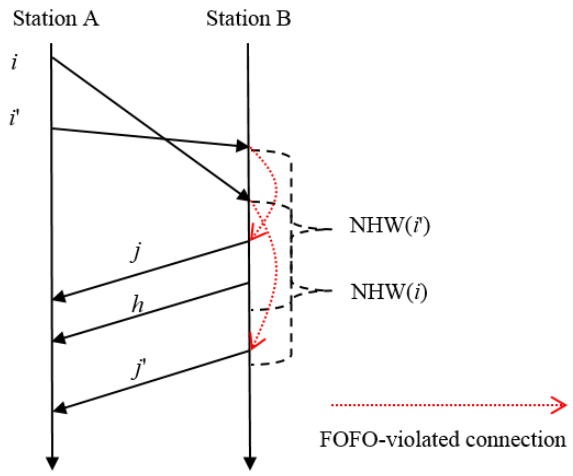


Figure 2.4. An example in Proposition 1

Proof: The condition is clearly sufficient, because repairing FOFO violation for this connection pair strictly increases layover cost. Only need to proof it is necessary.

First, we want to prove trip *j*' departs after the time interval *NHW*(*i*'). Because the connection time of <*i*, *j*> is less than the connection time of <*i*', *j*>, so the layover cost of <*i*, *j*> is no more than the layover cost of <*i*', *j*> which means cost(*i*, *j*) $\leq$ cost(*i*', *j*). Because cost(*i*', *j*')+cost(*i*, *j*) > cost(*i*', *j*)+cost(*i*, *j*') and cost(*i*, *j*) $\leq$ cost(*i*', *j*), then cost(*i*', *j*') > cost(*i*', *j*). Therefore, trip *j*' departs after the time interval *NHW*(*i*').

Second, we need to proof that trip *j* departs within the time interval *NHW*(*i*). Suppose trip *j* departs after the NHW(*i*), which means the departure time of trip *j* is 16 hours after the arrival time of trip *i*. So the departure time of trip *j*' is also 16 hours after the arrival time of trip *i*.

The layover hours of <*i*', *j*> equals to departure time of *j* – (arrival time of *i* + 16) + (arrival time of *i* + 16) – (arrival time of *i*' + 16). The layover hours of <*i*, *j*'> equals to the departure time of *j*' – (arrival time of *i* + 16).

The layover hours of <*i*, *j*> equals to departure time of *j* – (arrival time of *i* + 16). The layover hours of <*i*', *j*'> equals to departure time of *j*' – (arrival time of *i*' + 16).

In sum, layover hours of <i', j> + layover hours of <i, j'> = layover hours of <i, j> + layover hours of <i', j'> which counterparts with the positive increment of layover cost. Moreover, if trip j departs before the $NHW(i)$, then connection pair <i', j> and <i, j'> does not violate FOFO rule. Therefore, trip j departs within $NHW(i)$.

In conclusion, trip j' departs after the time interval $NHW(i')$ and trip j departs within the time interval $NHW(i)$. ■

Based on the Proposition 1, we formulate constraints (2.23) to eliminate expensive FOFO-violated connection pairs. Without Proposition 1, we have to create constraints (2.12) for every pair of trip j and j'. But now, for each pair of inbound trip i and i', constraint (2.23) only considers trip j and j' that satisfy the condition in the Proposition 1. Therefore, the number of connection-based RK constraints needed reduces.

$$\sum_{m\in M(ss_i)} x_{i,j'}^{k,m} + \sum_{m\in M(ss_i)} x_{i',j}^{k,m} \le 1$$ for any trip $i \in E$, trip i' crossing trip i, trip j departs within $NHW(i)$, trip j' that departs after $NHW(i')$, $ts_j < ts_{j'}$, occupation $k \in K$. (2.23)

If trip i or trip i' is a single-job trip, we can aggregate connections from trip i or i' so that the constraint (2.23) can be modified as (2.24), (2.25) and (2.26). After the aggregation, the number of constraints reduces further.

$$\sum_{j'\in S(i),ts_{j'}\ge ts_{j^*}} \sum_{m\in M(ss_i)} x_{i,j'}^{k,m} + \sum_{j\in S(i')\cap S(i),ts_j<ts_{j^*}} \sum_{m\in M(ss_i)} x_{i',j}^{k,m} \le 1$$

for any single-job trip $i \in E$, single-job trip i' that crosses trip i, trip j^* departs within $NHW(i)\backslash NHW(i')$, occupation $k \in K$. (2.24)

$$\sum_{j'\in S(i),ts_{j'}\ge ts_{j^*}} \sum_{m\in M(ss_i)} x_{i,j'}^{k,m} + \sum_{m\in M(ss_i)} x_{i',j}^{k,m} \le 1$$

for any single-job trip $i \in E$, multi-job trip i' that crosses trip i, trip j^* departs within $NHW(i)\backslash NHW(i')$, trip $j \in S(i)\cap S(i')$, $ts_j < ts_{j^*}$, occupation $k \in K$. (2.25)

$$\sum_{m\in M(ss_i)} x_{i,j'}^{k,m} + \sum_{j\in S(i')\cap S(i),ts_j<ts_{j^*}} \sum_{m\in M(ss_i)} x_{i',j}^{k,m} \le 1$$

for any multi-job trip $i \in E$, single-job trip i' that crosses trip i, trip j^* departs within $NHW(i)\backslash NHW(i')$, trip $j' \in S(i)$, $ts_{j'} \ge ts_{j^*}$, occupation $k \in K$. (2.26)

2.6.3 The impact of cascading order of crew assignment

To avoid increasing the total layover cost in the repairing procedure, the heuristic adds constraints (2.24), (2.25) and (2.26) to the stage 1 problem. Even though expensive FOFO-violated connection pairs are eliminated in the stage 1 solution, the total layover cost may still increase while repairing FOFO violations in stage 2. That is because of the cascading of assigning crews. For the example in Figure 2.4, suppose the away crew in trip i is now assigned to trip h which departs within $NHW(i')$ in the stage 1 solution. So the FOFO-violated connection pair $<i', j>$ and $<i, h>$ is not an expensive violation based on Proposition 1. When repairing FOFO violations in stage 2, the away crew in trip i is assigned to trip j. Nevertheless, when assigning away crews to trip h, the away crew in trip i' may not be the highest-priority away crew and this away crew has to wait to be assigned to trip j'. That leads to an increment of the total layover cost because of the away connection $<i', j'>$. In order to prevent all possible expensive FOFO-violated assignments for crews in trip i and i' in stage 2, the heuristic algorithm needs the same amount of RK constraints as the exact model.

2.6.4 Satisfy the maximum layover time limit

In the exact model, every feasible solution satisfies the maximum layover time limit (e.g. 24 hours), because the exact model does not generate away connections exceeding 24 hours. But, a repaired away connection may be extended such that it exceeds the max layover time. For the example in Figure 2.3, if trip j' departs more than 24 hours later than the arrival time of trip i', then the repaired away connection $<i', j'>$ exceeds the max layover time. Instead of developing a sophisticated repairing heuristic algorithm which re-decides assigning home or away crew to each trip to fix the max layover time violations, we develop with a scheme of preventing the max layover time violations in the repairing procedure. Suppose a single-job trip i is crossed by a single-job trip i'. We add the valid inequality (2.27) to the stage 1 problem.

$$\sum_{j' \in S(i) \setminus S(i')} \sum_{m \in M(ss_i)} x_{i,j'}^{k,m} + \sum_{j \in S(i') \cap S(i)} \sum_{m \in M(ss_i)} x_{i',j}^{k,m} \leq 1$$

for any single-job trip i, single-job trip i' that crosses trip i, occupation $k \in K$. (2.27)

In constraint (2.27), the first summation on the left hand side is the sum of valid connections from trip *i* to successors departing more than 24 hours later than the arrival time of trip *i'*, and the second term sums up connections from trip *i'* to common successors of trip *i* and *i'*. Suppose the left hand side equals to 2, then this solution violates FOFO rule. After repairing FOFO violation like the example in Figure 2.3, the away crew in trip *i'* is assigned to trip *j'* so that this away connection exceeds 24 hours. In sum, the valid inequality (2.27) helps to prevent AFHT violation in the repairing procedure. Since the constraint (2.27) is only applied to the case in which both trip *i* and *i'* are single-job trips, it cannot guarantee to prevent repaired away connections exceeding the max layover time limit. We tested adding the constraints (2.27) for multi-job trip *i* or *i'*, but it does not help to avoid the maximum layover time violations but increases the runtime. In fact, the cascading of assigning away crews is another reason why constraint (2.27) cannot prohibit repaired away connections exceeding the max layover time even for single-job trips.

If any extended away connection exceeds the max layover time in the FOFO-violation repairing procedure, the heuristic still tries to avoid the max layover time violations by adding additional deadhead trips to send those away crews back to their home station before the max layover time limit. The computational results indicate that no extended away connection exceeds the max layover time limit in all final heuristic solutions, but some violation-repaired solutions pay additional deadhead cost because of fixing the max layover time violations.

2.6.5 Summarize valid inequalities in stage 1 problem

The RK-violation-repair heuristic algorithm for FOFO rule adds constraints (2.24), (2.25) and (2.26) to the stage 1 problem. Moreover, to prevent the max layover time violations in the repairing procedure in stage 2, constraint (2.27) is also included in the stage 1 problem.

We do not add RK constraints to the stage 1 problem when we apply this heuristic to instances with FIFO rule, because repairing FIFO violations does not increase the total layover cost.

2.7. Computational results

In this section, we present computational results on real-life problem instances from a major North American railway company and demonstrate that the model improvements and solution techniques in Section 2.5 and 2.6 are effective. In Section 2.7.2, we conduct computational experiments to test the robustness of our optimization model and study how solution structures change when critical input features vary. We implemented our models and methodologies with Java programming language and solved them using ILOG CPLEX 12.5 solver. All computational tests were conducted on a 3.33 GHz processor with 2 GB RAM.

2.7.1 Validation of model enhancements and solution techniques

We first demonstrate the effectiveness of the model enhancements and solution techniques. Test instances are collected from three districts representing small, medium (M1~ M7) and high traffic volume districts (L1~ L9). Instances with small traffic volume are simple problems and even Model I can solve them quickly. Therefore, results for small instances are not presented. In all tests, the termination criterion is 1% MIP gap, and the maximum runtime is one hour. The MIP gap stands for the gap between the best integer objective and the objective of the best node remaining. We also use *final gap* to represent the same meaning.

Table 2.1 summarizes the problem size of all instances. In the medium size instances, the number of regular crews includes regular conductors only while in large size instances it includes regular conductors and engineers. Deadhead trips include all deadhead trips added in the model. Scheduled trains are trains that are planned in advance. Instances from medium traffic volume districts only contain one occupation type, while large instances consider engineers and conductors simultaneously. Therefore, although medium instances contain nearly the same amount of trips as large instances, the number of connection variables in large instances is about two times as many as that in medium instances in Table 2.2.

Table 2.1. Summary of problem size

Instance	Problem size			
	# of regular crews	# of scheduled trains	# of deadhead trips	# of total trips
M1	127	158	164	322
M2	128	124	121	245
M3	130	127	134	261
M4	128	157	140	297
M5	118	163	167	330
M6	193	184	190	374
M7	138	143	148	291
L1	408	156	154	310
L2	374	185	181	366
L3	466	177	157	334
L4	418	184	187	371
L5	434	139	134	273
L6	428	156	159	315
L7	464	132	137	269
L8	442	170	162	332
L9	368	177	163	340

Table 2.2 shows the effectiveness of model reduction. Note that Model I generates millions of connection-based RK constraints, especially for home pools. Model II eliminates more than a half of connection variables by removing home connections. The reduction of connection variables is more than 50% because away connection variables are fewer than home connection variables due to the maximum the maximum layover time limit for away connections. The % of reduction of connection variables is equal to the difference between the number of connection variables in Model I and that of in Model II divided by the former number. Moreover, the number of time-based RK constraints for away pools in Model II are only 5% of that of connection-based RK constraints for away pools in Model I. The percentage reduction of constraints is defined as the number of connection-based constraints minus the number of time-based constraints and divide by the number of connection-based constraints. And we remove

RK constraints for home pools in Model II. The % of reduction of RK constraint for away pools is equal to the difference between the number of connection based RK for away pools in Model I and the number of time based RK for away pools in Model II divided by the former number.

Table 2.2. Effectiveness of model enhancements (model reduction)

Instance	Model I				Model II	
	# connection variables	# total variables	# connection-based RK constraints for *home* pools	# connection-based RK constraints for *away* pools	% reduction of connection variables (remove home connection)	% reduction of RK constraint for *away* pools
M1	31665	33999	6848615	492762	66%	95%
M2	20309	22250	3273717	238365	68%	95%
M3	23777	25815	4544397	261015	68%	95%
M4	29667	31858	6133652	282523	68%	94%
M5	30748	33069	4864683	594349	61%	96%
M6	38583	41517	8324420	573039	65%	95%
M7	29176	31404	7221878	320038	70%	95%
L1	53781	59018	14449430	870171	63%	94%
L2	59424	65118	10430738	1891115	53%	97%
L3	62925	68631	16560103	1224765	62%	96%
L4	66814	72727	15546078	1784107	57%	96%
L5	46035	51031	10006353	614839	66%	94%
L6	55780	61167	15595988	674184	65%	92%
L7	45462	50572	11060264	616087	66%	94%
L8	61242	66865	17657164	898294	64%	94%
L9	58720	64051	12419859	1000814	60%	94%

Table 2.3 compares the performances of the Model I and Model II and verifies that model enhancements are effective. To solve the Model I, we have to first remove connection-based RK constraints for home pools, otherwise, the huge model size exceeds the memory limit in our PC for all instances. The final gap in Table 2.3 denotes the gap between the best integer objective and the objective of the best node remaining in branch and bound. While Model I without RK constraints for home pools is able to solve small and medium instances within 1 hour, it cannot find the optimal solution for L5 and L8

within 1% MIP gap. Moreover, it cannot find any feasible solution for instances (L2, L3, L4 and L9), which is shown as *NA* in Table 2.3. Model II outperforms Model I for all instances. Besides instance L2 and L3, it solves every instance within 1% MIP gap in 1 hour. Moreover, the Model II has a tighter lower bound than Model I. The % increment in LP bound for the Model II is equal to the difference between the initial LP bound of the Model I and Model II divided by the LP bound of Model I. The average increment is 0.19%.

Table 2.3. Computational results for Model I, Model II and Model III

Instance	Model I (without RK constraints for home pools)		Model II (Model reduction)			Model III (Model reduction and strengthening)			
	Final Gap	CPU time (sec)	Final Gap	CPU time (sec)	% increment in LP bound	Final Gap	CPU time (sec)	# new valid inequality	% increment in LP bound
M1	0.10%	526	1.00%	32	0.47%	0.88%	29	35851	1.41%
M2	0.50%	3765	0.00%	21	0.12%	0.92%	29	19205	0.30%
M3	0.86%	161	0.97%	14	0.48%	0.97%	24	20675	0.62%
M4	0.71%	303	0.39%	8	0.10%	0.06%	20	32982	0.59%
M5	0.51%	221	0.92%	10	0.02%	0.50%	30	39185	0.38%
M6	0.76%	1097	0.36%	52	0.01%	0.22%	60	45033	0.17%
M7	0.00%	137	0.13%	3	0.04%	0.00%	13	24959	0.14%
L1	0.97%	3753	0.02%	61	0.28%	0.42%	82	59713	0.44%
L2	*NA*		2.41%	3624	0.12%	0.57%	106	99563	0.22%
L3	*NA*		18.77%	3635	0.56%	0.00%	50	84542	1.66%
L4	*NA*		0.82%	57	0.15%	1.00%	145	118612	0.66%
L5	1.54%	4070	0.81%	47	0.00%	0.48%	34	41551	0.04%
L6	0.62%	1934	0.89%	239	0.08%	0.23%	176	58579	0.38%
L7	0.84%	741	0.00%	29	0.05%	0.83%	33	41888	0.24%
L8	4.29%	4216	0.90%	3392	0.44%	0.73%	369	79939	0.73%
L9	*NA*		0.91%	105	0.11%	0.06%	62	73387	0.34%

Although the model reduction is effective, the runtimes of Model II in Table 2.3 are still too long for the real-life crew assignments. In addition to the model reduction, we add constraints (2.13) to strengthen Model II. After model strengthening, the maximum

runtime among all instances decreases to 369 sec. Although on average over 50,000 valid inequalities are added to the optimization model, the extended model size is only about 10% of Model I (without RK constraints for home pools). The last column shows the % increment in LP bound for the Model III, which is the difference between the initial LP bound of the Model I and Model III divided by the LP bound of Model I. The average increment is 0.52%, which is much larger than the average increment in Model II (i.e., 0.19%).

Based on Table 2.3, Model III solves all instances within 1% MIP gap. But, the runtimes for large-scale instances L4, L6, and L8 exceed two minutes. So the goal of the RK-violation-repair heuristic method is to solve large instances quickly with near-optimal heuristic solutions.

Table 2.4 shows computational results for the RK-violation-repair heuristic algorithm and illustrates the effectiveness of the warm start method. The RK-violation-repair can solve all instances within 30 sec, which is a substantial improvement on runtime. The heuristic gap in the second column represents the difference between the heuristic solution and the optimal solution (within 1% MIP gap) of Model III divided by the objective of optimal solution. The negative heuristic gaps in the second column indicate that some heuristic solutions are even better than the optimal solution (within 1% MIP gap) of Model III. In the worst case, the heuristic solution for instance L8 is 1.2% larger than the optimal solution. The heuristic gap in the fourth column shows the difference between the first feasible solution in Model II and the optimal solution (within 1% MIP gap) for Model III divided by the objective of the optimal solution.

Table 2.4. RK-violation-repair heuristic and warm start

Instance	RK-violation repair heuristic		The first feasible solution in Model II		Warm start Model II with the heuristic solution	
	Heuristic gap	CPU time (Sec)	Heuristic gap	CPU time (sec)	Final gap	Total runtime (Sec)
M1	0.76%	10	0.15%	17	0.25%	21
M2	0.61%	7	2.28%	8	0.06%	64
M3	-0.13%	5	0.17%	11	0.85%	31
M4	0.04%	9	0.00%	9	0.45%	18
M5	-0.13%	11	0.00%	16	0.84%	24
M6	0.74%	12	108.35%	7	0.96%	30
M7	0.00%	5	0.00%	10	0.24%	17
L1	-0.25%	14	2.75%	23	0.79%	38
L2	-0.01%	18	0.00%	83	0.81%	57
L3	0.27%	18	0.00%	20	0.60%	53
L4	-0.58%	22	651.78%	108	0.91%	150
L5	-0.04%	18	0.00%	20	0.44%	97
L6	0.25%	16	0.03%	50	0.29%	123
L7	-0.41%	14	4.54%	24	0.26%	28
L8	1.20%	16	12.75%	78	0.41%	189
L9	0.63%	17	0.00%	52	0.37%	99

The heuristic solutions are used to warm start Model II because those near-optimal heuristic solutions are obtained quickly. To verify the effectiveness of the warm-start, in Table 2.4 we compare the heuristic solution with the first feasible solution in Model II without warm-start. Compared with first feasible solutions, heuristic solutions not only have smaller heuristic gaps but also require shorter runtime in large instances. With warm-start, the average runtime for hard instances (L1~L9) drops from 117 sec to 92 sec. Finally, Model II with warm start method outperforms Model I for every instance.

2.7.2 Sensitivity analysis

The goal of the following computational experiments is to study the effect of varying several important input features in the railway crew assignment problem and test the robustness of our optimization model. We perform this study on instance L7 which is a large instance with double occupations. In the optimal solution to this instance, there are

sufficient amount of deadhead trips so that the solution structure will change distinctly when input features vary. We run the computational tests using Model II.

In Table 2.5, we quantify how workload imbalance upper bounds affect the solution structure. Starting with a half of the default upper bounds, we increase the upper bounds by 50% of the default setting in each row. As the workload imbalance upper bounds increase, we observe that the number of crews in deadhead trips and the total cost decreases strictly. The runtime does not change significantly because the original runtime is only 33 seconds.

Table 2.6 discusses the effect of varying the maximum number of consecutive jobs (e.g. ρ) in the MCR requirement. When the ρ equals two, consecutively assigning three crews from one pool to outbound trips violates the MCR rule. Therefore, when the ρ increases, the MCR requirement becomes looser and more flexible and the total cost should decrease strictly. The results in Table 2.7 support our analysis. Moreover, the runtime is doubled and tripled when the ρ decreases to 4 and 3, which means the problem becomes much more difficult than the original one.

Table 2.5. Effect of changing the upper bound of workload imbalance

Upper bounds of workload imbalance	# taxi selected	# passenger train ticket	# crews in deadhead trip	Total layover hours	Total cost changes (%)	Runtime changes (%)
50%	4	8	42	169.1	8.3%	59.05%
100%	3	7	38	159.2	0.0%	0.00%
150%	3	2	32	188.5	-8.1%	21.90%
200%	2	2	28	196.4	-15.1%	-20.95%
250%	2	0	24	193.3	-21.7%	-3.81%

Table 2.6. Effect of changing the maximum consecutive jobs in MCR constraints

Max. consecutive jobs in MCR constraints	# taxi selected	# passenger train ticket	# crews in deadhead trip	Total layover hours	Total cost changes (%)	Runtime changes (%)
3	3	10	40	163.0	3.5%	346.15%
4	3	7	38	179.2	1.4%	273.08%
5	3	7	38	159.2	0.0%	0.00%
6	3	8	38	151.1	-0.4%	-7.69%
7	3	8	38	137.6	-1.4%	46.15%

Table 2.7 quantifies the effect of changing the per hour layover cost. If the layover becomes more expensive, the total layover hours will decrease. When the per hour layover cost is twice as much as the default setting, the total layover hours drops from 159.2 hours to 49.3 hours. The total cost increases because of the increasing per hour layover cost.

The taxi fixed cost is another critical input feature. Table 2.8 discusses the effect of varying the taxi fixed cost. In the first two rows, the taxi fixed cost divided by taxi capacity is cheaper than a passenger train ticket, so that deadheading prefer taxies rather than the passenger train. When the taxi fixed cost rises, the total layover time increases strictly, so does the total cost.

Table 2.7. Effect of changing the layover cost per hour

Layover costs per hour	# taxi selected	# passenger train ticket	# crews in deadhead trip	Total layover hours	Total cost changes (%)	Runtime changes (%)
50%	3	8	37	188.8	-5.8%	53.85%
100%	3	7	38	159.2	0.0%	0.00%
150%	3	11	42	76.6	3.8%	30.77%
200%	3	14	44	49.3	6.2%	23.08%
250%	4	10	44	44.6	7.9%	0.00%

Table 2.8. Effect of changing the fixed cost per taxi

Fixed costs per taxi	# taxi selected	# passenger train ticket	# crews in deadhead trip	Total layover hours	Total cost changes (%)	Runtime changes (%)
0%	8	0	40	109.5	-6.6%	15.38%
50%	5	0	38	151.4	-2.5%	34.62%
100%	3	7	38	159.2	0.0%	0.00%
150%	3	8	38	159.7	2.5%	46.15%
200%	2	12	38	182.2	4.2%	15.38%

2.8. Conclusion

In this chapter, we addressed the CSP in double-ended districts with workload balance requirements. Crew planners used to manually schedule crew assignments by following workload balance requirements. However, with an unbalanced traffic pattern, in which considerably more trains travel in one direction than in the reverse direction, balancing workloads among crew pools is a difficult task and it may need additional deadhead trips. To solve this problem to optimality, we formulated it as a network flow–based mixed-integer programming problem. Nevertheless, the basic optimization mode cannot solve medium or large instances quickly. To improve the optimization model, we reduced model size dramatically by providing a time-based RK formulation and strengthened the model by adding valid inequalities deliberately. Thus, large-scale instances are no longer computationally intractable. For the real-time crew scheduling environment, we developed effective solution techniques to provide optimal or near-optimal solutions within 30 seconds. We believe that our model and solution techniques can provide effective support to railway crew scheduling in real time.

This CSP assumes that trips can only depart at the planned departure time. However, delaying a planned departure time is allowed and used in practice. Delaying the departure time can potentially save layover cost because the delayed trip can send some just fully rested away crews back to their home station so that total layover cost or deadhead cost is reduced. In future research, delaying trips can be an option in the optimization model.

Chapter 3: Freight Railway Crew Scheduling with Uncertain Train Schedules

3.1. Introduction

In railway crew scheduling, crew planners need to assign the required number of crews to every regular train in a planning horizon. Unlike Chapter 2, in which future train arrival and departure times are assumed to be the same as the scheduled times, in this chapter, we discuss the freight railway crew scheduling problem while considering uncertainty in train arrivals and departures.

In this chapter, we solve the crew scheduling problem (CSP) in *single-ended* districts, where crews are based at one of two ending stations. We call the station where crews are homed as the *home* station and the other station as the *away* station. Therefore, there is only one crew pool for each occupation at each station, and crew planners do not need to choose a crew pool first to activate an engineer/conductor in single-ended districts. The only decision crew planners need to make is whether and when to *deadhead* crew members from one terminal to the other using deadhead trips. We consider two modes of deadhead trips: extra seats in scheduled trains (i.e., train deadhead) and taxi trips. Although deadheading crews results additional costs, it can cover crew shortages or reduce waiting time and cost at the away terminal.

One of the most important features in freight railway crew scheduling problem is rotation key rules. The RK rules govern the order of activating crews from a crew pool to departing trips. According to the FIFO rule, the crew with the earliest arrival time (i.e., end time of last trip) is activated first. According to the FOFO rule, the crew with the earliest departure time (i.e., start time of last trip) is activated first. In a single-ended district, either FIFO or FOFO rule applies. The RK rules automatically determine the sequence of trains/trips to which each crew is assigned given both scheduled trains and deadhead trips. In this chapter, we only consider the FIFO rule.

Freight railway crew scheduling is subject to uncertainties that keep railways from operating as per scheduled timetables. Uncertainties in train operations occur due to many factors such as weather, train condition, and railway traffic congestion. All of

these reasons prevent trains from traveling as scheduled. More importantly, railway disruptions in upstream districts have cascading impacts on downstream districts.

Without considering uncertainties in operations, current railway crew scheduling models perform poorly in practice. For example, optimal solutions of current deterministic models keep rest hours as short as possible because long rest hours at an away station increase layover costs. However, a crew assignment with short rest hours is more likely to be infeasible due to operational uncertainties (e.g., an upstream train arrives later than scheduled so that available rest hours for crews are inadequate). Therefore, the actual performance of a crew schedule can be significantly different from the planned performance.

In practice, crew planners would rather delay train departure than add deadhead trips when no crew is available if additional deadhead trips are expensive and the delay time is short. Although the option of delaying train departure time increases the difficulty of solving the problem, but it lowers the total cost. The model we propose in this chapter considers such options, which were not included in the model discussed in Chapter 1. We delay a ready-to-depart outbound train only when there is crew shortage.

The goal of this chapter is to support deadhead decisions and minimize the expected total cost in freight railway crew scheduling in single-ended districts. We first propose a stochastic programming (SP) model that considers uncertainty in train arrival and departure times. Although the SP model does not enforce FIFO constraints in the formulations, we prove that FIFO-satisfied solutions are the optimal solutions to the model because of the structure of the cost functions. The stochastic model minimizes the expected total cost including crew's waiting cost at the away station, train delay cost, and deadhead cost. Since the stochastic model requires a large number of scenarios to exactly model the uncertainty (e.g., more than 100 scenarios), solving the stochastic model even for small instances might be difficult. Moreover, we generate an analytical model to compute the expected total cost of a crew schedule, which is a set of selected deadhead trips. Given such analytical model, we can evaluate the real cost of the final solutions to the SP models. Finally, we develop four local search–based heuristic algorithms to

improve the deadhead decisions. The starting solution for the heuristic algorithms is the optimal solution to the SP model with few scenarios.

The main contributions of this chapter are as follows:

- We propose a stochastic programming model that incorporates uncertainty in train arrival and departure times. It minimizes the expected total cost and provides optimal deadhead decisions while considering the uncertainty.
- We develop an analytical model that evaluates the performance of a crew schedule in operations. Based on assumed distributions of train departure/arrival times, we compute the expected total cost of a deadhead plan without using simulation or a set of scenarios.
- We develop four efficient local search–based heuristic algorithms that improve deadhead decisions under uncertainty. Instead of calculating the expected cost for each crew schedule from scratch, our algorithms reuse previous calculation results and significantly reduce runtime.
- We conduct extensive computational experiments on randomly generated instances using the same cost components as those in crew assignment in a major North American railway company.

The remainder of this chapter is organized as follows. In Section 3.2, we review papers related to our problem. In Section 3.3, we provide a brief description of the general freight CSP in single-ended districts and describe the uncertainty studied in this chapter. Section 3.4 presents the stochastic programming model and analytical model that calculates the expected cost of a crew schedule. In Section 3.5, we present our solution methodologies including the four local search–based heuristic algorithms. In Section 3.6, we generate random instances, show computational results of the algorithms and present results of a sensitivity analysis for cost parameters. In Section 3.7, we present the conclusions of this chapter and provide suggestions for future research.

3.2. Literature review

To decrease operational crew cost, crew managers can use better recovery procedures in disruption management (see Jespersen-Groth et al. (2009), Potthoff et al. (2010), and

Veelenturf et al. (2012)). Another approach is to develop optimization models that incorporate operational uncertainties into crew scheduling decisions.

A large amount of literature on crew scheduling in the airline, railway, and bus transportation industries is available. However, we find little literature on improving the operational performance of crew scheduling decisions. The majority of the available literature focuses on airline crew scheduling.

Shebalov et al. (2006) improved the robustness of airline crew assignments by maximizing the number of move-up crews in crew schedule solutions. They defined a move-up crew as one that can take over the flight of another crew that should have arrived on time but is delayed because of disruptions. Therefore, maximizing the number of move-up crews is to maximize the opportunity of recovering flights by moving up crews (e.g. advancing their on-duty times). In their model, a crew can be moved up only if both crews are from the same crew base and have the same amount of remaining away days. Therefore, the tradeoff is between a robust schedule with more move-up crews and minimization of planned costs. They first solved the ordinary crew pairing problem without any disruption, thus obtaining the minimal planned cost. Then, they proposed an integer programming model to maximize the number of move-up crews while ensuring that the expected operational cost does not increase significantly over the planned cost. They solved the model using a combination of column generation and Lagrangian relaxation. The computational results indicate that more move-up crews lead to lower expected operational costs, but the trade-off between robustness and planned cost has to be considered judiciously by crew planners.

Filer et al. (2007) applied the concept of move-up crews to freight railway CSP. They based their work on Shebalov et al. (2006) but focused primarily on combinatorial problems encountered when maximizing number of move-up crews. Ignoring the covering requirement for each train, they studied combinatorial problems that either minimize cost with at least k move-up crews or maximize move-up crews with a maximum cost of l. Then, they transferred these problems into several variants of the

densest *k* subgraph problem. They used these results to develop a heuristic to the original problem, but they did not execute a computational experiment with the heuristic.

Sohoni et al. (2011) incorporated uncertainties associated with block-times into the process of designing airline schedules, which determine the departure and arrival times of each flight. A flight block-time is defined as the total time between the time a plane leaves its departure gate and arrives at its destination gate. To save on crew cost, which depends on total flight hours, airlines systematically allocate less flight block-time than expected. The authors aimed to balance the tradeoff between higher schedule reliability, which is achieved by increasing planned block-times, and lower planned profits. The schedule reliability represents the robustness of a schedule and is measured by the probability that flights arrive on time and the probability that passengers have adequate time to make their flight connections. The authors built a stochastic programming model that perturbs a proposed flight schedule. Then, they proposed a cut-generation algorithm to solve this stochastic programming model with chance constraints of schedule reliability. They not only proved the convergence of the cut-generation algorithm but also presented computational experiments using real-world data.

Rosenberger et al. (2000) presented a stochastic model that evaluates the performance of an airline crew schedule and recovery policies under disruptions. They implemented that model in a simulation software package called *SimAir*. Schaefer et al. (2005) discussed airline crew scheduling models that outperform deterministic models. They decomposed the operational cost of an entire crew schedule into operational costs of crew pairings in the schedule by assuming that crew pairings are independent. This method ignores the interaction between crew pairings when crews switch planes. Moreover, they included only the push-back recovery policy, which is not used often in practice for airline scheduling. In the push-back recovery procedure, a flight is delayed until all resources are available. Then, they estimated the operational cost of each pairing by means of simulation (i.e., SimAir) and solved a standard crew pairing model with the estimated pairing costs.

Yen and Birge (2006) proposed a two-stage stochastic binary programming model with recourse for an airline CSP in which they consider uncertainties in flight times and ground times. Apart from disruption costs due to a single crew's delay, they also identified disruption costs that occur only when crews are assigned to switch planes, defined as *Switch Delay* cost. To solve the two-stage stochastic programming model, they developed a *Flight-Pair Branching* algorithm that branches sequentially on the flight pair (e.g., two connecting flights) with the most expensive *Switch Delay*. At each node of a search tree, this algorithm solves the standard deterministic CSP and evaluates the operational cost of the solution in the recourse problem. After solving the recourse problem, the algorithm identifies the flight pair with most expensive Switch Delay and then branches on it by generating one node that excludes the flight pair and one node that allows the flight pair to appear in a pairing selected in the optimal solution. One drawback of this algorithm is that it is very computationally expensive, especially when the number of nodes grows exponentially, because the standard set-partitioning problem needs to be solved at each node. They only offered computational results on a small problem with a maximum of 79 flight pairs. Another drawback is that the algorithm converges slowly if the difference between planned cost and operational cost is large when it searches for an ϵ-optimal solution.

The problem in our study is very different from those discussed in the crew scheduling literature for airlines. First, compared with crew planning in airlines, our problem considers crew assignment in each district and has different cost components and operational rules. In freight railways, each crew member only travels within one district, which contains only two ending stations. Instead of considering a network of airports, our problem only considers crew assignment between two stations and the problem size is considerably smaller. The second fundamental difference is the RK rule in freight railways. In airlines, the order of activating pilots to departure flights at an airport is arbitrary so long as the pilots have adequate rest time. Shebalov et al. (2006) maximized the number of move-up crews in crew schedule solutions to improve the robustness of airline crew assignments. This approach is based on the fact that an

arriving pilot can connect with any departure flight as long as the pilot is fully rested. However, in freight railways, the RK rule determines which fully rested crew is activated for the next departing trip. The third difference is the deadheading trip decision in crew scheduling. Deadheading trips are undesired in some airline companies and not permitted in others. Most airline crew scheduling models use a set-partitioning formulation to ensure that each flight is assigned exactly one crew member. Some researchers use the set-covering formulation for simplification and then heuristically eliminate the deadhead trips in cases where more than one pilot is assigned to the same flight. However, our problem mainly aims to select deadhead trips rather than exactly assigning crew members to trains. Last, our solution methodologies are different. Apart from a stochastic programming model that optimizes deadhead decisions under uncertainty, we also developed an analytical model that computes the expected operational cost of a crew schedule. The results of our model are more accurate than an estimated cost based on a set of disruption scenarios or a limited number of iterations in simulation, especially when considering a large number of trains with high uncertainty in terms of arrival and departure times.

In the railway context, Gorman and Sarrafzadeh (2000) discussed a deterministic model and a dynamic programming–based heuristic to solve a basic CSP assuming that future train schedules in the input data are accurate. When supplied with inaccurate forecasts of train arrival and departure times, however, the deterministic model performs worse than well-experienced crew planners. To account for data uncertainty, they developed simple parametric approaches such as increasing the minimum rest time, keeping safety stock of crews, and assuming worst-case train arrival and departure scenarios. However, these approaches generally create high-cost solutions. They suggested that a methodology based on expected costs under uncertainty is needed to overcome the shortcomings of the deterministic model.

To the best of our knowledge, Si and Balakrishnan (2016) is the first and only study in railway crew scheduling literature that formally models uncertainty in train arrival and departure times. To optimize deadhead decisions under uncertainty, they developed a

recursive algorithm and two heuristic algorithms that use an ordering principle similar to that used for solving the newsvendor problem. Although their methods provide useful insights for crew planning under train schedule uncertainty, they cannot support practical crew scheduling decisions in real time for the following reasons. First, their model approximates the cost of layover hours using a linear function instead of the exact cost function. Second, their model does not consider all deadhead options such as train deadhead trip and taxi deadhead trip because crew members on these trips have the same arrival and departure times. Third, their recursive algorithm can only solve problems with a small number of deadhead candidate trips. In this chapter, we aim to generate crew scheduling solutions that support crew scheduling decisions in practice. Therefore, cost components in our model are the same as the cost functions in real operations. We propose a stochastic programming model that minimizes expected total cost and optimizes deadhead decisions under uncertainty. To solve large-scale instances, we developed an analytical model that computes the expected total cost of a crew schedule solution. This model can handle correlations between crews on the same trip. Therefore, it considers all deadhead options that crew planners can use. Moreover, it formulates the cost of each crew connection, which is not considered in Si and Balakrishnan (2016). To improve deadhead decisions under uncertainty, we developed four local search–based heuristic algorithms that provide local-optimal solutions in minutes.

3.3. Crew scheduling and uncertainty

Section 3.3.1 describes the general freight railway CSP in single-end districts. In Section 3.3.2, we discuss the uncertainty in train arrival and departure times and the impact of this uncertainty on crew assignments and costs.

3.3.1. Railway crew scheduling in single-ended districts

A freight railroad network is geographically divided into districts, and crew planners manage crew assignment within each district. Single- and double-ended are the two most common types of crew management in railway districts. Because we introduced the features of crew management in a double-ended district in Chapter 1, we focus on single-

ended districts in this chapter. In this section, we describe the freight railway CSP in single-ended districts.

A crew pool comprises crew members of the same occupation and based at the same terminal. An away crew pool is one based at terminal *A* but is currently away from terminal *A*. In single-ended districts, crews are based at only one ending terminal, called the home terminal, and the opposite terminal, denoted as the away terminal, has only away crew pools. We denote trains traveling from the home terminal to the away terminal as *inbound* trains and the trains traveling in the reverse direction as *outbound* trains. Extra crews at the home terminal are always available when no regular crew is fully rested to operate scheduled inbound trains. Using a new extra crew causes a large penalty. The new extra crew member will leave the district after returning to his/her home terminal.

In both home and away terminals, crew members need to rest at least 11.5 hours (*Minimum rest hours*) to be considered fully rested (i.e., available) for the next trip, as required by FRA regulations. When assigning an under-rested crew to a departing trip, the trip's departure time needs to be postponed. The company imposes a *maximum layover time* limit to forbid crews from waiting more than 24 hours at the away terminal. In Chapter 2, it is a hard constraint for connection hours. However, we make it to be a soft constraint in this chapter. The violation penalty is a per hour cost for layover time exceeding 24 hours.

Deadheading is unproductively repositioning crew from one terminal to the other. However, deadheading crews can cover crew shortages or reduce waiting costs. In both single- and double-ended districts, there are three modes of deadhead trips: passenger train, extra seats in a scheduled train (i.e., train deadhead), and taxi trip. Details are given in Chapter 1. However, we only consider train deadhead and taxi trips in this chapter because our models assume that regular train and deadhead trips can be delayed due to crew shortage. But crew dispatchers in freight railways cannot postpone a passenger train's departure time.

In the planning stage, the cost components include *trip rate*, *layover cost*, *max layover time penalty*, and *deadhead costs*. Trip rate is the cost of assigning a crew member to either a scheduled train or a deadhead trip. Waiting at crew members' home terminal is free. However, if a crew's waiting hours at the away terminal exceed the layover threshold (e.g., 16 h), a per-hour *layover cost* is charged for the time over and above the threshold. The company needs to pay a ticket fare for each crew on a passenger train. Freight train deadheads are free but offer limited capacity. Railways have to pay a fixed cost for each taxi that can carry up to five crew members. Therefore, *deadhead costs* include ticket fares in passenger trains and the fixed cost of taxis. The company pays a trip rate for each crew member in a deadhead trip.

An additional cost component called *trip delay cost* is charged by delaying train departure times. In operations, arrival times often go off schedule because of various reasons, which may cause a temporary shortage of crews at the arriving terminal. Furthermore, crew shortage may postpone a few trains' departure times even though the trains are ready to depart. The cost of delaying a trip is a per hour cost, regardless of the number of crews in the trip.

Identical to the crew requirement constraint in double-ended districts, crew requirements for each scheduled train must be satisfied in single-ended districts. In addition, crew management in single-ended districts must follow RK rules. RK rules govern the order of activating (or call on duty) crews from a crew pool to departing trips. According to the FIFO rule, crews are activated in a first-in first-out manner from each crew pool. Hence, the earliest-arriving crew has the highest *RK priority* and leaves the crew pool first. According to the FOFO rule, crews are assigned to trains in the same order they were activated in the last assignment before arriving into the current crew pool. Then, the earliest departing crew has the highest *RK priority* and leaves the crew pool first. Details about RK rules are given in Chapter 1.

The optimal solution of a deterministic model assigns every crew to a sequence of scheduled trains or deadhead trips and minimizes total costs while satisfying the following constraints: 1) Each regular train is assigned the required numbers of engineers

and conductors. 2) Each crew is rested adequately before a new trip, and the number of rest hours is within the maximum layover time limit if the crew is at the away terminal; 3) A crew is assigned to a trip only if it has the highest RK priority among all available crews when the trip departs.

Since arrival or departure times may deviate from the plan, the RK priorities of crews may differ from those in the planned solution. In the next section, we introduce a crew-assignment procedure that ensures the above crew assignment constraints are met when arrival and departure times change.

3.3.2. Uncertainty and its consequences

Our goal is to calculate the expected total cost at the away terminal under the FIFO rule. We focus on uncertainties in arrival times of inbound trains and ready-to-depart times of outbound trains because these uncertainties determine crew assignments at the away terminal under the FIFO rule. The ready-to-depart time of an outbound train is actually the train's arrival time at the away station from upstream terminals. The train is ready to depart from the away station to the home station except required crew members.

After crew planners determine deadhead trips in a planning horizon, a crew-assignment procedure ensures that crew assignments always follow operational rules. Under the FIFO rule, the procedure always assigns the highest FIFO priority crew in a crew pool to the current departing train if there is at least one available crew. Else, the procedure delays the train until a crew is available. In this section, we describe how this procedure is applied and how uncertainty in operations changes the planned cost.

An inbound train may arrive at the away terminal later than the planned arrival time such that the crew in the inbound train is no longer the highest-RK-priority crew when the outbound train to which it was assigned departs. Figure 3.1 (a) shows the scheduled arrival times and original crew connections ($<i_1, j_1>, <i_2, j_2>$). In this scenario, the crew in trip i_1 should be the highest-RK-priority (FIFO) crew when trip j_1 departs. Figure 3.1 (b) shows an example in which inbound train i_1 arrives late, so the crew in i_1 has lower FIFO priority than the crew in i_2. In such a case, the original crew connections violate the FIFO rule. Hence, the recovery procedure *swaps* two arriving crews and assigns the

crew in trip i_2 to trip j_1. Finally, the procedure generates new crew connections ($<i_2, j_1>$, $<i_1, j_2>$). The cost of the original crew connections $<i_1, j_1>$ and $<i_2, j_2>$ may differ from that of the new crew connections $<i_2, j_1>$ and $<i_1, j_2>$.

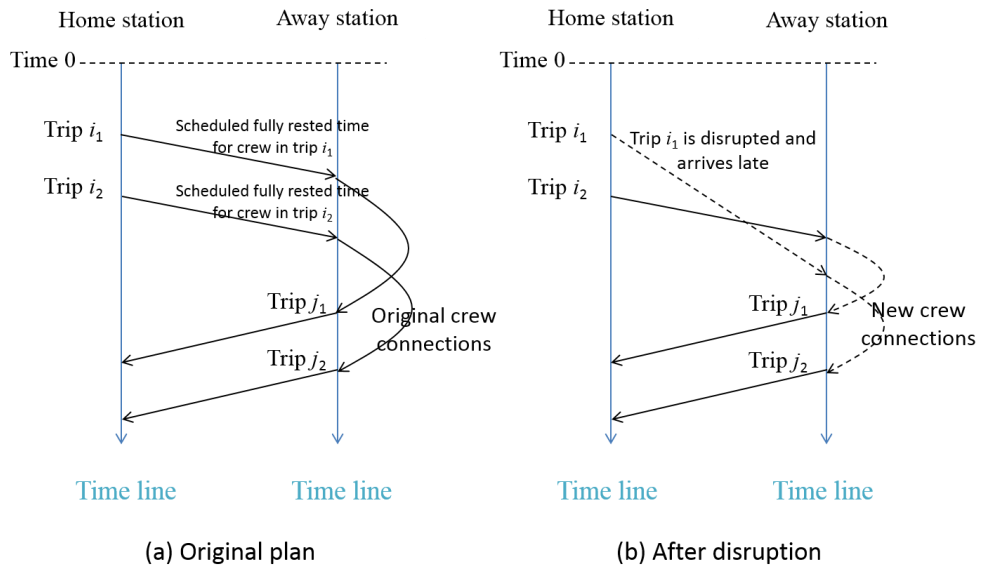


Figure 3.1. Crew swapping under FIFO rule

Shebalov et al. (2006) introduced a similar concept called *Move-Up crews* for recovery procedures in airline crew scheduling, where no RK rule is required. They move up a crew to an earlier departing flight only because one original crew is delayed and is does not have adequate rest hours for the scheduled flight. Then, they swap crews if other operational rules are satisfied. While they can move up an arbitrary arriving crew as long as the moved-up crew is fully rested when the new flight departs, RK rules in freight railways automatically determine which crew to move up.

The other possible consequence of the lateness of an inbound train is that the outbound train that takes the arriving crew in the schedule may have to wait for the arriving crew and get delayed in turn. Similar to the example in Figure 3.2, no crew is available when outbound trip j_1 is ready to depart because the inbound trip i_1 arrives later than scheduled. If no fully rested crew is available, the procedure delays the outbound train until a crew is fully rested (e.g., crew in trip i_1 in Figure 3.2). Delaying an outbound train leads to a per hour penalty depending on the type/priority of the delayed train.

It is very possible that more than one outbound trip is delayed due to crew shortage. When crew members become fully rested and available, the order of dispatching the delayed trips follows the ready-to-depart order. Regardless of trip type and priority, the delayed trip with earlier ready-to-depart time is dispatched earlier when there are available crews. We restate this assumption in the next section.

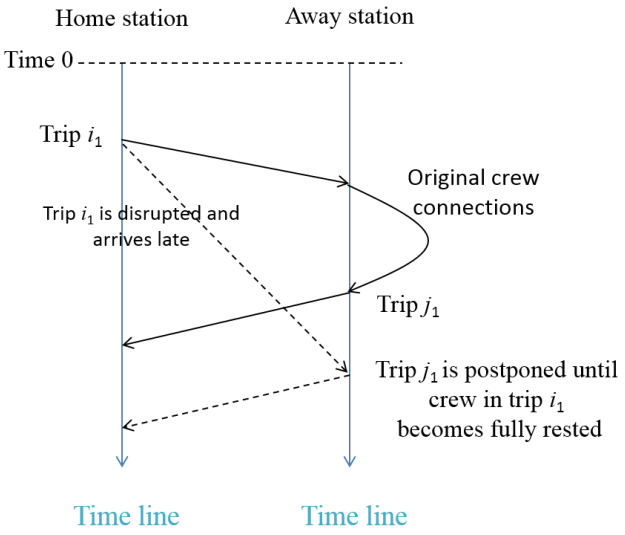


Figure 3.2. Delaying a trip

If an inbound train arrives late but the crew in the train still has sufficient rest hours to connect with the scheduled outbound train and the RK orders of the arriving crews do not change, the layover cost of this away connection can be lower than then planned value.

Similarly, an outbound train may be ready to depart from the away terminal later than the scheduled departure time because the train is delayed at upstream terminals. When it is ready to depart from the away terminal, the procedure simply assigns the highest-RK-priority crew to this train if such a crew is available. If the procedure assigns the crew in the original plan to this outbound train, the crew waits longer at the away terminal than scheduled and its layover cost might increase. If the procedure does not assign the planned crew to this outbound train, it swaps crews to ensure RK rules are followed. In the worst case, no crew is available when the outbound train is ready to depart from the away terminal; thus, the outbound train has to be delayed.

It is also possible that an outbound train is ready to depart before the scheduled departure time or an inbound train arrives at the away terminal earlier than the scheduled arrival time. In these cases, the total layover costs at the away terminal also differ from the planned cost.

3.4. Model description

In Section 3.4.1, we present three assumptions made to simplify the problem, followed by a stochastic programming model that formulates uncertainty and optimizes deadhead decisions in Section 3.4.2. Because solving the stochastic programming model is very difficult, especially for large-scale instances, an analytical model for determining expected total cost of each feasible solution is desired. We first develop a formulation for determining the probability of a crew connection according to the FIFO rule and then develop formulations for expected crew connection cost and expected trip delay cost in Section 3.4.3.

3.4.1. Assumptions

To simplify the problem and construct a model that measures the expected cost of a crew schedule with a given number of deadhead trips, we make the following assumptions.

- Assumption 1. The ready-to-depart and arrival times at the away station of each train are independent with known distributions.

Although this assumption is not generally true in practice (e.g., weather disruptions), it significantly simplifies our problem. Based on historical data and crew planners' experiences, we can estimate the distributions of arrival times of inbound trains and ready-to-depart times of outbound trains at the away station.

The arrival time distributions are obtained by adding the planned arrival times and random variations that depend on train type/priority. If a train has higher priority, the mean and standard deviation of the random variations are smaller. Similarly, we define the distributions of ready-to-depart times of outbound trips by adding the planned ready-to-depart times and variations.

Notably, the planning horizon is divided into equal time periods such that the distributions of arrival times or ready-to-depart times are discretized into sets of discrete

probabilities. For example, the arrival time distributions of an inbound train are discretized into a set of probabilities that the inbound train arrives in several time periods.

- Assumption 2. There are a sufficient number of available regular crews at the home terminal.

Based on this assumption, no extra crew is needed at the home terminal. More importantly, trains departing from the home terminal will never be delayed, even though trains might arrive at the home terminal later than scheduled. Therefore, the crew availability and assignment at the home station would not impact the trip delay penalty, layover cost and max layover time violation penalty at the away terminal.

- Assumption 3. Delayed trips are released in ascending order of ready-to-depart time.

Regardless of trip type and priority, the delayed trip with earlier ready-to-depart time is dispatched earlier when crew members become fully rested. This is a critical assumption when more than one outbound trip is delayed due to crew shortage.

3.4.2. Stochastic programming (SP) model

This stochastic programming model formally formulates uncertainty in arrival times of inbound trains and ready-to-depart times of outbound trains. It only considers single occupation. Besides Assumptions 1 and 2, the SP model also use Assumption 3 because delaying outbound trips due to crew shortage is allowed. There is a drawback in the SP model. In some extreme cases, this formulation underestimates the expected crew connection cost. We provide an example after the SP model formulation.

If we can solve this model, we can optimize deadhead decisions under the uncertainty. We first define notations and then present the formulation, followed by the validity of the model. We prove that the optimal solution to this SP model satisfies the FIFO rule even though the FIFO rule is not enforced in the formulation. Finally, we discuss solution methods of the SP model and the reason why the L-shaped method or the Bender's decomposition is not applicable.

Notation

i index of inbound trips that belong to inbound trip set IN, $i \in IN$

j index of outbound trips that belong to outbound trip set OUT, $j \in OUT$

TR set of regular train

DR set of train deadhead candidate trip

TX set of taxi deadhead candidate trip

h index of deadhead trip, $h \in TX \bigcup DR$

C_h deadhead capacity of deadhead trip h. A train deadhead trip's capacity is one and the deadhead capacity for each taxi deadhead trip is five.

RT fully rested hours at the away station (i.e., 11.5 hours)

HT layover hour beginning threshold (i.e., 16 hours)

FT max away from home time limit (i.e., 24 hours)

CT trip rate for deadheading each crew member

CD per hour trip delay cost for delaying an outbound trip that is ready to depart

CF taxi fixed setup cost

CH per hour layover cost for waiting hours exceeding HT

CM per hour violation penalty for waiting hours exceeding FT

$PP(j)$ set of possible immediate predecessor trips of outbound trip j. There is no restriction on the waiting hours between predecessor and trip j because max layover time and minimal rest hours are soft constraints and penalized.

$PS(i)$ set of possible immediate successor trips of inbound trip i. There is no restriction on the waiting hours between trip i and its successor.

$j(h)$ the regular train that has deadhead capacity and produces train deadhead trip h

ω index of scenarios representing random vector that contains random arrival times of all inbound regular trains and random ready-to-depart times of all outbound regular trains. It belongs to sample space Ω, $\omega \in \Omega$

at_i^{ω} arrival time of trip i in scenario ω

dt_j^{ω} ready-to-depart time of trip j in scenario ω

c_{ij}^{ω} crew connection cost from trip i to trip j in scenario ω. It includes layover cost and max layover time penalty.

$$c_{ij}^{\omega} = CH \max\{0, dt_j^{\omega} - at_i^{\omega} - HT\} + CM \max\{0, dt_j^{\omega} - at_i^{\omega} - FT\}$$

Decision variables

x_{ij}^{ω} = 1 if inbound trip i connects with outbound trip j in scenario ω; 0 otherwise

z_j^{ω} non-negative continuous variable representing delay hours of trip j in scenario ω

y_h = 1 if taxi deadhead candidate trip h is selected; 0 otherwise

v_h non-negative integer variable representing the number of crews in deadhead trip h

SP Model

$$\text{Minimize} \quad E_{\omega}\left\{\sum_{i\in IN}\sum_{j\in PS(i)} c_{ij}^{\omega} x_{ij}^{\omega} + \sum_{j\in OUT\setminus DR} CD\, z_j^{\omega}\right\} + \sum_{h\in TX\cup DR} CT\, v_h + \sum_{h\in TX} CF\, y_h \tag{3.1}$$

Subject to

$$\sum_{j\in PS(i)} x_{ij}^{\omega} = 1, \qquad \forall i \in IN \cap TR, \omega \in \Omega \tag{3.2}$$

$$\sum_{i\in PP(j)} x_{ij}^{\omega} = 1, \qquad \forall j \in OUT \cap TR, \omega \in \Omega \tag{3.3}$$

$$\sum_{j\in PS(h)} x_{hj}^{\omega} = v_h, \qquad \forall h \in IN \cap \{DR \cup TX\}, \omega \in \Omega \tag{3.4}$$

$$\sum_{i\in PP(h)} x_{ih}^{\omega} = v_h, \qquad \forall h \in OUT \cap \{DR \cup TX\}, \omega \in \Omega \tag{3.5}$$

$$v_h \le C_h, \qquad \forall h \in DR \tag{3.6}$$

$$v_h \le C_h y_h, \qquad \forall h \in TX \tag{3.7}$$

$$(at_i^{\omega} + RT - dt_j^{\omega}) x_{ij}^{\omega} \le z_j^{\omega}, \qquad \forall i \in IN, j \in PS(i), \omega \in \Omega \tag{3.8}$$

$$z_h^{\omega} \le z_{j(h)}^{\omega} \qquad \forall h \in DR \cap OUT, \omega \in \Omega \tag{3.9}$$

$$x_{ij}^{\omega}, y_h \in \{0,1\},\ z_j^{\omega} \ge 0,\ v_h \in \{0,1,...\} \quad \forall i \in IN, j \in PS(i), h \in DR \cup TX, \omega \in \Omega \tag{3.10}$$

The objective function in (3.1) is to minimize the expected crew connection cost and trip delay cost plus constant trip rate in deadhead trips and taxi trip fixed setup cost.

Notice that the trip delay cost of a train deadhead trip is not counted in the objective function because its corresponding regular train's delay cost is already included. Constraints (3.2) ensure that each inbound regular trip connects with one successor in any scenario and constraints (3.3) ensure that each outbound regular train connects with one predecessor in any scenario. Constraints (3.4) and (3.5) count the number of crews in each deadhead trip in any scenario and enforce that number to be the same for all scenarios. The deadhead capacity of train deadhead trip is enforced in constraints (3.6) and deadhead capacity of taxi candidate trip is ensured in constraints (3.7). Constraints (3.8) are the forcing constraints for trip delay hours for each outbound trip and each scenario. We create this constraint only when $at_i^\omega + RT - dt_j^\omega$ is positive for connection from i to j and scenario ω. Since a train deadhead trip departs at the same time as its corresponding regular train, constraints (3.9) enforces this relationship. Finally, decision variables are defined in constraints (3.10).

This model defines the trip delay hour variable so that it considers the case that an inbound trip arrives later than scheduled time and an outbound trip to which the inbound crew is assigned has to be delayed until the crew gets fully rested.

In some extreme cases, this formulation underestimates the expected crew connection cost. Suppose two crew connect to a multi-job outbound trip. If the first crew has waited more than 16 hours and results layover cost when the outbound trip j is ready to depart, but the second crew is not fully rested yet and needs to delay the outbound trip, then the layover hours of the first connection is extended by a number of delaying hours. However, our formulation does not consider the possible increment of layover hours.

This model only considers the FIFO rotation key rule and does not enforce this rule specifically in constraints. However, the convex crew connection cost function (i.e., the definition of parameters c_{ij}^ω) and trip delay cost function (i.e., constraints 3.8) guarantee that an optimal solution can be rearranged to satisfy FIFO rule without increasing total cost. The following lemmas and proposition describe this claim. Vaidyanathan et al. (2007) considered a special case of Lemma 1. They have shown that the FIFO satisfied solution is no more expensive than FIFO unsatisfied solutions when the crew connection

cost is a linear function of waiting hours at the away station. However, Lemma 1 considers general convex functions of the crew connection cost.

Let trip i_1 and i_2 be inbound trips and the trip i_1 arrives earlier than the trip i_2. Suppose the scheduled departure time of outbound trip j_1 is earlier than that of outbound trip j_2.

Lemma 1. If the crew connection cost function is a convex function of connection hours (i.e., crew waiting hours), then the total crew connection cost from i_1 to j_1 and from i_2 to j_2 is not more than that from i_1 to j_2 and from i_2 to j_1.

Proof. We denote the convex crew connection cost as $f(t)$, where t is connection hours. Let t_{ij} be the connection time from trip i to trip j. So we have connection hours $t_{i_1,j_1}, t_{i_1,j_2}, t_{i_2,j_1}$, and t_{i_2,j_2}. Our goal is to prove $f(t_{i_1,j_1}) + f(t_{i_2,j_2}) \le f(t_{i_1,j_2}) + f(t_{i_2,j_1})$.

Since $t_{i_2,j_1} \le t_{i_1,j_1} \le t_{i_1,j_2}$, we can find a $\lambda_0 \in [0,1]$ such that $\lambda_0 t_{i_2,j_1} + (1-\lambda_0) t_{i_1,j_2} = t_{i_1,j_1}$, which is denoted as equation (1). We define inequality (2) $\lambda_0 f(t_{i_2,j_1}) + (1-\lambda_0) f(t_{i_1,j_2}) \ge f(\lambda_0 t_{i_2,j_1} + (1-\lambda_0) t_{i_1,j_2})$ because of the convexity of $f(t)$. Similarly, Since $t_{i_2,j_1} \le t_{i_2,j_2} \le t_{i_1,j_2}$, we can find a $\lambda_1 \in [0,1]$ such that $\lambda_1 t_{i_2,j_1} + (1-\lambda_1) t_{i_1,j_2} = t_{i_2,j_2}$, which is denoted as equation (3). For the same reason, we define inequality (4) $\lambda_1 f(t_{i_2,j_1}) + (1-\lambda_1) f(t_{i_1,j_2}) \ge f(\lambda_1 t_{i_2,j_1} + (1-\lambda_1) t_{i_1,j_2})$. Since $t_{i_1,j_1} + t_{i_2,j_2} = t_{i_1,j_2} + t_{i_2,j_1}$, $\lambda_0 + \lambda_1 = \frac{t_{i_1,j_1} - t_{i_1,j_2}}{t_{i_2,j_1} - t_{i_1,j_2}} + \frac{t_{i_2,j_2} - t_{i_1,j_2}}{t_{i_2,j_1} - t_{i_1,j_2}} = \frac{t_{i_1,j_1} + t_{i_2,j_2} - 2t_{i_1,j_2}}{t_{i_2,j_1} - t_{i_1,j_2}} = \frac{t_{i_2,j_1} - t_{i_1,j_2}}{t_{i_2,j_1} - t_{i_1,j_2}} = 1$. Finally, we add (2) with (4),

$f(t_{i_2,j_1}) + f(t_{i_1,j_2}) \ge f(\lambda_0 t_{i_2,j_1} + (1-\lambda_0) t_{i_1,j_2}) + f(\lambda_1 t_{i_2,j_1} + (1-\lambda_1) t_{i_1,j_2}) = f(t_{i_1,j_1}) + f(t_{i_2,j_2})$. ■

This lemma shows that for any pair of inbound trips it is always no more expensive to activate the first arriving crew first, which is required by the FIFO rule. Moreover, the convex crew connection cost function and the trip delay cost function guarantees that there exits an optimal solution which satisfies the FIFO rule. The following lemma and proposition support this claim.

Let i_1, i_2, i_3 and i_4 be single-job inbound trips. Let j_1 and j_2 denote outbound trips that have two crew requirements in each trip. Suppose $at_{i_1} \le at_{i_2} \le at_{i_3} \le at_{i_4}$ and $dt_{j_1} \le dt_{j_2}$.

Trip delay hour for an outbound trip is the maximum of shortages of rest hours for the two crews who are assigned to this outbound trip. Trip delay cost is the trip delay hours times a per hour penalty. Based on Assumption 3, delayed trips are released in ascending order of ready-to-depart time.

Lemma 2. If the crew connection cost function is a convex function of connection hours (i.e., crew waiting hours), then the crew schedule solution that assigns the first two crews to trip j_1 and the last two crews to trip j_2 minimizes the total crew connection costs and trip delay costs.

Proof. We denote the convex crew connection cost as $f(t)$, where t is connection hours. Let t_{ij} be the connection time from trip i to trip j and $t_{ij} = dt_j - at_i$. Suppose crew i_1 and i_2 are assigned to trip j_1; then the trip delay cost for trip j_1, denoted as $d(t_{i_1 j_1}, t_{i_2 j_1})$, is equivalent to $\max\{RT + at_{i_1} - dt_{j_1}, RT + at_{i_2} - dt_{j_1}, 0\}$. Notice that we omit the per hour penalty parameter in the delay cost function to simplify our notation. Our goal is to prove that $f(t_{i_1, j_1}) + f(t_{i_2, j_1}) + f(t_{i_3, j_2}) + f(t_{i_4, j_2}) + d(t_{i_1 j_1}, t_{i_2 j_1}) + d(t_{i_3 j_2}, t_{i_4 j_2})$ is not larger than the total crew connection costs and trip delay costs of any other crew assignment solution.

There are six different ways of assigning two inbound crews to each outbound trip. Let the above crew assignment solution be the first one, denoted as *Case* 0. In each of the following 5 cases, we will show that the crew assignment solution in Case 0 is no worse than any other solution. Figure 3.3 demonstrates the solution in each case.

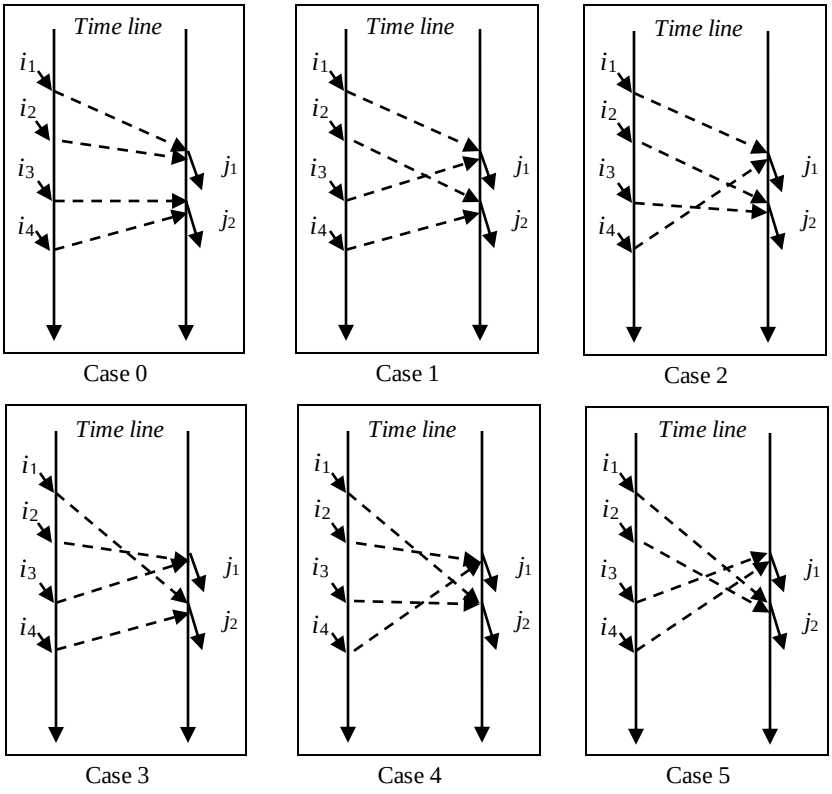


Figure 3.3. All possible solutions to the example in Proposition 3

Case 1

In this case, crew i_1 and i_3 are assigned to j_1 while crew i_2 and i_4 are assigned to j_2. We want to show that $f(t_{i_1,j_1}) + f(t_{i_2,j_1}) + f(t_{i_3,j_2}) + f(t_{i_4,j_2}) + d(t_{i_1j_1}, t_{i_2j_1}) + d(t_{i_3j_2}, t_{i_4j_2})$ is not greater than $f(t_{i_1,j_1}) + f(t_{i_3,j_1}) + f(t_{i_2,j_2}) + f(t_{i_4,j_2}) + d(t_{i_1j_1}, t_{i_3j_1}) + d(t_{i_2j_2}, t_{i_4j_2})$. Because of Lemma 1, $f(t_{i_2,j_1}) + f(t_{i_3,j_2}) \le f(t_{i_3,j_1}) + f(t_{i_2,j_2})$. We only need to show that $d(t_{i_1j_1}, t_{i_2j_1}) + d(t_{i_3j_2}, t_{i_4j_2}) \le d(t_{i_1j_1}, t_{i_3j_1}) + d(t_{i_2j_2}, t_{i_4j_2})$.

$\max\{RT + at_{i_1} - dt_{j_1}, RT + at_{i_2} - dt_{j_1}, 0\} \le \max\{RT + at_{i_1} - dt_{j_1}, RT + at_{i_3} - dt_{j_1}, 0\}$ because $at_{i_2} \le at_{i_3}$. Since $at_{i_2} \le at_{i_3} \le at_{i_4}$, $\max\{RT + at_{i_3} - dt_{j_2}, RT + at_{i_4} - dt_{j_2}, 0\}$ equals to $\max\{RT + at_{i_2} - dt_{j_2}, RT + at_{i_4} - dt_{j_2}, 0\}$. In sum, $d(t_{i_1j_1}, t_{i_2j_1}) + d(t_{i_3j_2}, t_{i_4j_2}) \le d(t_{i_1j_1}, t_{i_3j_1}) + d(t_{i_2j_2}, t_{i_4j_2})$.

Case 2

In this case, crew i_1 and i_4 are assigned to j_1 while crew i_2 and i_3 are assigned to j_2. We want to show that $f(t_{i_1,j_1}) + f(t_{i_3,j_1}) + f(t_{i_2,j_2}) + f(t_{i_4,j_2}) + d(t_{i_1j_1}, t_{i_3j_1}) + d(t_{i_2j_2}, t_{i_4j_2})$ is not greater than $f(t_{i_1,j_1}) + f(t_{i_2,j_2}) + f(t_{i_3,j_2}) + f(t_{i_4,j_1}) + d(t_{i_1j_1}, t_{i_4j_1}) + d(t_{i_2j_2}, t_{i_3j_2})$. If the solution in this case is worse than the solution in Case 1, then solution in Case 0 is better than this solution. Because of Lemma 1, $f(t_{i_3,j_1}) + f(t_{i_4,j_2}) \le f(t_{i_3,j_2}) + f(t_{i_4,j_1})$. We only need to show that $d(t_{i_1j_1}, t_{i_3j_1}) + d(t_{i_2j_2}, t_{i_4j_2}) \le d(t_{i_1j_1}, t_{i_4j_1}) + d(t_{i_2j_2}, t_{i_3j_2})$.

Because of $at_{i_1} \le at_{i_3}$, $d(t_{i_1j_1}, t_{i_3j_1}) = \max\{RT + at_{i_1} - dt_{j_1}, RT + at_{i_3} - dt_{j_1}, 0\} = \max\{RT + at_{i_3} - dt_{j_1}, 0\}$. Similarly, $d(t_{i_2j_2}, t_{i_4j_2}) = \max\{RT + at_{i_2} - dt_{j_2}, RT + at_{i_4} - dt_{j_2}, 0\} = \max\{RT + at_{i_4} - dt_{j_2}, 0\}$. On the right hand side, $\max\{RT + at_{i_1} - dt_{j_1}, RT + at_{i_4} - dt_{j_1}, 0\} = \max\{RT + at_{i_4} - dt_{j_1}, 0\}$. $\max\{RT + at_{i_2} - dt_{j_2}, RT + at_{i_3} - dt_{j_2}, 0\} = \max\{RT + at_{i_3} - dt_{j_2}, 0\}$. If we denote $g(t_{ij})$ as $\max\{RT - t_{ij}, 0\} = \max\{RT + at_i - dt_j, 0\}$, what we want to prove is $g(t_{i_3j_1}) + g(t_{i_4j_2}) \le g(t_{i_3j_2}) + g(t_{i_4j_1})$. Since $g(x)$ is a convex function of x, Lemma 1 proves this inequality.

Case 3

In this case, crew i_2 and i_3 are assigned to j_1 while crew i_1 and i_4 are assigned to j_2. Like the Case 2, we want to show that this solution is no better than the solution in Case 1. We want to show that $f(t_{i_1,j_1}) + f(t_{i_3,j_1}) + f(t_{i_2,j_2}) + f(t_{i_4,j_2}) + d(t_{i_1j_1}, t_{i_3j_1}) + d(t_{i_2j_2}, t_{i_4j_2})$ is not greater than $f(t_{i_1,j_2}) + f(t_{i_2,j_1}) + f(t_{i_3,j_1}) + f(t_{i_4,j_2}) + d(t_{i_2j_1}, t_{i_3j_1}) + d(t_{i_1j_2}, t_{i_4j_2})$. Because of Lemma 1, $f(t_{i_1,j_1}) + f(t_{i_2,j_2}) \le f(t_{i_1,j_2}) + f(t_{i_2,j_1})$. We only need to show that $d(t_{i_1j_1}, t_{i_3j_1}) + d(t_{i_2j_2}, t_{i_4j_2}) \le d(t_{i_2j_1}, t_{i_3j_1}) + d(t_{i_1j_2}, t_{i_4j_2})$.

Because $at_{i_1} \le at_{i_2} \le at_{i_3} \le at_{i_4}$, the left hand size is transformed to $\max\{RT + at_{i_3} - dt_{j_1}, 0\} + \max\{RT + at_{i_4} - dt_{j_2}, 0\}$ and the right hand is also $\max\{RT + at_{i_3} - dt_{j_1}, 0\} + \max\{RT + at_{i_4} - dt_{j_2}, 0\}$. So these two solution have the same total trip delay costs.

Case 4

In this case, crew i_2 and i_4 are assigned to j_1 while crew i_1 and i_3 are assigned to j_2. We want to show that this solution is no better than the solution in Case 3 thus it is no better than the solution in Case 0. We want to show that $f(t_{i_1,j_2}) + f(t_{i_2,j_1}) + f(t_{i_3,j_1}) + f(t_{i_4,j_2}) + d(t_{i_2j_1}, t_{i_3j_1}) + d(t_{i_1j_2}, t_{i_4j_2})$ is not greater than $f(t_{i_1,j_2}) + f(t_{i_2,j_1}) + f(t_{i_3,j_2}) + f(t_{i_4,j_1}) + d(t_{i_2j_1}, t_{i_4j_1}) + d(t_{i_1j_2}, t_{i_3j_2})$. Based on Lemma 1, $f(t_{i_3,j_1}) + f(t_{i_4,j_2}) \leq f(t_{i_3,j_2}) + f(t_{i_4,j_1})$. We only need to show that $d(t_{i_2j_1}, t_{i_3j_1}) + d(t_{i_1j_2}, t_{i_4j_2}) \leq d(t_{i_2j_1}, t_{i_4j_1}) + d(t_{i_1j_2}, t_{i_3j_2})$.

Like the proof in the Case 2, the left side of the inequality is $g(t_{i_3j_1}) + g(t_{i_4j_2})$ where the right hand side is transformed to be $g(t_{i_4j_1}) + g(t_{i_3j_2})$. Since $g(x)$ is a convex function of x, Lemma 1 proves that $g(t_{i_3j_1}) + g(t_{i_4j_2}) \leq g(t_{i_4j_1}) + g(t_{i_3j_2})$.

Case 5

In this case, crew i_1 and i_2 are assigned to j_2 while crew i_3 and i_4 are assigned to j_1. We want to show that this solution is no better than the solution in Case 0. Our goal is to show that $f(t_{i_1,j_1}) + f(t_{i_2,j_1}) + f(t_{i_3,j_2}) + f(t_{i_4,j_2}) + d(t_{i_1j_1}, t_{i_2j_1}) + d(t_{i_3j_2}, t_{i_4j_2})$ is not greater than $f(t_{i_1,j_2}) + f(t_{i_2,j_2}) + f(t_{i_3,j_1}) + f(t_{i_4,j_1}) + d(t_{i_3j_1}, t_{i_4j_1}) + d(t_{i_1j_2}, t_{i_2j_2})$. Based on Lemma 1, $f(t_{i_1,j_1}) + f(t_{i_3,j_2}) \leq f(t_{i_1,j_2}) + f(t_{i_3,j_1})$ and $f(t_{i_2,j_1}) + f(t_{i_4,j_2}) \leq f(t_{i_2,j_2}) + f(t_{i_4,j_1})$. We only need to show that $d(t_{i_1j_1}, t_{i_2j_1}) + d(t_{i_3j_2}, t_{i_4j_2}) \leq d(t_{i_3j_1}, t_{i_4j_1}) + d(t_{i_1j_2}, t_{i_2j_2})$.

Because $at_{i_1} \leq at_{i_2} \leq at_{i_3} \leq at_{i_4}$, the left hand side is transformed to $\max\{RT + at_{i_2} - dt_{j_1}, 0\} + \max\{RT + at_{i_4} - dt_{j_2}, 0\}$ and the right hand is also $\max\{RT + at_{i_4} - dt_{j_1}, 0\} + \max\{RT + at_{i_2} - dt_{j_2}, 0\}$. According to the definition of convex function $g(x)$, the left hand side is $g(t_{i_2j_1}) + g(t_{i_4j_2})$ while the right hand side is equivalent with $g(t_{i_4j_1}) + g(t_{i_2j_2})$. The left hand side is less than or equal to the right hand side because of Lemma 1.

In summary, we have shown that the solution in Case 0 is no worse than solutions in other cases in terms of the total crew connection cost and trip delay cost. ■

It is worth mentioning that the potential increment of crew connection cost due to delaying a multi-job outbound trip does not change the fact that FIFO satisfied solutions are no more expensive than the FIFO unsatisfied solutions. The reason is that both the FIFO satisfied solution and the FIFO unsatisfied solution ignore that cost increment.

This proposition shows that for any pair of outbound trips it is always no more expensive to activate the first arriving crew first, which is required by the FIFO rule. The total cost includes crew connection costs and trip delay costs. In order to demonstrate that the SP model does not need to explicitly formulate FIFO constraints, we propose the following proposition.

Define trip delay hour of an outbound trip as the maximum shortage of rest hours for crews who are assigned to this outbound trip. Trip delay cost is the trip delay hour times a positive per hour penalty. This proposition is also based on Assumption 2 and 3.

Proposition 2. If the crew connection cost function is a convex function of connection hours, then at least one optimal solution of the SP model satisfies the FIFO rule.

Proof. We prove this by contradiction like the proof of Proposition 1. Given a deadhead plan, the crew assignment in each scenario does not impact each other. Therefore, the crew assignment in every scenario satisfies the FIFO rule is a necessary and sufficient condition that the overall solution satisfies the FIFO rule. Suppose none optimal solution satisfies the FIFO rule. For each scenario, we choose one of the optimal crew assignments and check if the crew assignment for every two outbound trips satisfies the FIFO rule. If it violates the FIFO rule, such as assignments in the Case 1 to Case 5 in the proof of Lemma 2, then we switch the assignment to the solution in the Case 0 so that FIFO rule is satisfied. According to Lemma 2, such switch does not increase the total connection costs and delay costs. If the outbound trips are not multi-job trips, it is a special case of the example in Lemma 2. The claim of Lemma 2 still holds. If the outbound trips are multi-job trips with more than two crew requirements, we can consolidate crew requirements in outbound trips and crews in inbound trips such that the problem degenerate into the cases in Lemma 2.

If the total cost of the new solution decreases, then the original solution is not an optimal solution and it contradicts with the assumption. If the total cost of the new solution remains the same, the new solution is also an optimal solution. We repeat this procedure until the crew assignment for every two outbound trips satisfies the FIFO rule. Therefore, the crew assignment in the final solution of this scenario satisfies the FIFO rule. We repeat this for every scenario. If the overall final solution has the same total cost, it contradicts with the assumption that none of the multiple optimal solutions satisfies the FIFO rule. ■

For any deadhead plan, this model formulates corresponding crew connections in each scenario and estimates the expected total connection cost by averaging the total connection costs in all scenarios. To obtain an accurate estimation, the stochastic programming model needs to incorporate at least hundreds of scenarios. Moreover, this number increases as problem size grows and the uncertainty of train arrival and departure time increases.

The L-shaped method or Benders' decomposition is widely used to solve very large linear programming problems that have special block structure. Although our stochastic programming model has that special block structure, we cannot follow that approach because of the integrality constraints for connection variables in each scenario. Relaxing the integrality constraints for the connection variables in the recourse problem would lead to fractional variable solutions that violate FIFO rules and have cheaper total cost than that of the optimal solution of the original problem.

In the computational results part, we use CPLEX to solve the stochastic programming models with various numbers of scenarios. The results indicate that solving the stochastic model for large instances with accurate estimation is very difficult. Therefore, we propose local search based heuristic algorithms to provide heuristic solutions quickly. The next two sections present the analytic model that calculates the expected total cost of a crew schedule, which is determined by a selected set of deadhead trips. The analytic model is used in the heuristic algorithms to evaluate each solution searched in the heuristics.

3.4.3. Analytical Model

The goal of the analytic model is to calculate the expected total cost when a deadhead plan is given. Our analytic model also focuses on the FIFO rotation key rule and considers single occupation. This analytical model is based on Assumption 1, 2 and 3. In this section, we first derive the formulation of the probability of a crew connection. And then we develop a formulation of the expected total cost. Like the SP model, the analytical model does not count the potential crew connection cost increment when a multi-job trip is delayed.

The total cost consists of crew layover cost, max layover time violation penalty, trip delay cost and deadhead cost. An inbound trip may contain multiple crews and an outbound trip may have multiple crew requirements (e.g., outbound taxi trip). We define *crew requirement* to distinguish jobs in inbound trips and outbound trips. Note that crews indicate inbound trips and crew requirements correspond to outbound trips.

A *crew connection* denotes when a crew arrives at the away station and when he is assigned to an outbound crew requirement. Recall that the layover cost is a linear function of waiting hours exceeding a certain threshold and the maximum layover time violation penalty incurs when waiting hours exceed 24 hours. Therefore, a crew connection indicates both layover cost and max layover time violation penalty. We denote the sum of them as *crew connection cost*. If an outbound trip is ready to depart but there was not enough crew members available, then the outbound trip has to be held until enough crews are available. Trip delay cost is a linear function of trip delay hour for each delayed trip.

Besides a constant deadhead cost for a deadhead plan, the expected total cost includes the summation of expected crew connection cost of each inbound crew plus the summation of expected trip delay cost of each outbound trip. The expected crew connection cost of an inbound crew is the sum, over all possible arrival times t_1 and ready-to-depart times t_2 of crew requirements, of the probability of the connection starting from t_1 to t_2 multiplied by the connection cost. Unlike the layover cost and the max layover time violation penalty, trip delay cost is charged for each delayed outbound

trip which may contain multiple crew requirements. If an outbound trip j contains only one crew requirement, the expected trip delay cost is the sum of the probability of any connection starting from an inbound crew at time t_1 and ending at the crew requirement in outbound trip j at time t_2 multiplied by the corresponding delay cost over all possible ready-to-depart time t_2, inbound crew and its arrival time t_1. If the outbound trip j contains more than one crew requirement, the delayed hours of the trip j is determined by the last arriving crew that connects with trip j.

In sum, the key issue is to formulate the probability of a connection that starts from a crew at time t_1 and ends at a crew requirement at time t_2.

The probability that an inbound crew arriving at time t_1 connects with an outbound crew requirement that is ready to depart at time t_2 involves all of the previous inbound crews and outbound trips because this crew has to be the highest-RK-priority crew among all available crews when the outbound crew requirement departs. However, calculating this probability by enumerating all combinations of inbound trip arrival times and outbound trip departure times is very difficult. Alternatively, our formulation considers inbound crews and outbound crew requirements separately and thus significantly reduces the complexity.

Before introducing the formulation, we need to define several more concepts. An inbound trip may contain multiple crews that have the same arrival time. Similarly, an outbound trip may have multiple crew requirements. When an outbound trip is ready to depart, all crew requirements in the same trip are ready to depart at the same time. If a trip contains only one crew (also called job in freight railways), then we denote this trip as a single-job trip. Otherwise, we denote the trip as a multi-job trip. We discretize the planning horizon into time periods (e.g., 3 minutes) so that our model approximates the continuous case.

The *arrive-at-away order* of inbound trips and crews is determined by arrival times, where earlier arrivals have precedence over later arrivals. If two inbound trips arrive at the same time period, we sort them in increasing order of their trip IDs and assign them different arrive-at-away orders. We denote this order as *ID order*. A higher ID order

corresponds to a lower ID. If a trip i has higher ID order than a trip i', then we denote $i \succ i'$. We give each trip a unique ID.

For a multi-job inbound trip, we sort crews in increasing order of crew IDs and assign them different arrive-at-away orders. If a crew r has higher ID order than a crew r', then we denote $r \succ r'$. We also give each crew a unique ID. Notice that arrive-at-away order of a trip equals to the arrive-at-away order of the last crew in the trip, where crews are sorted in increasing order of crew IDs.

Figure 3.4 shows an example of arrive-at-away orders of inbound trips and crews. In this example, two multi-job trips arrive at the same time period, but we need to assign unique arrive-at-away order to each crew. We first sort two trips according to their trip IDs and then sort crews within each trip based on their crew IDs. Let AO_m denote the arrive-at-away order of crew m and AO_i denote the arrive-at-away order of trip i. Since T001 $\succ$ T002, trip i_2 has a higher order than trip i_1. Moreover, crew C01 has higher arrive-at-away order than crew C02, therefore $AO_{C01} = 1$ and $AO_{C02} = AO_{T001} = 2$. On other hand, crew C04 has higher arrive-at-away order than crew C05, so $AO_{C04} = 3$ and $AO_{C05} = AO_{T002} = 4$. The crew C01 in trip i_2 has the highest arrive-at-away order.

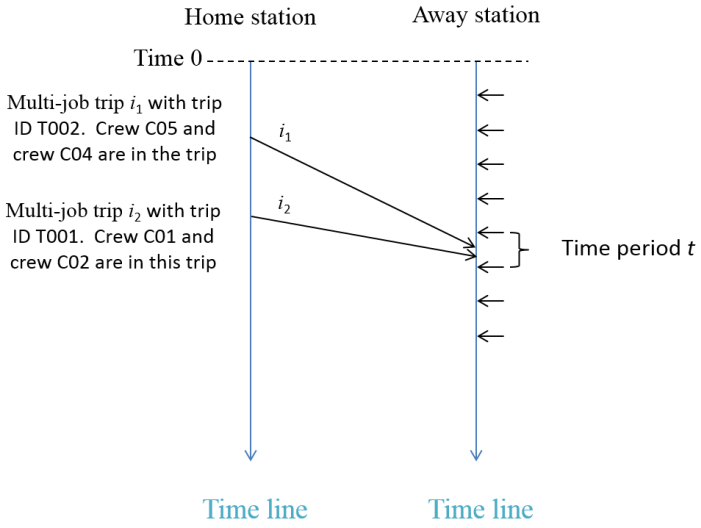


Figure 3.4. Arrive-at-away orders of crews and trips

The *depart-from-away order* of outbound trip and crew requirements is determined by ready-to-depart times, where earlier ready-to-depart times have precedence over later

ones. It is worth mentioning that the ready-to-depart time may be different with real departure time because an outbound regular train or deadhead trip can be delayed because of crew shortage. If two outbound trips are ready to depart at the same time period, we sort them according to their trip IDs. For a multi-job outbound trip, we sort crew requirements in the increasing order of crew requirement IDs. We give each crew requirement a unique crew requirement ID. Similar to the arrive-at-away order, we define that the depart-from-away order of an outbound trip equals to the depart-from-away order of the last crew requirement in the trip, which is sorted by crew requirement IDs.

We find that under the FIFO rule an inbound crew connects with an outbound crew requirement if and only if the arrive-at-away order of the crew equals to the depart-from-away order of the crew requirement. This condition also indicates the reason why we need to assign each crew unique arrive-at-away order and assign each crew requirement unique depart-from-away order.

We define the following notations to demonstrate the formulation.

m index of crew member

$M(i)$ set of inbound crews that belong to the trip i

i_m index of the inbound trip that carries inbound crew m

M set of inbound crews. $M = \bigcup_{i \in IN} M(i)$

$\Gamma(I)$ number of crews in an inbound trip set I. $\Gamma(I) = \sum_{i \in I} |M(i)|$

r index of crew requirement

$R(j)$ set of crew requirements in outbound trip j

j_r index of the outbound trip that contains the crew requirement r

R set of outbound crew requirements. $R = \bigcup_{j \in OUT} R(j)$

$\Gamma(J)$ number of crew requirements in an outbound trip set J. $\Gamma(J) = \sum_{j \in J} |R(j)|$

t index of time period

T set of all time periods

AT_m random arrival time of inbound crew m

AT_i random arrival time of inbound trip i

AO_m the arrive-at-away order of inbound crew $m \in M$

AO_i the arrive-at-away order of inbound trip $i \in IN$.

$$AO_i = \max_{m \in M(i)} \{AO_m\}$$

RT_r random ready-to-depart time of crew requirement r

RT_j random ready-to-depart time of outbound trip j

RO_r the depart-from-away order of crew requirement $r \in R$

RO_j the depart-from-away order of outbound trip $j \in OUT$.

$$RO_j = \max_{r \in R(j)} \{RO_r\}$$

k index of arrive-at-away order or depart-from-away order. $k \in K$

We define the probability of a crew connection that starts from a crew m at time t_1 and ends at an outbound crew requirement r at time t_2 as: $\Pr(AT_m = t_1, RT_r = t_2, m \text{ connects to } r)$. Based on the observation, this probability is equal to the probability that the arrive-at-away order of the crew with arrival time t_1 equals to the depart-from-away order of the crew requirement with ready-to-depart time t_2.

$$\Pr(AT_m = t_1, RT_r = t_2, m \text{ connects to } r) = \Pr(AT_m = t_1, RT_r = t_2, AO_m = RO_r)$$

$$= \sum_{k \in K} \Pr(AT_m = t_1, RT_r = t_2, AO_m = k, RO_r = k). \quad (3.11)$$

According to Assumption 1, every trip is independent of others and thus equation (3.11) is equivalent to $\sum_{k \in K} \Pr(AT_m = t_1, AO_m = k) \Pr(RT_r = t_2, RO_r = k)$

$$= \sum_{k \in K} \Pr(AO_m = k \mid AT_m = t_1) \Pr(AT_m = t_1) \Pr(RO_r = k \mid RT_r = t_2) \Pr(RT_r = t_2) \quad (3.12)$$

Since $\Pr(AT_m = t_1)$ and $\Pr(RT_r = t_2)$ can be calculated from discretizing the presumed distribution of arrival times/ready-to-depart times, the remaining challenge is to calculate $\Pr(AO_m = k \mid AT_m = t_1)$ and $\Pr(RO_r = k \mid RT_r = t_2)$. To formulate these two conditional probabilities, we need to define the following notations.

α_{it} the probability that inbound trip i arrives before or at time period t

$u_{i'it}$ the probability that the arrive-at-away order of inbound trip i' is higher than that order of inbound trip i given that trip i arrives at time t.

If $i' \succ i$, then trip i' still has higher arrive-at-away order than trip i when they arrive at the same time period. Thus $u_{i'it} = \alpha_{i't}$ if $i' \succ i$, otherwise $u_{i'it} = \alpha_{i't-1}$.

$I(i, t)$ set of inbound trips that may have higher arrive-at-away arriving order than that of inbound trip i given that trip i arrives at time t.

β_{jt} the probability that outbound trip j is ready to depart before or at time period t.

$v_{j'jt}$ the probability that the depart-from-away order of outbound trip j' is higher than trip j given that trip j is ready to depart at time t. If $j' \succ j$, then $v_{j'jt} = \beta_{j't}$, otherwise $v_{j'jt} = \beta_{j't-1}$.

$J(j, t)$ set of outbound trips that may have higher depart-from-away order than that of outbound trip j given that trip j is ready to depart at time t.

δ_{il} index of the l-th crew in crew set $M(i)$ of trip i, where $l \in \{1, 2, 3, ..., |M(i)|\}$

Notice that the l-th crew in $M(i)$ has the l-th crew ID order in the trip i.

γ_{jl} index of the l-th crew requirement in $R(j)$, where $l \in \{1, 2, 3, ..., |R(j)|\}$

Note that the l-th crew requirement in $R(j)$ has the l-th crew requirement ID order in trip j.

When the random arrival time of a crew m is realized, arrival times of other crews in the same trip are also determined. Suppose there are $|M(i_m)|$ crews (including crew m) in the inbound trip i_m, which contains crew m. If the crew m is the l-th crew in the trip i_m, then $(l-1)$ of crews in the same trip has higher order than the crew m and $|M(i_m)| - l$ of crews have lower order than the crew m. According to the definition of AO_i, the arrive-at-away order of the trip i is equal to the arrive-at-away order of the last crew in the trip i. In sum, the event that the arrive-at-away order of the l-th crew in trip i equals to k is equivalent to the event that the arrive-at-away order of the trip i is $k + |M(i)| - l$. Therefore $\Pr(AO_m = k \mid AT_m = t_1) = \Pr(AO_{i_m} = k + |M(i_m)| - l \mid AT_{i_m} = t_1)$, where $\forall m \in M$ such that crew m is the l-th crew in trip i_m (i.e., $m = \delta_{i_m l}$).

To calculate $\Pr(AO_i = k \mid AT_i = t_1)$ for every k, which is the probability mass function of arrive-at-away order of inbound trips, we only need to know the probability that there are $k - |M(i)|$ crews that have higher arrive-at-away order than trip i given that trip i arrives at time t_1. The total number of crews that have higher arrive-at-away order than trip i is a sum of independent random variables that are not identically distributed. Each random variable represents if an inbound trip has higher arrive-at-away order than trip i given that trip i arrives at time t_1. If inbound trip i' is a single-job trip, then its random variable is either 0 or 1. If inbound trip i' has M crews, then its random variable is either 0 or M.

$$\Pr(AO_i = k \mid AT_i = t_1) = \sum_{I \subseteq I(i,t_1), \Gamma(I) = k - |M(i)|} \left[\prod_{i' \in I} u_{i' i t_1} \prod_{i' \in I(i,t_1) \setminus I} (1 - u_{i' i t_1}) \right] \quad (3.13)$$

The equation (3.13) enumerates all possible trip sets in which a set of trips has higher arrive-at-away order than trip i and the number of crews in the trip set equals to $k - |M(i)|$. The probability that such a set of trips I is chosen from $I(i,t)$ is the product of the probability that trips in I actually have higher order and the probability that trips in $I(i,t) \setminus I$ actually have lower order.

Similarly, the event that the depart-from-away order of the l-th crew requirement in trip j equals to k is equivalent with the event that the depart-from-away order of the trip j is $k + |R(j)| - l$.

Therefore $\Pr(RO_r = k \mid RT_r = t_2) = \Pr\left(RO_{j_r} = k + |R(j_r)| - l \,\middle|\, RT_{j_r} = t_2 \right)$, where $\forall r \in R$ such that r is the l-th crew requirement in outbound trip j_r (i.e., $r = \gamma_{j_r l}$). To calculate $\Pr(RO_j = k \mid RT_j = t_2)$, we only need to know the probability that there are $k - |R(j)|$ crews that have higher depart-from-away order than that order of trip j given that trip j is ready to depart at time t_2.

$$\Pr(RO_j = k \mid RT_j = t_2) = \sum_{J \subseteq J(j,t_2), \Gamma(J) = k - |R(j)|} \left[\prod_{j' \in J} v_{j' j t_2} \prod_{j' \in J(j,t_2) \setminus J} (1 - v_{j' j t_2}) \right] \quad (3.14)$$

The equation (3.14) enumerates all possible trip sets in which a set of outbound trips has higher depart-from-away order than trip j and the number of crew requirements in the trip set equals to $k - |R(j)|$.

In sum, we can calculate the probability of each connection that starts from a crew m ($m = \delta_{i_m l}$) at time t_1 and ends at an outbound crew requirement r ($r = \gamma_{j_r l'}$) at time t_2. We start from the definition of a crew connection and use equations (3.11)-(3.14) to derive equation (3.15).

$$\Pr(AT_m = t_1, RT_r = t_2, m \text{ connects to } r)$$

$$\text{, where } \forall m \in M \text{ such that } m = \delta_{i_m l}, \forall r \in R \text{ such that } r = \gamma_{j_r l'}, t_1, t_2.$$

$$= \Pr(AT_m = t_1)\Pr(RT_r = t_2)\sum_{k \in K}\Pr(AO_m = k \mid AT_m = t_1)\Pr(RO_r = k \mid RT_r = t_2)$$

$$\text{, where } \forall m \in M \text{ such that } m = \delta_{i_m l}, \forall r \in R \text{ such that } r = \gamma_{j_r l'}, t_1, t_2.$$

$$= \Pr(AT_m = t_1)\Pr(RT_r = t_2)$$
$$*\sum_{k \in K}\Pr(AO_{i_m} = k+ \mid M(i_m) \mid -l \mid AT_{i_m} = t_1)\Pr(RO_{j_r} = k+ \mid R(j_r) \mid -l' \mid RT_{j_r} = t_2)$$

$$\text{, where } \forall m \in M \text{ such that } m = \delta_{i_m l}, \forall r \in R \text{ such that } r = \gamma_{j_r l'}, t_1, t_2.$$

$$= \Pr(AT_m = t_1)\Pr(RT_r = t_2)$$
$$*\sum_{k \in K}\{\sum_{I \subseteq I(i_m, t_1), \Gamma(I) = k-l}[\prod_{i' \in I} u_{i' i_m t_1}\prod_{i' \in I(i_m, t_1) \setminus I}(1 - u_{i' i_m t_1})]\sum_{J \subseteq J(j_r, t_2), \Gamma(J) = k-l'}[\prod_{j' \in J} v_{j' j_r t_2}\prod_{j' \in J(j_r, t_2) \setminus J}(1 - v_{j' j_r t_2})]\}$$

$$\text{, where } \forall m \in M \text{ such that } m = \delta_{i_m l}, \forall r \in R \text{ such that } r = \gamma_{j_r l'}, t_1, t_2. \qquad (3.15)$$

Similarly, we calculate the probability of each crew connection that starts from a crew m ($m = \delta_{i_m l}$) at time t_1 and ends at the last crew requirement in outbound trip j at time t_2. If trip j is a multi-job trip with $|R(j)|$ crew requirements, then there are $|R(j)|$ crew connections that end at trip j. Among those $|R(j)|$ crew connections, only the one that ends at the last crew requirement of trip j determines the delay hours of the outbound trip j. In this case, the arrive-at-away order of the crew m equals to depart-from-away order of outbound trip j. In sum, we only need to focus on this type of crew connections when calculating trip delay cost. We start with defining the crew connections and conduct equation (3.16) because of assuming independence.

$\Pr(AT_m = t_1, RT_j = t_2, m \text{ connects to the last } r \text{ in } j)$, where $\forall m \in M$ such that $m = \delta_{i_m l}$, $\forall j \in OUT$, t_1, t_2

$= \sum_{k \in K} \Pr(AO_m = k, AT_m = t_1, RO_j = k, RT_j = t_2)$, where $\forall m \in M$ such that $m = \delta_{i_m l}$, $\forall j \in OUT$, t_1, t_2

$= \Pr(AT_m = t_1) \Pr(RT_j = t_2) \sum_{k \in K} \Pr(AO_m = k \mid AT_m = t_1) \Pr(RO_j = k \mid RT_j = t_2)$

, where $\forall m \in M$ such that $m = \delta_{i_m l}$, $\forall j \in OUT$, t_1, t_2 (3.16)

Like the argument we made in equations (3.11)-(3.15), the probability of a connection ending at the last crew requirement at trip j is equivalent with equation (3.17).

$= \Pr(AT_m = t_1) \Pr(RT_j = t_2)$

$\sum_{k \in K} \Pr(AO_{i_m} = k + \mid M(i_m) \mid - l \mid AT_{i_m} = t_1) \Pr(RO_j = k \mid RT_j = t_2)$

, where $\forall m \in M$ such that $m = \delta_{i_m l}$, $\forall j \in OUT$, t_1, t_2.

$= \Pr(AT_m = t_1) \Pr(RT_j = t_2)$

$\sum_{k \in K} \{ \sum_{I \subseteq I(i_m, t_1), \Gamma(I) = k - l} [\prod_{i' \in I} u_{i' i_m t_1} \prod_{i' \in I(i_m, t_1) \setminus I} (1 - u_{i' i_m t_1})] \sum_{J \subseteq J(j, t_2), \Gamma(J) = k - \mid R(j) \mid} [\prod_{j' \in J} v_{j' j t_2} \prod_{j' \in J(j, t_2) \setminus J} (1 - v_{j' j t_2})] \}$

, where $\forall m \in M$ such that $m = \delta_{i_m l}$, $\forall j \in OUT$, t_1, t_2. (3.17)

Besides the constant deadhead cost of a deadhead plan, the total expected cost consists of the summation of the expected crew connection costs over all inbound crews and the summation of expected trip delay costs for all outbound trips. The expected crew connection cost of an inbound crew is the sum of the probability of any connection starting from the inbound crew at time t_1 and ending at an outbound crew requirement r at time t_2 multiplied by its connection cost over all possible arrival time t_1, outbound crew requirement r and its ready-to-depart time t_2. The expected trip delay cost of an outbound trip is the sum of probability of any connection starting from any crew m at time t_1 and ending at the last crew requirement in the outbound trip at time t_2 multiplied by the corresponding trip delay cost over all possible connections. Recall that only this type of crew connections determines delay hours of the outbound trip.

We define the following notations.

$TA(m)$ set of possible time periods for the random arrival time of inbound crew m

$TD(j)$ set of possible time periods for the random ready-to-depart time of trip j

$TD(r)$ set of possible time periods for the random ready-to-depart time of crew requirement r

c_{m,t_1,r,t_2} crew connection cost for a connection starts from the inbound crew m at time t_1 and ends at an outbound crew requirement r at time t_2

d_{m,t_1,j,t_2} trip delay cost for a connection starts from the inbound crew m at time t_1 and ends at the last crew requirement in an outbound trip j at time t_2

Total expected crew connection cost

$$= \sum_{m \in M} \sum_{t_1 \in TA(m)} \sum_{r \in R} \sum_{t_2 \in TD(r)} c_{m,t_1,r,t_2} \Pr(AT_m = t_1, RT_r = t_2, m \text{ connects to } r) \quad (3.18)$$

Total expected trip delay cost

$$= \sum_{j \in OUT} \sum_{t_2 \in TD(j)} \sum_{m \in M} \sum_{t_1 \in TA(m)} d_{m,t_1,j,t_2} \Pr(AT_m = t_1, RT_j = t_2, m \text{ connects to the last } r \text{ in } j) \quad (3.19)$$

Equation (3.15) formulates the probability in equation (3.18) and equation (3.17) is equivalent to the probability in the equation (3.19). In sum, the total expected cost is modeled based on the presumed distributions of arrival times of inbound trips and ready-to-depart times of outbound trips.

The same as the stochastic programming model, this formulation underestimates the expected connection cost. Suppose two crew connect to a multi-job outbound trip. If the first crew has waited more than 16 hours and results layover cost when the outbound trip j is ready to depart, but the second crew is not fully rested yet and needs to delay the outbound trip, then the layover hours of the first connection is extended by the amount of delaying hours. However, our formulation of expected connection cost does not consider the possible extension of layover hours.

3.5. Solution methodology

In this section, we describe our methodologies for solving the railway CSP under uncertainty. We develop an *evaluator* that implements the formulation of the expected total cost of a crew schedule given deadhead trips. The evaluator only works for the FIFO rotation key rule. The method of calculating total expected cost for any feasible solution and the computational complexity are shown in Section 3.5.1. We develop a

greedy local search algorithm, Tabu search algorithm, Simulated Annealing algorithm and Genetic algorithm to improve crew schedule solutions. We present these local and global search algorithms in Section 3.5.2 – 3.5.5. Instead of calculating the expected cost from scratch for each crew schedule solution, our algorithms reuse previous calculation results and significantly reduce runtime in these search algorithms. This update strategy is presented in Section 3.5.6.

3.5.1. Calculate total expected cost

To calculate equations (3.18) and (3.19), we need to first compute the $\Pr(AT_m = t_1, RT_j = t_2, m \text{ connects } j)$ for any inbound crew m, time period t_1, outbound crew requirement r, and time period t_2, and compute $\Pr(AT_m = t_1, RT_j = t_2, m \text{ connects } j)$ for any inbound crew m, time period t_1, outbound trip j, and time period t_2. According to equations (3.16) and (3.17), we need to first calculate the conditional probabilities $\Pr(AO_i = k \mid AT_i = t_1)$ and $\Pr(RO_j = k \mid RT_j = t_2)$, which are computed by equations (3.15) and (3.16) respectively. Basically, we need to calculate the probability that the sum of N independent random variables which are not necessarily identically distributed equals to k. While computing $\Pr(AO_i = k \mid AT_i = t_1)$, each random variable represents if an inbound trip i' has higher arrive-at-away order than trip i given i arrives at time t_1. The outcome of each random variable is zero or non-negative integer $M(i')$, which is the number of crews in inbound trip i'. The same analysis applies to outbound trips. Therefore, computing the conditional probabilities $\Pr(AO_i = k \mid AT_i = t_1)$ and $\Pr(RO_j = k \mid RT_j = t_2)$ efficiently is the key of our algorithms, especially when problem size is large or the planning horizon is long. If we compute those probabilities quickly, we can approximate the continuous case more accurately by dividing the planning horizon into more time periods.

Conditioning on a trip i's arrival time, the distribution of its arrive-at-away order is an extension to the Poisson Binomial distribution if other inbound trips' arrival time distributions are given. A typical Poisson Binomial distribution is a discrete probability distribution of a sum of independent Bernoulli trials that are not necessarily identically distributed. And the outcome of the Bernoulli trials are zero or one. However, the

outcome of each trail in our case is zero or non-negative integer number. The sum of these trails follows a *Weighted Poisson Binomial* distribution. Fortunately we found that algorithms for computing the Poisson Binomial distribution can be extended to solve our problem.

Barlow and Heidtmann (1984) propose a recursive formula and describe the algorithm in the form of a BASIC program. Chen and Liu (1997) provide a general theory of the Poisson Binomial distribution about its computation. Hong (2011) present two exact formulas for the Poisson Binomial distribution and three approximation methods.

The recursive formula for the Poisson Binomial distribution is summarized as follows. Let z_i be the outcome of random variable Z_i, $i = 1, \ldots, N$. The series of random variables is sorted in arbitrary order. Each variable is independent and follows $Z_i \sim$ Bernoulli(p_i), for $i = 1, \ldots, N$, where p_i is the success probability for Z_i. Let $S_i = \sum_{n=1}^{i} Z_n$ and $\gamma_{k,i} = \Pr(S_i = k)$. Then $\gamma_{k,i}$ represents the probability that the sum of outcomes of the first i variables is k. The recursive formula is given by

$$\gamma_{k,i} = \gamma_{k,i-1}(1-p_i) + \gamma_{k-1,i-1}p_i \quad \text{, for } 0 \le i \le N,\ 0 \le k \le i \tag{3.20}$$

Note that $\gamma_{-1,i} = 0$, $0 \le i \le N$, and $\gamma_{0,0} = 1$ are boundary conditions. Equation (3.20) follows the fact that total k successes in the first i experiments are either from k successes in the first $i-1$ experiments (fail in the i-th trail) or from $k-1$ successes in the first $i-1$ experiments and one success in the i-th trail. In the end, $\gamma_{k,N}$, $0 \le k \le N$ denotes the Poisson Binomial distribution.

We extend the above recursive formula to calculate $\Pr(AO_i = k \mid AT_i = t_1)$ in equation (3.21) that is a Weighted Poisson Binomial distribution. Since letter i has been used to denote the trip whose arrival time is fixed, we use i' as the index of other inbound trips whose arrival times are random. Let $Z_{i'}$, $i' = 1, \ldots, N$ be a series of random variables, where the trip i was excluded. Notice that the success probability $p_{i'}$ also depends on the arrival time of trip i and trip-ID order of i. It is $u_{i'it}$ as we defined in the model description section. However, to simply the discussion, we use $p_{i'}$ in Equation (3.21) and the

following pseudo code. Each of experiment represents if inbound trip i' with $M(i')$ crews has higher arrive-at-away order than trip i, given that trip i arrives at time t_1. If the results of the random experiment i' is a success, its outcome is $|M(i')|$. Otherwise, its outcome is 0. Each random variable is independent and follows $Z_{i'} \sim |M(i')|$*Bernoulli($p_{i'}$), for $i' = 1, \ldots, N$, where $p_{i'}$ is the success probability for $Z_{i'}$. Let $N_{i'} = \sum_{m=1}^{i'} Z_m$ and $\gamma_{k,i'} = \Pr(N_{i'} = k)$. Therefore, the recursive formula is given by

$$\gamma_{k,i'} = \gamma_{k,i'-1}(1-p_{i'}) + \gamma_{k-|M(i')|,i'-1}p_{i'} \quad \text{, for } 0 \le i' \le N, 0 \le k \le \sum_{n=0}^{i'} |M(n)| \tag{3.21}$$

Note that $\gamma_{k,i'} = 0$, for $0 \le i' \le N, k < 0$, and $\gamma_{0,0} = 1$ are boundary conditions.

Similarly, Equation (3.21) follows the fact that total k successes in the first i' experiments are either from k successes in the first $i'-1$ experiments (fail in the i'-th trail) or from $k-|M(i')|$ successes in the first $i'-1$ experiments and $|M(i')|$ successes in the i'-th trail. In the end, $\gamma_{k,N}$ for $0 \le k \le \sum_{n=0}^{N} |M(n)|$ denotes the distribution of arrive-at-away order of trip i given trip i's arrival time t_1.

The same formula applies to $\Pr(RO_j = k \mid RT_j = t_2)$ in equation (3.14).

Our algorithm starts with $i' = 1$ and iterates $k \in [0, \sum_{n=0}^{i'} M(n)]$ for the equation (3.21). And then it moves to $i' = 2$ and so on, until $i' = N$. The following pseudo code illustrates the algorithm in details. The complexity of the algorithm is $O(NQ)$, where N denotes number of Bernoulli trials (i.e., number of trips) and Q denotes $\sum_{n=0}^{N} B_n$.

Weighted_Poisson_Binomial_distribution_algorithm

Notation

i'	index of independent Bernoulli random variables, $i' = 1,2,\ldots, N$
$p_{i'}$	success probability of the Bernoulli random variable $Z_{i'}$, $i' = 1,2,\ldots, N$
$B_{i'}$	outcome of the Bernoulli random variable $Z_{i'}$ if it successes, 0 otherwise
$\gamma_{k,i'}$	probability that the sum of the first i' outcomes is k, for $0 \le i' \le N, 0 \le k \le \sum_{n=0}^{i'} B_n$
$\gamma_{k,N}$	probability that the sum of all outcomes is k, for $0 \le k \le \sum_{n=0}^{N} B_n$

Input: the presumed arrival time of trip i; success probability $p_{i'}$ and outcomes $B_{i'}$ for each Bernoulli random variable $Z_{i'}$ $i' = 1,2,\ldots, N$.

Output: $\gamma_{k,N}$, for $0 \leq k \leq \sum_{n=0}^{N} B_n$

Begin

Set $\gamma_{k,i'} = 0$, for $0 \leq i' \leq N$, $k < 0$, and $\gamma_{0,0} = 1$

for $i' = 1$ to N

{

for $k = 0$ to $\sum_{n=0}^{i'} B_n$

{

$$\gamma_{k,i'} = \gamma_{k,i'-1}(1 - p_{i'}) + \gamma_{k-|M(i')|,i'-1} p_{i'}$$

}

}

Return $\gamma_{k,N}$, for $0 \leq k \leq \sum_{n=0}^{N} B_n$

End

In the end, we analyze the overall computational complexity of the evaluation method. Based on the formulation of the analytical model at the end of Section 3.4.4, the overall computational complexity is $O(|M|(|T|)^2 |R| KNQ)$. $|M|$ and $|R|$ represent the number of inbound crews and outbound crew requirements respectively. $|T|$ denotes the number of discrete time period we generate in the planning horizon. K denotes the maximum possible number of arrive-at-away order and depart-from-away order. N denotes number of Bernoulli trials (i.e., number of trips). Q denotes $\sum_{n=0}^{N} B_n$.

3.5.2. Greedy local search algorithm

We first propose a greedy local search algorithm for the CSP with uncertainty. Starting from an initial deadhead plan denoted as DP_0, our algorithm moves to the best solution in the neighborhood $N(DP_0)$, which is denoted as DP^*, if DP^* is better than DP_0. At iteration t, it moves to the best solution in the neighborhood $N(DP_t)$ if it is better than DP_t. Figure 3.5 shows the flowchart of the search algorithm.

The search algorithm considers three types of movements, which define the set of neighbors $N(DP)$ of a deadhead plan DP. The first adds one crew in an inbound deadhead trip or one crew requirement in an outbound deadhead trip without exceeding deadhead capacity. The second removes one crew in an inbound trip or one crew requirement in an outbound deadhead trip which is selected in deadhead plan DP. The last one is the combination of the first two movements. It adds one deadhead crew/crew-requirement and removes one deadhead crew/crew-requirement simultaneously. The neighbor set of a deadhead plan $N(DP)$ is not only used in this greedy local search algorithm but also used in the Tabu search algorithm and Simulated Annealing algorithm.

The following pseudo-code summarizes the algorithm.

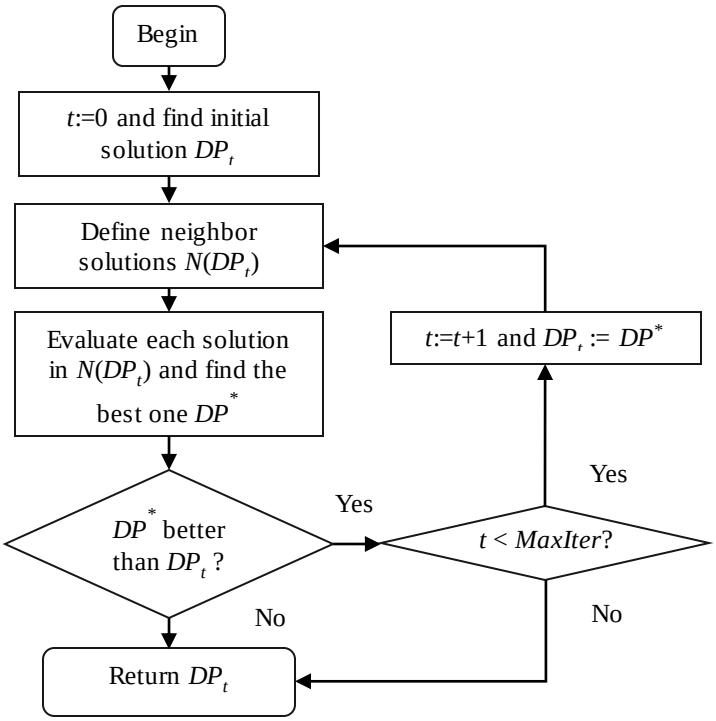


Figure 3.5. Flowchart of the greedy local search algorithm

Greedy_local_search_algorithm

Notation

DP_t set of deadhead trips selected in the solution at the t'th iteration

SP a candidate solution that corresponds to a set of deadhead trips

MaxIter number of iterations the algorithm repeats

Output: minimal expected cost solution that the greedy local search algorithm found

Begin

Set t :=1, $DP_t = \phi$,

while ($t < MaxIter$)

bestCandidate $\leftarrow \phi$

for (SP in $N(DP_t)$)

{

if (fitness(SP) > fitness(bestCandidate)))

bestCandidate $\leftarrow SP$

}

if (fitness(DP_t) > fitness(bestCandidate))

Return DP_t and **Exit**

$DP_t \leftarrow$ bestCandidate

$t \leftarrow t + 1$

end

Return DP_t

End

In the algorithm, the computational effort of evaluating the expected cost of each deadhead plan in $N(DP_t)$ quickly grows with the size of the problem. When the evaluator calculates the expected cost of a deadhead plan, computing $\Pr(AO_i = k \mid AT_i = t_1)$ and $\Pr(RO_j = k \mid RT_j = t_2)$ takes most run times. We develop a method that reuses these two probabilities in DP_t for neighbors in $N(DP_t)$, as shown in the Section 3.5.6.

3.5.3. Tabu search algorithm

The next search algorithm we tried is Tabu search algorithm. Like the greedy local search algorithm, in each iteration the Tabu search algorithm also check a solution's immediate neighbors (i.e., $N(DP)$) in the hope of finding an improved solution. However, the Tabu search enhance the performance of local search by accepting worsening moves in each iteration and using a Tabu list to avoid sticking in local optimal

solutions. A Tabu list records a list of solution that are visited within a certain short-term period. Figure 3.6 shows the flowchart of the Tabu search algorithm. The following pseudo code explains the algorithm in details.

Tabu_search_algorithm

Notation

DP_t set of deadhead trips selected in the solution at the *t*'th iteration

SP a candidate solution that corresponds to a set of deadhead trips

TabuList list of solutions that are temporarily forbidden

maxTabuSize maximal size of the Tabu list

MaxIter number of iterations the algorithm repeats

Output: minimal expected cost solution that the Tabu search algorithm found

Begin

Set $t := 1$, $DP_t = \emptyset$, $TabuList = \emptyset$

while ($t < MaxIter$)

    bestCandidate $\leftarrow \emptyset$

    for (*SP* in $N(DP_t)$)

    {

        if (*SP* is not in *TabuList* and fitness(*SP*) > fitness(bestCandidate)))

            bestCandidate $\leftarrow$ *SP*

    }

    $DP_t \leftarrow$ bestCandidate

    TabuList.push(bestCandidate)

    if (*TabuList*.size > maxTabuSize)

        TabuList.removeFirst()

    $t \leftarrow t + 1$

end

Return DP_t

End

Like the greedy local search algorithm, the main computational burden of the Tabu search is to calculate the expected cost of each solution in the neighbor of concurrent solution. This algorithm also use the update strategy in Section 3.5.6 to reduce this computational effort.

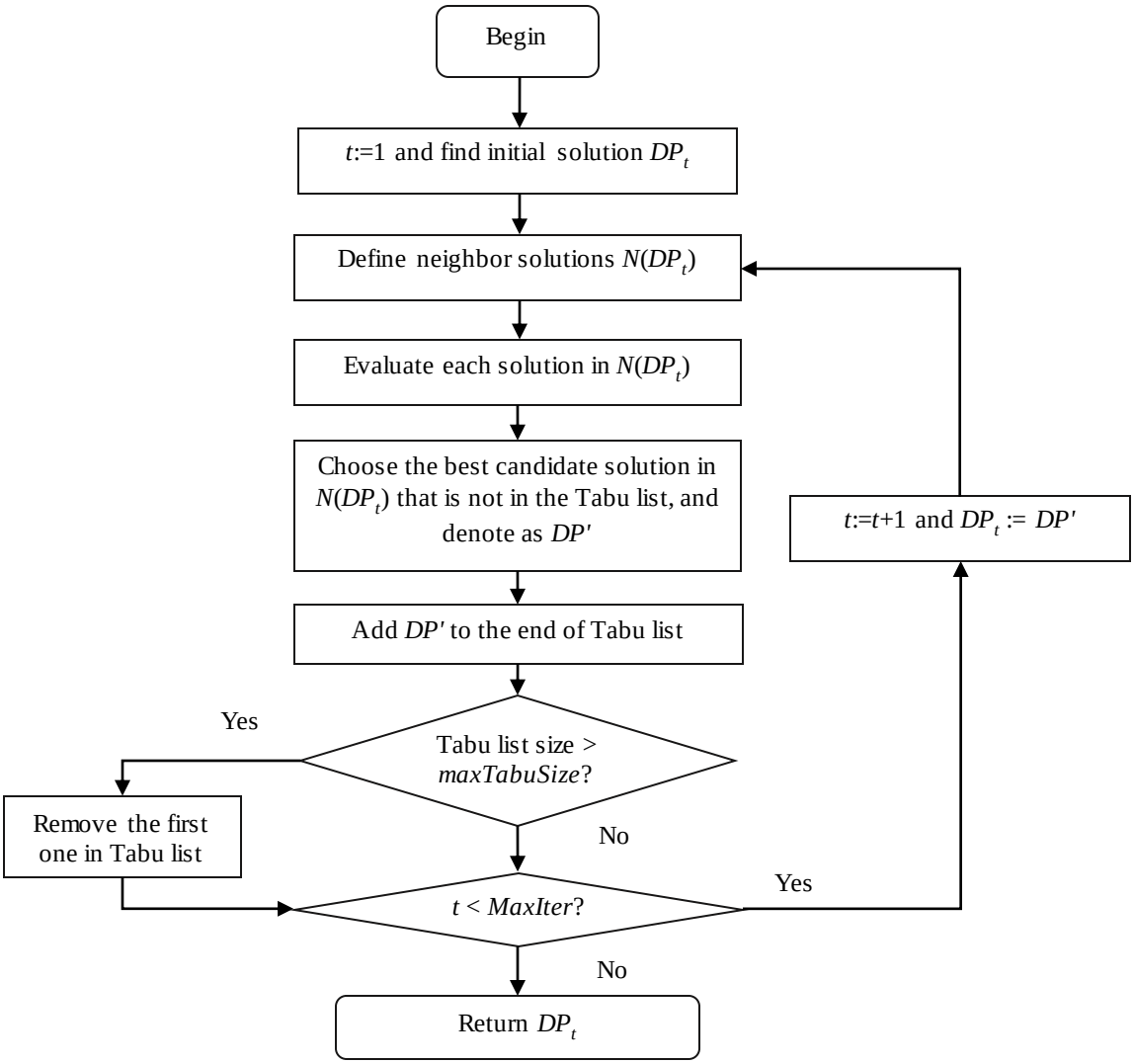


Figure 3.6. Flowchart of the Tabu search algorithm

3.5.4. Simulated Annealing algorithm

Besides the Tabu search algorithm, Simulated Annealing algorithm is another approach that enhances local search algorithm by introducing two tricks to avoid sticking at a local minimum. The first is that so-called "Metropolis algorithm", where solutions

that are worse than current solution are accepted when the algorithm intends to explore more of the space of solutions. Such non-improving solutions are allowed using the criterion that $e^{-\Delta D/k} > R(0,1)$, where ΔD is the change of objective functions from current solution (non-improving solutions have non-positive ΔD), K is called "synthetic temperature", and $R(0,1)$ represents a random number in the interval [0,1]. If K is large, it is easy to accept non-improving solutions so that the algorithm explores the solution space. The second trick is to lower the temperature gradually. After lowering the temperature several times to a low value, the algorithm only accepts good solutions and may eventually find the local minimum of the cost function. The flowchart of this algorithms is drawn in Figure 3.7.

The following pseudocode presents the simulated annealing heuristic as described above. It starts from a state DP_1 and continues to a maximum of *MaxIter* steps. In the process, the set $N(DP_t)$ is a neighbor set of a given state DP_t; the function *Random*(0, 1) returns a value in the range [0, 1], uniformly at random. The function $E(DP_t)$ return the expected cost of solution DP_t.

Simulated_Annealing_algorithm

Notation

DP_t set of deadhead trips selected in the solution at the t'th iteration

SP a candidate solution that corresponds to a set of deadhead trips

K_{max} maximal synthetic temperature

MaxIter number of iterations the algorithm repeats

p accept probability of a new solution

Output: minimal expected cost solution that the algorithm found

Begin

Set $t := 1$, $DP_t = \emptyset$

while ($t < MaxIter$)

$K \leftarrow K_{max} - t + 1$

Generate neighbor set of concurrent solution DP_t, denoted as $N(DP_t)$

Randomly choose *SP* from $N(DP_t)$

$p = e^{-(E(SP)-E(DP_t))/k}$

if $p > Random(0,1)$

$DP_t \leftarrow SP$

$t \leftarrow t + 1$

end

Return DP_t

End

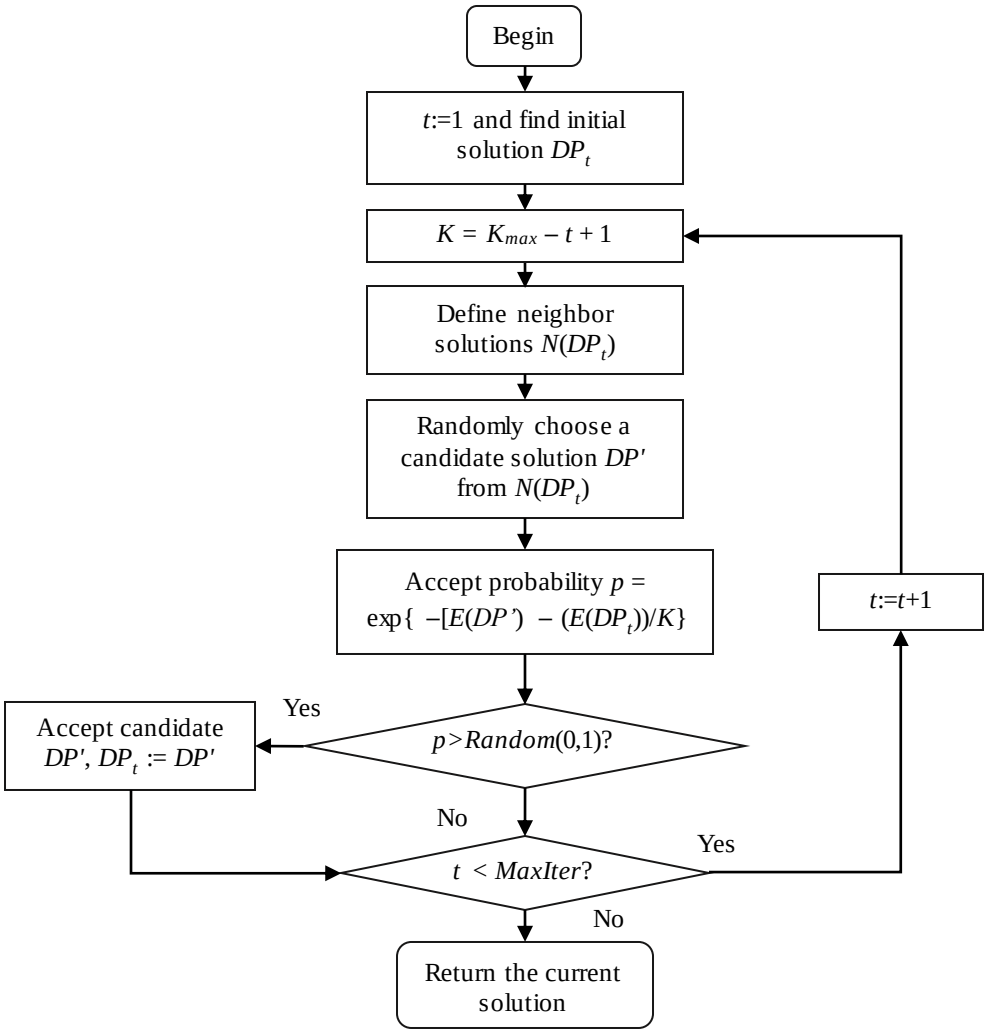


Figure 3.7. Flowchart of the Simulated Annealing algorithm

3.5.5. Genetic algorithm

A genetic algorithm is a method for solving optimization problems based on a natural selection process that mimics the process of natural selection. It usually uses the same combination of selection, inheritance and mutation to evolve a solution to a problem. Before we demonstrate the flowchart of our algorithm, it is necessary to present the way of encoding solution to the problem. We use a bit string as the chromosome that encodes any solution. Figure 3.8 provides an example of the chromosome and its corresponding solution. Each bit in the chromosome represents if the solution contains a particular crew/crew-requirement in a deadhead trip. For instance, an inbound taxi deadhead can send five crews to the away station. Then there are five consecutive bits in the chromosome representing these five crews in a candidate taxi respectively. When generating or modifying the chromosome for a taxi candidate, the first bit needs to be 1 before the second, and so on. For train deadhead trip with just one seat capacity, it takes only one bit to represent the trip in chromosomes.

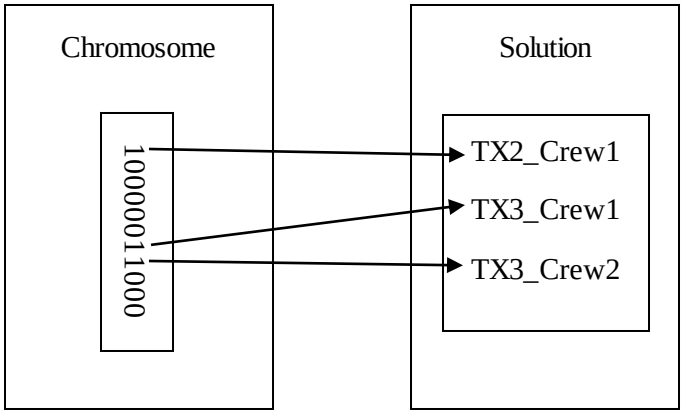


Figure 3.8. Chromosome example

Figure 3.9 presents the flowchart of our Genetic algorithm. It starts with initializing a set of solutions P_t. In each iteration, it calculates the total expected cost of each of the solution and then sort the solutions in ascending order of the cost. The algorithm copies all solutions in P_t to P_{t+1} and creates next generation by altering solutions in P_{t+1}. First the algorithm replaces the best solution in P_{t+1} by its best neighbor if such neighbor is a

better solution. Secondly, it pairs up the second half of solutions in P_{t+1} and crossovers their chromosomes. Thirdly, it mutates the last twenty percent of solutions in P_{t+1} by randomly replacing each one with one of its neighbors. With these three procedures, the P_{t+1} now represents the population in the (t+1)'s iteration. The algorithm repeatedly generates populations until the maximal iteration limit.

Genetic_algorithm

Notation

P_t population (i.e., set of solutions) at the t'th iteration

N number of solutions in population

$P_t(n)$ the n'th solution in population P_t

MaxIter number of iterations the algorithm repeats

Output: minimal expected cost solution that the algorithm found

Begin

Set t :=1, $SP = \emptyset$, $P_t = N(SP)$

while (t < *MaxIter*)

for every SP in P_t

Calculate $E(SP)$

Sort P_t in the ascending order of expect cost

$P_{t+1} \leftarrow P_t$

for n = 1 to N/2 do

{

$SP \leftarrow P_{t+1}(n)$

$SP \leftarrow$ the best solution in $N(SP)$

if $E(SP) < E(P_{t+1}(n))$, **then** $P_{t+1}(n) \leftarrow SP$

}

for n = N/2 to N*3/4

$m \leftarrow Random(0,1)*N/4 + N*3/4$

Switch the inbound deadhead trips selected in $P_{t+1}(n)$ and $P_{t+1}(m)$

for n = N*3/4 to N

{

$SP \leftarrow P_{t+1}(n)$

Choose one from $N(SP)$ at random and replace $P_{t+1}(n)$

}

$t \leftarrow t + 1$

end

for every SP in P_t

Calculate $E(SP)$

Sort P_t in the ascending order of expected total cost

Return $P_t(1)$

End

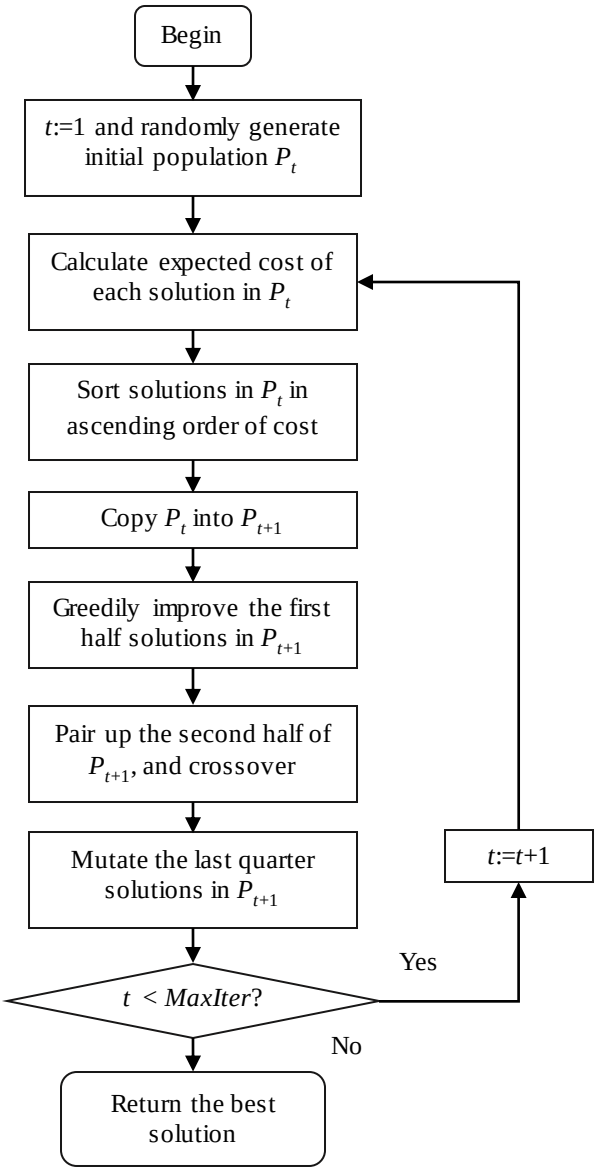


Figure 3.9. Flowchart of the Genetic algorithm

3.5.6. An update strategy in search algorithm

In our analytical model, we calculate $\Pr(AO_i = k \mid AT_i = t_1)$ and $\Pr(RO_j = k \mid RT_j = t_2)$ using the recursive method. Since different deadhead plans add different sets of deadhead trips, each deadhead plan needs to recalculate the two probabilities from scratch. But in our search algorithm, adjacent nodes/solutions only differ by one or two

deadhead trips. Thus, the motivation of reusing the previous probability results is intuitive.

We assume that the arrival and departure times of inbound and outbound taxi deadhead are certain. In this section, we show how the update strategy computes $\Pr(RO_j = k \mid RT_j = t_2)$ when a neighbor of DP_t adds a crew to an outbound trip h in DP_t or removes a crew from an outbound trip h in DP_t. In the meanwhile, $\Pr(AO_i = k \mid AT_i = t_1)$ does not change. The same analysis applies to inbound crews.

Let $\Pr_{+h}(RO_j = k \mid RT_j = t_2)$ denote the probability that outbound trip j is ready to depart at time t_2 with depart-from-away order k after adding a new crew to deadhead trip h. Similarly, let $\Pr_{-h}(RO_j = k \mid RT_j = t_2)$ denote the probability that outbound trip j is ready to depart at time t_2 with depart-from-away order k after removing a crew from deadhead trip h.

Case 1: Add a crew in an outbound deadhead trip

The deadhead trip h can be a train deadhead trip with ready-to-depart time within a range of $[et_h, lt_h]$ or a taxi trip or passenger train (Amtrak) with fixed ready-to-depart time. We first consider the former scenario and then discuss the later one.

If the ready-to-depart time t_2 of trip j is earlier than the earliest possible ready-to-depart time of train deadhead trip h, then the depart-from-away order of trip j should not change.

$$\Pr_{+h}(RO_j = k \mid RT_j = t_2) = \Pr(RO_j = k \mid RT_j = t_2) \quad \forall j, \forall t_2 < et_h, \forall k \tag{3.30}$$

Note that it still holds when $t_2 = et_h$ if trip j has higher train ID order than train deadhead trip h.

If the ready-to-depart time t_2 of trip j is later than the latest possible ready-to-depart time of train deadhead trip h, then the depart-from-away order of trip j should decreases by one (e.g. the value of RO_j adds one). Note that $RO_j = 1$means trip j has the highest depart-from-away order.

$$\Pr_{+h}(RO_j = k+1 \mid RT_j = t_2) = \Pr(RO_j = k \mid RT_j = t_2) \qquad \forall j, \forall t_2 > lt_h, \forall k \tag{3.31}$$

It still holds when $t_2 = lt_h$ if trip j has lower train ID order than train deadhead trip h.

For other trip j that may have a higher order than deadhead trip h, its new depart-from-away order changes as follows. Recall that v_{hjt_2} denotes the probability that the depart-from-away order of outbound trip h is higher than trip j given that trip j is ready to depart at time t_2.

$$\begin{aligned}\Pr_{+h}(RO_j = k+1 \mid RT_j = t_2) = v_{hjt_2} \Pr(RO_j = k \mid RT_j = t_2) \\ +(1-v_{hjt_2}) \Pr(RO_j = k+1 \mid RT_j = t_2) \qquad \forall j, \forall et_h \le t_2 \le lt_h, \forall k\end{aligned} \tag{3.32}$$

In the last, we consider the depart-from-away order of train deadhead trip h itself. Since there is one more crew in h, RO_h increases by one.

$$\Pr_{+h}(RO_h = k+1 \mid RT_h = t_2) = \Pr(RO_h = k \mid RT_h = t_2) \qquad \forall t_2, \forall k \tag{3.33}$$

Then we consider the scenario in which deadhead h is a taxi trip or passenger train (Amtrak) with fixed ready-to-depart time rt_h. If the ready-to-depart time t_2 of trip j is earlier than rt_h, then its depart-from-away order at time t_2 does not change. Otherwise, its depart-from-away order becomes lower (e.g. RO_j increases by one). When the ready-to-depart time t_2 of trip j is equal to rt_h, the departing order of trip j increases by one only if trip j has lower train ID than deadhead trip h.

Case 2: Remove a crew in an outbound deadhead trip

Similarly, we can remove a crew from a train deadhead trip with ready-to-depart time within a range of $[et_h, lt_h]$ or from a taxi trip or passenger train (Amtrak) with fixed ready-to-depart time. Since the latter scenario is just a special case of the former one, we only present the update strategy for the former one in this case.

For outbound trips with ready-to-depart time t_2, if $t_2 < et_h$ or $t_2 > lt_h$, they either have higher or lower depart-from-away order than deadhead trip h. We update $\Pr_{-h}(RO_j = k \mid RT_j = t_2)$ as follows.

$$\Pr_{-h}(RO_j = k \mid RT_j = t_2) = \Pr(RO_j = k \mid RT_j = t_2) \quad \forall j, \forall t_2 < et_h, \forall k \tag{3.34}$$

$$\Pr_{-h}(RO_j = k \mid RT_j = t_2) = \Pr(RO_j = k+1 \mid RT_j = t_2) \qquad \forall j, \forall t_2 > lt_h, \forall k \tag{3.35}$$

If the ready-to-depart time t_2 of an outbound trip is within a range $[et_h, lt_h]$, the relationship between $\Pr(RO_j = k \mid RT_j = t_2)$ and $\Pr_{-h}(RO_j = k \mid RT_j = t_2)$ can be expressed as follows.

$$\Pr(RO_j = k \mid RT_j = t_2) = v_{hjt_2} \Pr_{-h}(RO_j = k-1 \mid RT_j = t_2) + (1 - v_{hjt_2}) \Pr_{-h}(RO_j = k \mid RT_j = t_2) \qquad \forall j, \forall et_h \le t_2 \le lt_h, \forall k \tag{3.36}$$

$$\Leftrightarrow \Pr_{-h}(RO_j = k \mid RT_j = t_2) = \frac{1}{1 - v_{hjt_2}} [\Pr(RO_j = k \mid RT_j = t_2) - v_{hjt_2} \Pr_{-h}(RO_j = k-1 \mid RT_j = t_2)] \qquad \forall j, \forall et_h \le t_2 \le lt_h, \forall k \tag{3.37}$$

In the last, we consider the multi-job trip *h*. Since the number of crew in *h* decreases by one, RO_h decreases by one.

$$\Pr_{-h}(RO_h = k \mid RT_h = t_2) = \Pr(RO_h = k+1 \mid RT_h = t_2) \qquad \forall t_2, \forall k \tag{3.38}$$

Although the third type of movement in the search algorithm adds and removes a crew simultaneously, we can update probabilities $\Pr(RO_j = k \mid RT_j = t_2)$ and $\Pr(AO_i = k \mid AT_i = t_1)$ in two steps. We can first update these probabilities as the method in Case 1, and then update them as the method in Case 2.

The analysis of the updating strategies for $\Pr(RO_j = k \mid RT_j = t_2)$ also applies to $\Pr(AO_i = k \mid AT_i = t_1)$.

3.6. Computational results

In this section, we first generate random instances that differ in problem features to represent different scenarios of the freight railway crew scheduling problem. In Section 3.6.1 we discuss these problem features including the number of trains and distributions of regular train schedules. Then in Section 3.6.2 we test the stochastic programming model and the heuristic algorithms based on the random instances. Finally, we alter input parameters, such as cost parameters, and analyze the effects on final solutions of the optimization models and heuristic algorithms in Section 3.6.3. Observations and insights from the computational results are made in the end.

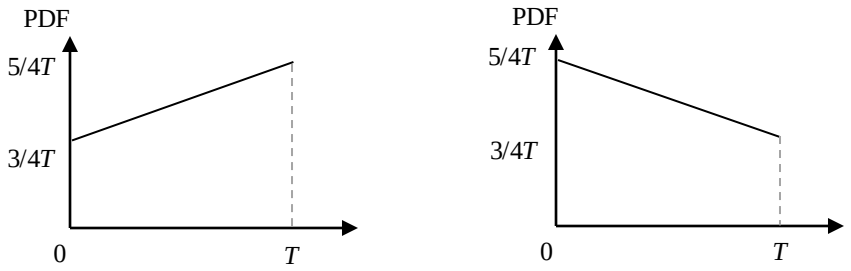
3.6.1. Generate random instances

When generating test instances at random, we consider six problem features: 1) planning horizon; 2) length of each time period; 3) numbers of inbound and outbound regular trains; 4) scheduled arrival time and departure time for regular trains; 5) actual arrival and departure time distribution for regular trains; 6) deadhead candidate trips.

Since we focus on large-scale instances, we only consider 48 hours planning horizon. When discretizing the planning horizon, the length of each discrete time period determines the accuracy of approximating the continuous problem. To be accurate, we use three minutes as the length of each time period. In this sub-section, we discuss features (3)-(6) and present a table that summarizes how each random instance is generated in the end.

The number of inbound and outbound trains in a given planning horizon mainly indicates the problem difficulty, especially the difference between amounts of inbound and outbound trains. If the difference is zero, it means that the instance may not need deadhead trips and is likely easy to solve. If the difference is non-zero, the instance intends to require deadhead trips to balance the traffic and to be hard to solve. To represent medium and large size problems, we generate two sets of instances with 20 and 35 outbound trains in each instance respectively. The amount of inbound trains in each instance is generated at random. For instances with 20 outbound trains, the number of inbound trains is generated randomly from a Uniform distribution in the range of [16, 24]. For instances with 35 outbound trains, the number of inbound trains is generated uniformly from the range of [31, 39].

The scheduled arrival times of inbound trains and ready-to-depart times of outbound trains are also important problem features. For example, if most inbound trains are scheduled to arrive at the early horizon but more than a half of outbound trains are scheduled to depart at the late horizon, then a large crew connection cost is expected. We consider two types of distributions: 1) Increasing distribution where the probability of train arriving/departing increases along time horizon with slope 0.5, and 2) Decreasing distribution where the probability of train arriving/departing decreases along time horizon with slope −0.5. Figure 3.10 depicts these two distributions. For example, to generate a trip's scheduled arrival time from the Increasing distribution, we draw a random variable from the PDF in Figure 3.10 (a).



(a) Increasing distribution (b) Decreasing distribution

Figure 3.10. PDFs of scheduled arrival/departure time

Once the scheduled arrival time of an inbound train is drawn, we generate its travel hours hence the departure time from the home station is realized. The travel hours of each train is generated uniformly in the range of [5, 15]. Similarly, we generate the arrival time at the home station for outbound trains.

Eighty percent of regular trains have a random arrival and ready-to-depart times. For each such train, we generate its timespan of uncertainty from a Uniform distribution in the interval of [1, 4 hours]. Let α denote the timespan of uncertainty for a regular train. Therefore, the actual arrival time of an inbound train is a random variable following a Uniform distribution in the interval of $[AT_i, AT_i + \alpha]$, where AT_i is its scheduled arrival time. In the same way, the actual ready-to-depart time of an outbound train is a random variable following a Uniform distribution in $[DT_j, DT_j + \alpha]$, where DT_j is its scheduled ready-to-depart time.

In the last, we discuss how train deadhead and taxi deadhead candidate trips are generated. We randomly choose forty percent of regular trains and add one train deadhead capacity to each train. For each inbound regular train, we add an outbound taxi deadhead candidate to avoid potential crew connection cost. The taxi departs at the scheduled arrival time of the inbound regular train plus the timespan of arrival delay distribution. For each outbound regular train, we add an inbound taxi deadhead candidate arriving at the scheduled departure time of the outbound regular train so that it can

prevent potential trip delay. It is worth mentioning that taxi deadhead trips are assumed to operate as the scheduled time table.

Each row in Table 3.2 represents a test instance. For each instance, we first generate the number of inbound regular trains based on the given number of outbound trains. And then we sample scheduled arrival time for each inbound regular train and scheduled departure time for each outbound regular train. For each regular train, we generate its timespan of uncertainty from a Uniform distribution in the interval of [1, 4 hours]. The distribution of actual arrival time of an inbound regular train is the uniform distribution plus its scheduled arrival time. Similarly, the distribution of actual ready-to-depart time of an outbound regular train is the uniform distribution plus its scheduled departure time. Given the scheduled timetable of regular trains, we construct deadhead candidates.

Table 3.1. Test instances summary

Instance	# of outbound train	Distr. of scheduled arrival time of inbound train	Distr. of scheduled depart time of outbound train
1	20	Increasing	Increasing
2	20	Decreasing	Decreasing
3	20	Increasing	Decreasing
4	20	Decreasing	Increasing
5	35	Increasing	Increasing
6	35	Decreasing	Decreasing
7	35	Increasing	Decreasing
8	35	Decreasing	Increasing

3.6.2. Test instances

For the stochastic programming model, we try different amounts of scenarios and solve the corresponding model by CPLEX. We test the four local search based heuristic algorithms starting from the optimal solutions of the static model and optimal solutions of the SP model. Finally, we compare the heuristic results to the optimal solution from the stochastic programming model. All algorithms were implemented in Java. Optimization models are solved by CPLEX 12.5 with its default settings. The termination gap is 0.5% and the maximal runtime limit is one hour. The computations were performed on a Linux server with a 2.3 GHz Xeon processor and 13 GB RAM.

The stochastic model is based on a set of scenarios. The more scenarios the model includes, the better it approximates the real objective function. But the expanding model size significantly increases the difficulty of solving the problem. Since the model size grows linearly with the number of scenarios, the model size grows about 150 times when 150 scenarios are added. We test the stochastic model with 10, 50, 100 and 150 scenarios and present the results in Figure 3.11. Model sizes (number of variables and constraints) are shown in Figure 3.11 (a) and (b). The final MIP gap of each SP model for each instance is shown in Figure 3.11 (c). The % gap is equal to the difference between the best integer objective and the objective of the best node remaining divided by the objective of the best node remaining. Figure 3.11 (d) shows the runtime for each SP model on each instance.

From Figure 3.11 (c), we can see that solving the stochastic model with 50 scenarios become very difficult even for small instances. For the instance 3, the SP model with 50 scenarios is already hard to solve since the CPLEX program terminates with 66% gap after one-hour runtime. Moreover, the final optimality gap of the SP model with 50 scenarios in the instance 7 is even 96%. However, the SP model with 10 scenarios solves all instances within 0.5% optimality gap in one hour. From Figure 3.11 (d), we can see that only the SP model with 10 scenarios can solve instance 7, which is the most difficult instance, within the maximum runtime limit.

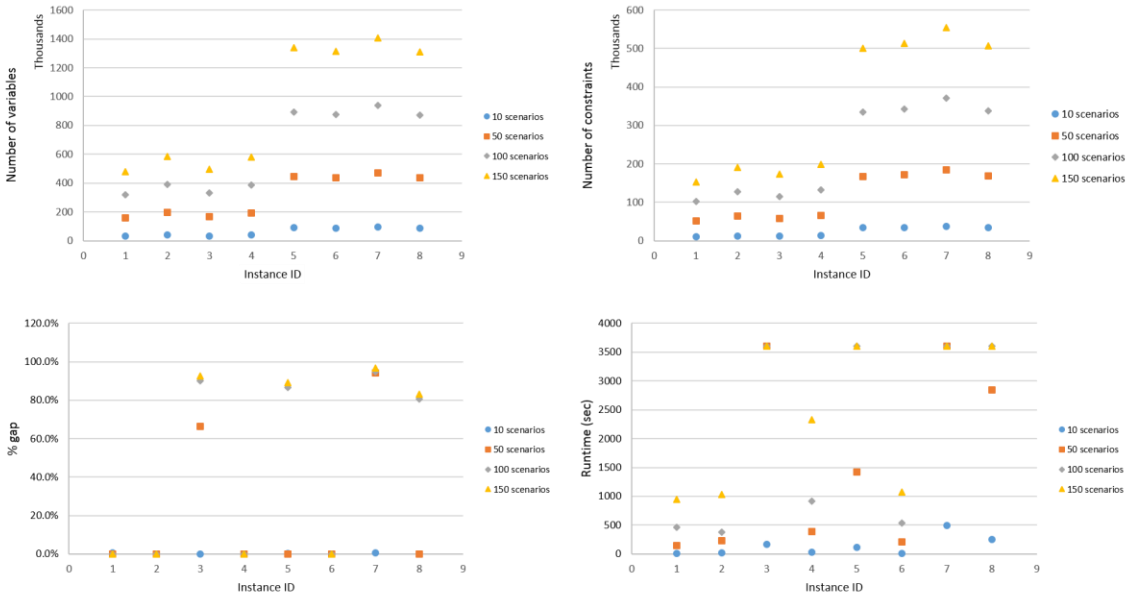


Figure 3.11. Compare SP models

Given the optimal solutions of the stochastic programming models, we want to know two things about the solution quality. One is the accuracy of the expected total costs of the final solutions that are estimated by the stochastic programming models. Second, we want to know if the optimal solutions to the stochastic model are optimal solutions to the original problem.

Table 3.2 answers the first question. It shows the relative difference between the total cost of the optimal solution that is estimated by SP models and the real expected total cost that is calculated by the analytical model based on the deadhead plan in the optimal solution. If the stochastic programming model poorly estimates the objective function of solutions, then the relative difference would be large. The bold relative difference numbers in Table 3.2 mean that the SP model does not solve the instance within 0.5% optimality gap at the termination. For those instances, the final solution of the SP model does not find the cheapest connection solution for a given deadhead plan. Thus, it violates the FIFO rule and results in the large objective value and relative differences. If we focus on the instances which have been solved to optimality, such as instances in the 10 scenarios case, the percentage gap tends to decrease as more scenarios are included. However, there is no clear evidence to indicate that 100 or 150 scenarios are enough to

get an accurate approximation of the real objective function. Even though more scenarios are needed in the SP model, the SP model with 50 scenarios has been too large to be solved for some instances.

Table 3.2. Compare solutions of the SP models

Instance	10 scenarios			50 scenarios			100 scenarios			150 scenarios		
	Estimated total cost	Real total cost	% gap	Estimated total cost	Real total cost	% gap	Estimated total cost	Real total cost	% gap	Estimated total cost	Real total cost	% gap
1	2444	2303	-6%	2249	2303	2%	2272	2303	1%	2275	2303	1%
2	2400	2456	2%	2423	2456	1%	2441	2456	1%	2426	2456	1%
3	7746	7846	1%	17585	17355	**-1%**	57847	57485	**-1%**	73890	15644	**-79%**
4	2242	2344	5%	2383	2344	-2%	2354	2344	0%	2372	2344	-1%
5	3210	3329	4%	3265	3329	2%	13677	9585	**-30%**	16477	13232	**-20%**
6	1013	1006	-1%	1006	1006	0%	1012	1006	-1%	1001	1006	1%
7	4101	4056	-1%	46113	20179	**-56%**	44498	22690	**-49%**	60649	32734	**-46%**
8	3134	3167	1%	3135	3167	1%	10305	9533	**-7%**	10423	9533	**-9%**

To answer the second question about solution quality in the SP model, we test the four heuristic algorithms and compare the heuristic solutions with the optimal solutions to the SP model with 10 scenarios. All of these local search based heuristic algorithms start from the optimal solution to the SP model with 10 scenarios. We will show the improvement of heuristic solutions to the origin solution. The reason why we choose the SP model with 10 scenarios is because of the tradeoff between difficult of solving the model and estimation accuracy.

Table 3.3 summarizes the greedy local search algorithm starting from the optimal solution of the optimal solution of the SP model with 10 scenarios. In each iteration of the algorithm, it calculates the total cost of each neighbor of current solution node and tries to move to a better solution node. The second column shows how many times it moves to a better solution node and the third column counts the number of neighbors whose total cost has been evaluated. The following columns show the expected cost components and total cost of the final solution. As we can see from the table, the algorithm does not find a better neighbor in the first iteration for the instances 4, 5 and 6.

Therefore, it ends at the starting solution without improvement. For the instance 3, the algorithm repeats for three iterations and finds three better nodes.

Tabu search modifies the local greedy search algorithm. In each iteration, the Tabu search algorithm also check a solution's immediate neighbors in the hope of finding an improved solution. However, the Tabu search enhance the performance of local search by accepting worsening moves in each iteration and using a Tabu list to avoid sticking in local optimal solutions. Table 3.4 presents the heuristic results. The third column indicates that the algorithm repeats for 25 iterations, which is the maximal amount of iteration, for all instances. Hence, it calculates more solution nodes' total cost and uses more runtime than the greedy local search algorithm.

Table 3.3. Greedy local search algorithm

Instance ID	Runtime (sec)	# of iterations	# of node visited	Expected crew connection cost	Expected trip delay cost	Deadhead cost	Expected total cost
1	6	1	193	97	352	1850	2299
2	6	1	219	834	165	1450	2449
3	15	3	585	240	342	6300	6881
4	5	0	104	506	388	1450	2344
5	25	0	249	480	848	2000	3329
6	9	0	106	242	364	400	1006
7	53	1	485	339	298	3300	3936
8	39	1	485	77	346	2650	3074

Table 3.4. Tabu search algorithm

Instance ID	Runtime (sec)	# of iterations	# of node visited	Expected crew connection cost	Expected trip delay cost	Deadhead cost	Expected total cost
1	53	25	2521	97	352	1850	2299
2	48	25	1883	834	165	1450	2449
3	85	25	3560	240	342	6300	6881
4	78	25	2108	506	388	1450	2344
5	496	25	4903	524	150	2650	3324
6	235	25	2839	242	364	400	1006
7	570	25	5915	235	721	2900	3856
8	414	25	5121	77	346	2650	3074

Table 3.5 presents the computational results to the Simulated Annealing algorithm. The algorithm randomly selects and evaluates one solution node of neighbors in each iteration. So it repeats 2000 iterations and calculates the expected total cost of 2001 solution node including the origin solution node. Since the main computational burden is to calculate expected total cost, this algorithm runs quicker than the Tabu search algorithm for most instances.

In the last, Table 3.6 demonstrates the heuristic results of the Genetic algorithm. Each iteration of the Genetic algorithm combines local search, crossover and mutation procedures. The local search procedure greedily improves the best solution node in the population while the crossover and mutation are applied to the rest of population. From Table 3.6, we can see that the number of nodes visited are less than that in the Tabu search and Simulated Annealing algorithm.

Table 3.5. Simulated Annealing algorithm

Instance ID	Runtime (sec)	# of iterations	# of node visited	Expected crew connection cost	Expected trip delay cost	Deadhead cost	Expected total cost
1	39	2000	2001	97	356	1850	2303
2	52	2000	2001	841	165	1450	2456
3	50	2000	2001	311	361	6300	6971
4	76	2000	2001	506	388	1450	2344
5	204	2000	2001	480	848	2000	3329
6	181	2000	2001	242	364	400	1006
7	191	2000	2001	460	296	3300	4056
8	159	2000	2001	168	349	2650	3167

Table 3.6. Genetic algorithm

Instance ID	Runtime (sec)	# of iterations	# of node visited	Expected crew connection cost	Expected trip delay cost	Deadhead cost	Expected total cost
1	21	5	845	97	356	1850	2303
2	24	5	910	834	165	1450	2449
3	29	5	1108	240	342	6300	6881
4	39	5	890	496	388	1450	2334
5	185	5	1615	480	848	2000	3329
6	84	5	890	242	364	400	1006
7	157	5	1585	339	298	3300	3936
8	134	5	1585	75	349	2650	3074

In Table 3.7, we want to compare all heuristic solutions with their origin solutions, which are the optimal solutions to the SP model with 10 scenarios. The second column of the table presents the runtime of the greedy local search algorithm. The percentage gap in the third column indicates the relative difference between the real expected total cost of the heuristic solution with the real expected total cost of the optimal solution to the SP model with 10 scenarios. Therefore, this gap indicates the improvement from the starting solution. The larger absolute value of this gap is, the larger improvement the

heuristic algorithm achieved. Since all the search based heuristics start from the optimal solution of the SP model, the heuristic solutions are either the same as or better than the starting solution node. So the all % gaps are non-positive values. Comparing the four heuristic algorithms, the greedy local search uses the shortest runtime while the Tabu search solutions are the best. The Genetic algorithm has similar results as the Tabu search. For instance 4, it finds even better solution than that of Tabu search. Overall, we suggest to use Tabu search and Genetic algorithm but it is upon to crew planners to decide which one they prefer.

Table 3.7. Compare to optimal solutions of the SP model with 10 scenarios

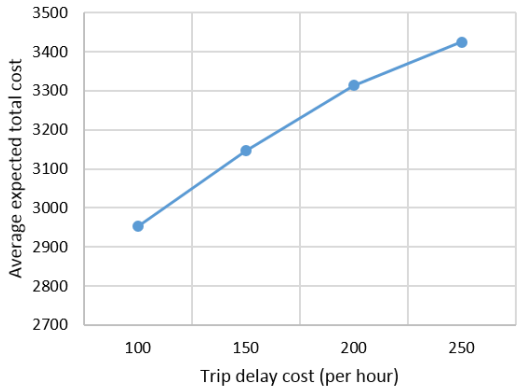
Instance ID	Local search		Tabu search		Simulated Annealing		Genetic algorithm	
	Run time (sec)	% gap	Run time (sec)	% gap	Run time (sec)	% gap	Run time (sec)	% gap
1	6	-0.1%	53	-0.1%	39	0.0%	21	0.0%
2	6	-0.3%	48	-0.3%	52	0.0%	24	-0.3%
3	15	-12.3%	85	-12.3%	50	-11.1%	29	-12.3%
4	5	0.0%	78	0.0%	76	0.0%	39	-0.4%
5	25	0.0%	496	-0.1%	204	0.0%	185	0.0%
6	9	0.0%	235	0.0%	181	0.0%	84	0.0%
7	53	-3.0%	570	-4.9%	191	0.0%	157	-3.0%
8	39	-3.0%	414	-3.0%	159	0.0%	134	-2.9%

3.6.3. Sensitivity analysis

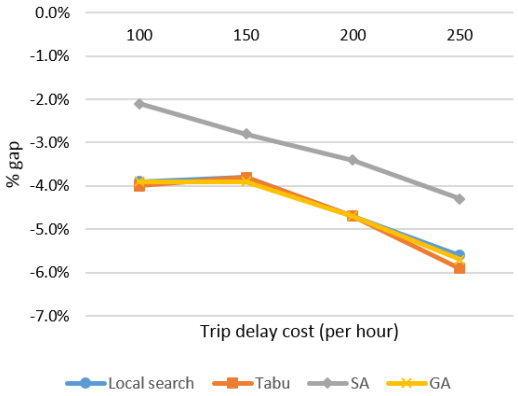
In this section, we examine how cost parameters impact the crew assignment decision and the performance of heuristic algorithms. We test the same set of instances that we randomly generate above. The cost parameters we studied are per hour trip delay cost, per hour layover cost and per crew deadhead trip cost.

Figure 3.12 summarizes the test results as we change the per hour trip delay cost. The expected total cost raises as the per hour trip delay cost, which is shown in Figure 3.12 (a). Figure 3.12 (b) indicates the performances of the heuristic algorithms as the trip delay cost varies. Compared to the SA algorithm, which visits neighbors at random, the

rest of local search based algorithms obtain similar results. All heuristics tend to get better solutions (more improvement from the staring solution node) when per hour trip delay cost increases. As the increment of the variance of total cost makes the optimal solution to the SP model with only 10 scenarios very poor, the heuristics are more likely to find better heuristic solutions.



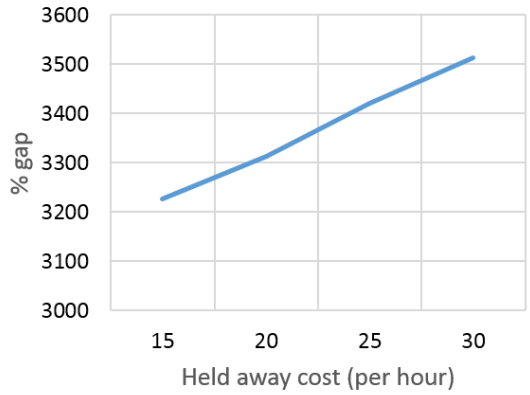
(a) Average expected cost of the optimal solution to the SP model with 10 scenarios



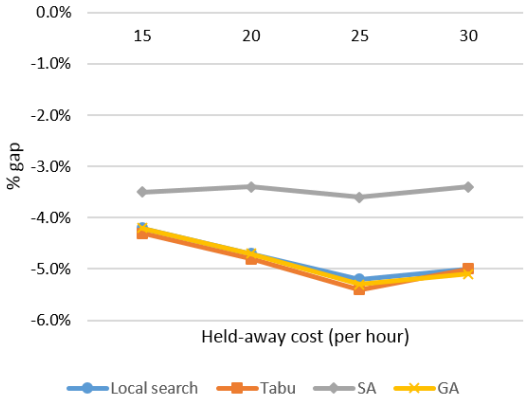
(b) Average gap between heuristic solutions and the optimal solution to the SP model with 1o scenarios

Figure 3.12. Sensitivity analysis to the trip delay cost

Figure 3.13 shows test results as the per hour layover cost changes. Figure 3.13 (a) shows that the average expected total cost increases as the per hour layover cost increases. The layover cost does not seem to significantly impact the performance of the heuristic algorithms, which are shown in Figure 3.13 (b).



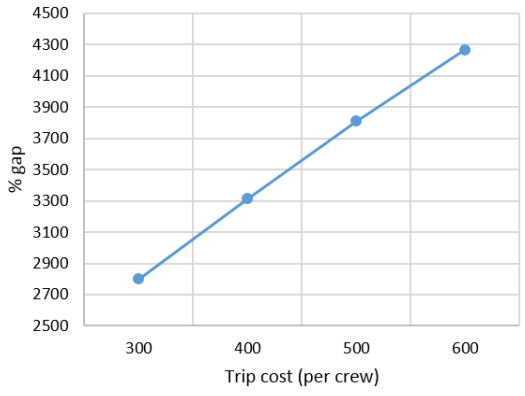
(a) Average expected cost of the optimal solution to the SP model with 10 scenarios



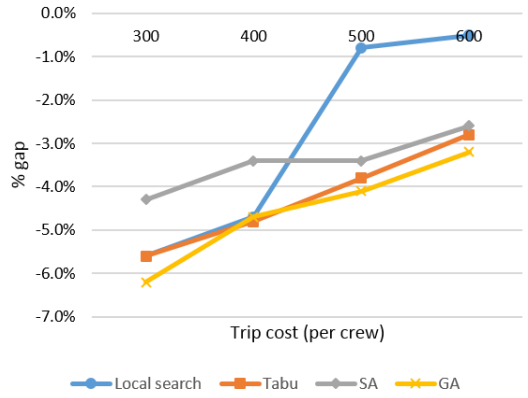
(b) Average gap between heuristic solutions and the optimal solution to the SP model with 1o scenarios

Figure 3.13. Sensitivity analysis to the layover cost

Figure 3.14 demonstrates the test results as we change the per crew deadhead trip cost. Figure 3.14 (a) shows that the average expected total cost increases as the deadhead trip cost increases. Figure 3.14 (b) indicates that the expensive trip cost leads to fewer deadhead trips selected in the optimal solutions of the SP model and the original problem. That is the reason why heuristic algorithms have less improvements from the starting solution when deadhead trip cost increases.



(a) Average expected cost of the optimal solution to the SP model with 10 scenarios



(b) Average gap between heuristic solutions and the optimal solution to the SP model with 1o scenarios

Figure 3.14. Sensitivity analysis to the deadhead trip cost

We can draw the following conclusions based the computational results in this section:

- Besides the distribution of train arrival/departure times, per hour trip delay cost and per hour layover cost parameters determine the variance of a given deadhead plan solution. Large values of such parameters make the stochastic programming model more beneficial.
- We use the solution improvement from the starting solution node to represent the performance of the local search based heuristic algorithms. The increasing per hour trip delay cost enlarges the variance of total cost and makes the SP model with 10 scenarios a poor estimation. Therefore, the heuristic algorithms make better improvements since the starting solution node (optimal solution of the SP model) is very poor. On the other hand, the expensive trip cost leads to fewer deadhead trips selected in the optimal solutions of the SP model and the original problem. Then heuristic algorithms have less improvements from the starting solution node when trip cost increases.

3.7 Conclusion

In this work, we studied the freight railway CSP by considering uncertainty in train schedules. We aimed to optimize deadhead decisions under uncertainty. We first introduced a stochastic programming model considering uncertainties in train schedules. Although this model does not enforce FIFO constraints in the formulations, we proved that the FIFO-satisfied solutions are the optimal solutions to the SP model because of the structure of their cost functions. To obtain an accurate approximation of the real objective function, the SP model needs a large number of scenarios. However, we showed that solving the SP model for even small instances can be very difficult. Therefore, we developed an analytical model to compute the expected total cost of a crew schedule and four local search based heuristic algorithms to improve deadhead decisions. Because the SP model with few scenarios provides a rough approximation, the search heuristics start from the optimal solution of the SP model with 10 scenarios.

The main contributions of this chapter are as follows:

- We proposed a stochastic programming model that considers uncertainty in train arrival and departure times. Given a set of scenarios of train arrival and departure

times, it minimizes expected total cost and provides optimal deadhead decisions with approximation. Because solving the SP model with large number of scenarios is difficult, we investigated the tradeoff between its approximation accuracy and computational difficulty.

- We proposed an analytical model that evaluates the performance of a deadhead plan solution under uncertainty. Based on presumed distributions of train departure/arrival times, the analytical model computes the expected total cost of a deadhead plan without using simulation or a set of scenarios. In a special case, the analytical model ignores potential layover cost increment caused by delaying a multi-job trip. We developed a polynomial algorithm to implement this model, which significantly reduces the computational burden.
- We developed four local search–based heuristic algorithms that improve deadhead decisions under uncertainty. The initial solution of the heuristic algorithms is the optimal solution to the SP model with 10 scenarios. Instead of calculating the expected cost for each crew schedule solution from scratch, our algorithms reuse previous calculation results and significantly reduce runtime.
- We randomly generated instances using the same cost components as those in crew assignment in a major North American railway company. We tested the stochastic programming model and heuristic algorithms based on the random instances. Finally, we altered input cost parameters and analyzed the impact of considering uncertainty, as well as compared the solutions obtained using the SP models and the solutions to the heuristic algorithms.

In future work on this topic, one may consider relaxing the two assumptions and explore real data to make the models more realistic. The first assumption we make is that trains operate independent of each other. However, this is not true in practice. One simple and realistic improvement would be adding an additional layer of uncertainty, such as weather, and making trains' arrival or ready-to-depart times depend on this random variable. Fortunately, the analytical model can be extended easily to handle this additional uncertainty. However, estimating such complex conditional probability

distributions is challenging. Even for the current analytical model, a good estimation of the independent distributions of trains' arrival or ready-to-depart times is essential. This can be solved by exploring a massive amount of train operational data and information about other factors that affect train operations (e.g., weather). Our second assumption is that a sufficient number of available regular crews are present at the home terminal. However, sometimes, crew planners need to deadhead crews home to maintain crew availability at the home station.

Chapter 4: Internal Mail Transport at Processing & Distribution Centers

4.1. Introduction

In 2011 the United States Postal Service (USPS) announced the Network Rationalization Initiative aimed at consolidating their national network of 673 processing and distribution centers (P&DCs). The goal was to streamline operations and reduce mail processing infrastructure costs (USPS report 2013). The consolidation had the effect of increasing what is still a staggering amount of mail that passes through the remaining facilities.

P&DCs are like high volume factories that run 24 hours a day, 7 days a week. On a typical day, a medium-size facility might receive as many as 5,000,000 letters, 500,000 flats, and thousands of parcels. To handle such a large volume, highly sophisticated optical character readers, bar code sorters, and other advanced equipment have been specially designed to automate as much of the mail stream as possible. Material handling also plays a key role. Bulk quantities of mail need to be transferred from and to the docks and between the various workstations. During a busy weekday, there may be over 4000 transfer requests in a typical P&DC.

To meet these requests, most facilities rely on forklifts and tugs, which are collectively called *powered industrial vehicles* (PIVs). In the short term, the USPS must determine the optimal number of PIVs to plan for each shift and then how to route them for a given equipment schedule and expected mail flow (Zhang and Bard 2005). To facilitate supervision and to avoid violating union rules, a complicating factor is the need to first cluster the pickup and dropoff points into zones and then assign each PIV to one of those zones. Of course, as the number of zones increases, so does the number of PIVs needed to move the mail. Understanding the tradeoff between cost and efficiency is fundamental to configuring and managing a facility, a challenge that the USPS has yet to meet.

With these considerations in mind, the purpose of this paper is to develop a methodology for determining the minimal number of PIVs needed to transfer mail

between P&DC workstations. As part of the methodology, we also determine the optimal route for each PIV within its assigned zone during a given shift. Because the number and composition of the zones is not input to the problem, we take a parametric approach and examine the relationship between number of zones and PIV requirements. Ideally, we would have liked to solve the facility layout, resource planning, and vehicle routing problems simultaneously but this was not possible due to the overwhelming complexity of the operational environment. Instead, we first cluster pickup and dropoff points, otherwise known as *control points*, into zones using the algorithm developed by Deng and Bard (2011). We then determine the minimum number of PIVs required to meet all demand for moving mail while minimizing the total vehicle travel times. That is, we solve a vehicle routing problem with time windows (VRPTW) subject to cross-zone movement restrictions.

The contributions of this work can be summarized as follows: 1) we outline an optimization model that clusters control points into zones; 2) we formulate a mixed-integer linear program for the pickup and dropoff problem in P&DC operations; 3) we follow the Column Generation approach to find a lower bound on the mixed-integer linear program; 4) we provide two optimization-based, rolling horizon heuristics for solving the joint capacity planning and routing problems; and 5) we present an extensive analysis of the Chicago P&DC that illustrates the efficiency of our computational scheme and the tradeoffs that exist in the operations of the facility. The Column Generation method is presented in Appendix A.

In the next section, we review the relevant literature. In Section 4.3, we define the routing problem, state the modeling assumptions, and describe the operational requirements. The formulation for the VRPTW is presented in Section 4.4. The solution methodologies are described in Section 4.5, followed in Section 4.6 with our computation results. Conclusions are drawn in Section 4.7 along with some suggestions for future research.

4.2. Literature Review

The VRPTW and its variants have been studied extensively over the last four decades. Comprehensive reviews can be found in Bodin et al. (1983), Solomon and Desrosiers (1988), and Parragh et al. (2008). Complexity results for a number of vehicle routing and scheduling problems are provided by Lenstra and Rinnooy Kan (1981) who show that most variants are strongly NP-hard. Instances involving 100 or more requests remain a challenge for the research community as indicated by the fact that not all of Solomon's benchmark VRPTW instances (Solomon 1987) with 100 customers have been solved optimally.

With some qualifications, our work can be considered a pickup and delivery problem with time windows (PDPTW). This problem is a generalization of the VRPTW in which every transportation request is associated with a pickup and a delivery point and can be preemptive, non-preemptive or mixed according to the problem context. Savelsbergh and Sol (1995) discuss several characteristics that distinguish pickup and delivery problems from standard VRPs. They present a survey of possible versions of PDPs and provide an overview of existing solution methods through 1995. A review of more recent work is given by Berbeglia et al. (2007).

Both exact and heuristic solution methods have been developed for the PDPTW. Exact methods mainly center on either branch-and-price (B&P) or branch-and-cut (B&C) algorithms, which can reliable solve most instances with up to 50 and often up to 100 requests, depending on the variant and length of the time windows. Much of the current research relies on heuristics to find high quality solutions quickly, as required in real world settings.

B&P algorithms integrate column generation and branch and bound. Column generation provides a flexible framework that can accommodate complex constraints and a large number of decision variables. Column generation terminates with a relaxed (fractional) solution, though, and so is usually embedded a branch-and-bound scheme to find integer (optimal) solutions. Dumas et al. (1991) were one of the first to use B&P for the PDPTW. Unique at the time, they proposed a dynamic programming label-correcting

algorithm to solve the shortest path subproblem with capacity and time window constraints. Their algorithm was successfully applied to eight instances with 19 to 55 requests, finding solutions within 5 minutes. Desrochers et al. (1992) tackled the VRPTW using B&P, and proposed several different dynamic programs for the subproblem. Their method was capable of solving a small subset of the 100 customer instances in the Solomon data sets.

B&C algorithms integrate cutting planes and branch and bound. The key is to find 'tight' cutting planes quickly, which often requires the use of heuristics since solving the corresponding separation problems can be computationally expensive. Araque et al. (1994) provide a B&C algorithm for a capacitated VRP in which all customers have unit demand. They present facet-inducing inequalities for a path-partitioning model, including trivial inequalities (linearization of binary connection variables), degree-2 constraints, generalized subtour elimination constraints, as well as large, intermediate and small multistar constraints. Separation heuristics were used to detect violations of the subtour elimination constraints and multistar constraints. Testing showed that their algorithm could optimally solve randomly generated instances with up to 60 customers. Bard et al. (2002) proposed a B&C algorithm for a VRPTW in which each customer imposes one or two service requests during the day. Each pickup request was required to be delivered to the depot. The contribution of their work was the introduction and integration of a wide variety of cuts including comb inequalities, 2-path inequalities, directed cycle elimination constraints, box inequalities, and incompatible pair inequalities. Separation heuristics were presented for each. Computational testing showed that they were able to solve most of Solomon's 50 customer instances and many of the 100 customer instances with tight time windows.

Branch-and-price-and-cut (B&P&C) algorithms are a combination of the above two exact methods. For the VRPTW, Desaulniers et al. (2008) proposed a B&P&C framework in which tabu search is called to solve the subproblem. To facilitate the computations, the elementarity requirement was relaxed in the subproblem and generalized k-path inequalities were imposed. Using the Solomon 100-customer

instances they were able to provide five new optimal solutions. Ropke and Cordeau (2009) tackled the PDPTW using a B&P&C algorithm. They introduced several new valid inequalities to the master problem and used a labeling algorithm to solve the elementary shortest path subproblem with time window, capacity, and pickup and delivery constraints. An important contribution was the procedure used to modify the subproblems when cuts were added to the master problem.

Gutiérrez-Jarpa et al. (2010) studied a PDP in which all deliveries must be made but only those pickups that are profitable and can be met within their time windows. The objective was to maximize the revenue associated with the pickups minus the routing costs. They developed a B&P algorithm and presented computational results for instances containing up to 100 customers.

Cherkesly et al. (2015) proposed models and algorithms for the PDPTW and multiple stacks, where each stack represents a queue of items and is operated on a last-in-first-out basis; that is, the last loaded item must be the first to be unloaded. They developed two different B&P&C algorithms. The first incorporates the multi-stack policy in the shortest path pricing subproblem, while the second incorporates this policy partly in the pricing subproblem and adds cuts to the master problem when infeasible multi-stack routes are encountered. They showed that instances with up to 75 requests and with up to three stacks could be solved to optimality within two hours.

To overcome the difficulties in solving generic versions of the VRP with exact algorithms, researchers have developed a wide range of heuristics to find high quality solutions in limited time. In a short survey, Laporte et al. (1992) categorize routing heuristics as being either classical or modern. They review tabu search heuristics, which represent the modern or meta-heuristic approach, and the savings method, cluster-first route-second approaches, and so on, which they consider classical.

Van der Bruggen et al. (1993) employ a two-phase local search algorithm for a single vehicle PDPTW with capacity constraints. In the first phase, a feasible route is constructed. While capacity constraints are never violated, this phase starts with a time window infeasible solution and iteratively reduces the infeasibility by penalizing

violations. The second phase iteratively minimizes route duration. In both phases, they use a variable-depth search that is based on seven basic types of arc-exchange procedures. To avoid poor or even infeasible solutions, they also developed an alternative approach designed around a penalized simulated annealing algorithm that finds high quality solutions when computation time is not a factor.

Li and Lim (2003) developed a tabu-embedded simulated annealing algorithm for the PDPTW. The algorithm has the feature of restarting a search from the current best solution after several non-improving iterations. Three pickup and deliver pair swapping operators are used to define a neighborhood. For testing purposes, they generated 56 instances from Solomon's 100-customer data sets by randomly pairing up demand points. The computational results led them to claim that their algorithms are the first to efficiently solve practical size multiple vehicle PDPTWs.

Lau and Liang (2002) proposed a two-phase procedure for the PDPTW. In the first phase, they applied a novel construction heuristic to generate an initial solution, which relies on insertions and angular sweeps with the radius at the center of the feasible. In the second phase, tabu search is used to iteratively improve the solution. Another contribution of the work is that they developed a strategy to generate good problem instances and benchmarking solutions based on Solomon's data sets.

A two-stage hybrid algorithm for the PDPTW is presented in Bent and Van Hentenryck (2006). Their objective was to first minimize the fleet size and then travel costs. The first stage consists of a simulating annealing algorithm that minimizes the number of routes. A large neighborhood search (LNS) algorithm is then applied in the second stage to decrease total travel costs. The neighborhood used in their LNS extends the approach of Shaw (1998). The full algorithm improves 10 of the 56 best published solutions for Solomon's instances and matches 35 others. Ioannou (2007) also used a simulated annealing-based algorithm to find feasible solutions to a VRP with a twofold objective identical to ours; that is, minimize the number of vehicles and then the total distance traveled. In his application, the VRP is embedded in a concurrent layout and material handling system design problem.

Derigs and Döhmer (2008) discuss an indirect (evolutionary) local search heuristic for the PDPTW. Such heuristics separate the problem of determining the feasibility of solutions from the objective-driven meta-heuristic search process by using simple encodings and appropriate decoders. They claim that their algorithms are not only flexible and simple but also give results that are competitive with the tabu search method proposed by Li and Lim (2003). Schneider and Henning (2014) studied a VRPTW with driver-specific times. In this environment, travel and service times are based on the drivers' familiarity with the customers and service regions. They propose a tabu search algorithm that is able to find good results for the Solomon instances.

More recently, Liu et al. (2013) addressed a PDPTW encountered in home health care logistics where it is necessary to transport material between pharmacies, patients, hospitals and labs. To solve this problem with both pickup and delivery and pickup windows, they first proposed two mixed-integer programming models and then a genetic algorithm and a tabu search heuristic. For testing purposes, they generated instances from Solomon's 100-customer data sets and from the data sets investigated by Gehring and Homberger (1999). The results indicated that the performance of their heuristics significantly dominated CPLEX in terms of solution quality and runtime. They also compared their heuristics with the simulated annealing algorithm of Hasama et al. (1998) on the 100 customer instances and were again able to show dominance.

In addition to work cited on the PDPTW, there is also a large body of literature on static dial-a-ride problems, which are similar to ours since we are primarily concerned with origin-destination transport. Various solution approaches are presented by Cordeau (2006), Parragh (2011), and Xiang et al. (2006); for a hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem, see Berbeglia et al. (2012).

4.3. Problem Definition

For a given set of zones within a P&DC, we are faced with the problem of minimizing the number of PIVs needed to transfer the mail between workstations for each of 3 shifts a day, 7 days a week. Each transfer request specifies a pickup and a dropoff location, and a time window in which service must begin. Transshipments at

intermediate locations are not permitted so a PIV services no more than one request at a time. If a vehicle arrives at a pickup location before the time window, it has to wait until the opening time.

For planning purposes, all requests are assumed to be known and deterministic. In the analysis, we use mail volume for a typical week in the year. The number of requests varies from day to day and from shift to shift. In our test instances, daily requests approach 4000 at their peak, and hourly requests can be more than 200 on high volume days. Minimizing the number of vehicles required to meet this demand minimizes operational costs, reduces congestion in the facility, and simplifies supervisory oversight.

Pickup and dropoff locations are fixed positions in a two-dimensional space. We call a pickup or dropoff location a *control point*. As a consequence of the mail flow, a control point may be a pickup point for one request and also a dropoff point for another request.

A *segment* denotes a mail transfer request from a pickup point to a dropoff point. Figure 4.1 depicts a simplified example of a facility in which the control points are clustered into three zones. In the diagram, the circles are the control points and the large arrows represent a transfer request from a pickup location to a dropoff location. Requests are labeled by numbers in square brackets; for example, for request [3], mail needs to be picked up at control point 5 and dropped off at control point 6. The thin arrows in the diagram correspond to the possible paths that a PIV can follow between service requests when it is not transporting mail. Notice that the pickup and dropoff points of segment 3 are located in different zones. The implication is that there is not much traffic between control points 5 and 6.

What is missing from Figure 4.1 is the time component of the operations. Each request i has an associated time window $[a_i, b_i]$ during which service must begin. Over the day, there may be many requests from control point l to control point k. The diagram is just a snapshot in time and does not show all requests or all possible transitions. Also, missing from Figure 4.1 are the workstations which are generally located at the control points. Their operations are modeled implicitly by the time windows. For example, let the pickup and dropoff points associated with segment i be l and k, respectively. The

value of a_i indicates the earliest time at which mail can be pickup up at l and transported to k to begin processing on the equipment at k. The value of b_i indicates the latest time permitted for a pickup at l. The implication is that if the mail were picked up at l later than b_i, it could not be delivered to k early enough to ensure that the equipment at k remains busy. The effects of this delay may ripple throughout the facility causing bottlenecks and other disruptions.

A solution to a simple two-zone, two-vehicle problem is depicted in Figure 4.2. The segments can either be thought of as nodes that must be visited or as arcs that must be traversed. Nominally, a depot is included for each zone but in reality, all vehicles start and end their shift at the same location. The path of the first vehicle starts at depot 1 and connects segments 1, 2 and 3; the path of the second vehicle starts at depot 2 connects segments 5 and 4 in that order. Note that the dropoff point of segment 1 is the same zone as the pickup point of segment 2. This is indicated by the loop at control point 2.

Cross-zone movement of vehicles is subject to operational rules. First, requests whose pickup and dropoff locations are within the same zone must be serviced by vehicles assigned to that zone. Second, a request whose pickup location is in zone k and dropoff location is in zone l may be serviced by a vehicle assigned to either zone k or zone l. While the former case is intuitive, the latter case is only permitted when a vehicle based in zone l drops off mail in k and on its return trip is able to satisfy a request to transport mail back to zone l.

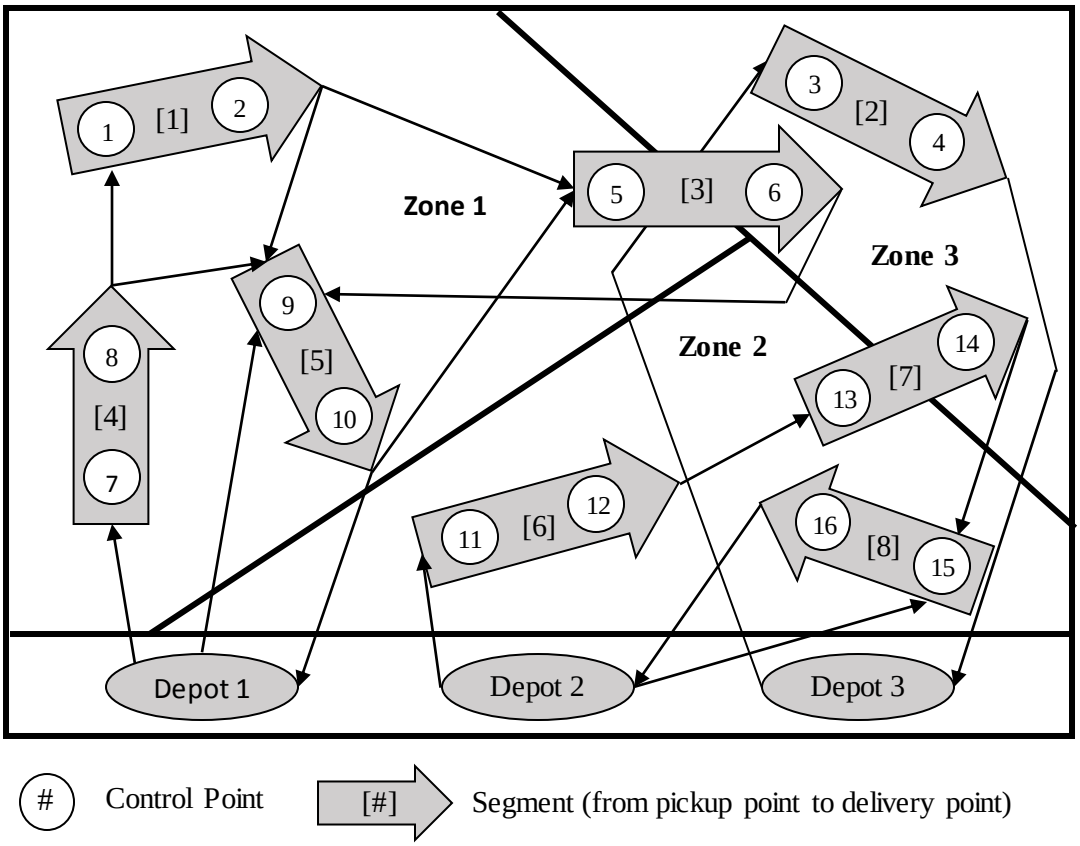


Figure 4.1. Example of clustering control points and PIV routes

The solution in Figure 4.2 illustrates two possible scenarios of cross-zone movement. After transporting mail from control point 2 to control point 3 in segment 2, the zone 1 PIV drives without a load to control point 4, satisfies the request for segment 3 and then returns to depot 1. In the other scenario, the zone 2 PIV starts at depot 2, drives to control point 8 to service segment 5, and after dropping off the mail at control point 9 in zone 1, returns to zone 2 to pick up mail at control point 6. Eventually, it returns to depot 2.

As mentioned, we wish to cluster the control points into zones with the understanding that this might lead to more PIVs than actually needed to meet demand. To perform the clustering, we must establish a measure that reflects the desirability of placing two control points in the same zone. This measure could be a function of the distance between the two control points, the volume of mail that must be moved between the two

control points, or a combination thereof. Once the zones are determined, we need to decide the number of vehicles to assign to each. This is a long-term planning problem, so within some limits, the decisions remain fixed for up to a year. Under various conditions, though, daily adjustments can be made to deal with demand spikes and absent drivers.

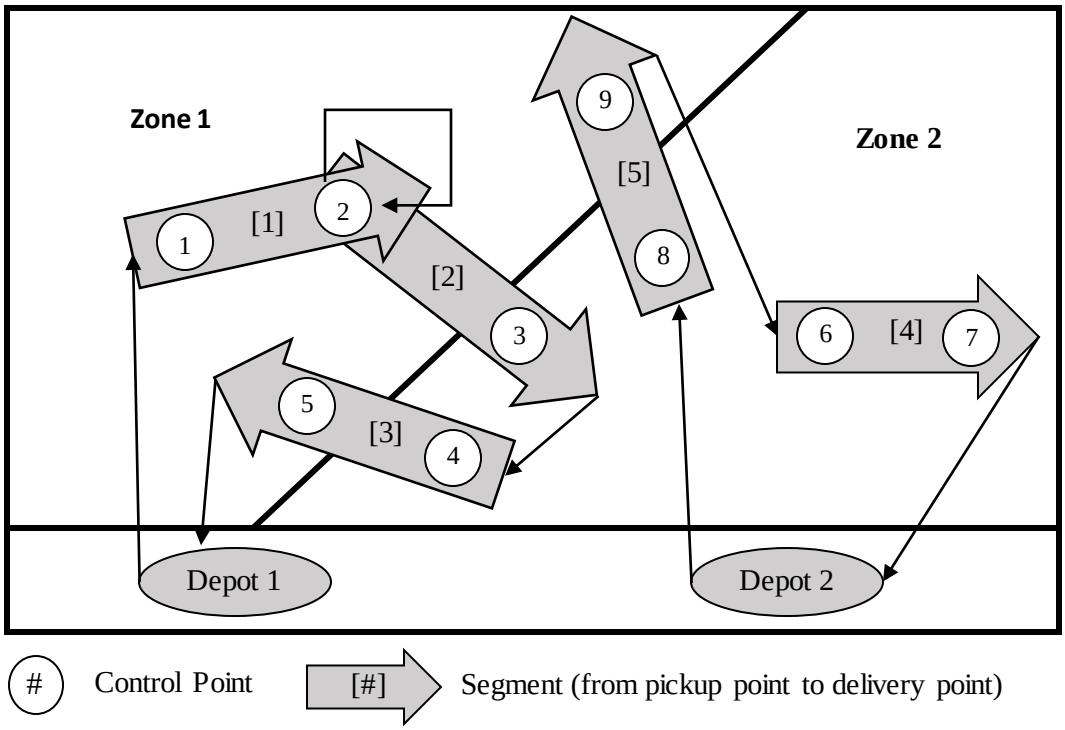


Figure 4.2. Feasible solution for vehicle routing problem

Due to the impracticality of formulating and solving a single model that embodies all the components of the P&DC resource planning problem, we take a two-stage approach. In the first stage, we solve a constrained clustering problem to fix the zones using the algorithm of Deng and Bard (2011). Results for the current problem are given by Yang (2016). In the second stage, we determine the minimum number of PIVs to assign to each zone. This is done by solving a VRPTW with a secondary objective of minimizing the total travel time of the PIVs. As by-product of the analysis, we get individual vehicle routes.

4.4. PIV Routing Model

Before we can cluster the control points, we need a measure that quantifies the desirability of placing two control points in the same zone. In Section 4.1 we provide a model that takes this information and creates up to K zones by maximizing the weighted sum of the edges in a cluster. The VRPTW with cross-zone constraints is presented in Section 4.2.

4.4.1 Clustering control points

For analytic purposes, control points can be represented by nodes in a graph $G = (E, V)$, where V is the set of n control points in the facility and E is the set of edges between them. In G, an edge $e = (i, j) \in E$ would only exist if there were some flow between its endpoints, i and j, over the week (actually, the flow is directional so "arcs" would be more appropriate than "edges"; however, to be compatible with the clustering literature we have adopted the latter term).

The formulation given below is aimed at maximizing the sum of the "benefits" of the edges strictly within each zone. It is assumed that the solution is static; i.e., it doesn't change over the day or during the week. A total of K zones will be formed satisfying the property that the sum of the node weights in each zone should be at least C^{min} and does not exceed some threshold C^{max}. These values are treated as user-supplied parameters.

In the developments, the following notation is used.

Indices and sets

- k index for zones; $k = 1,\ldots,K$
- i index for control points
- e index for edges in the graph of the facility [two control points define each edge; edges have direction so $(i,j) \neq (j,i)$]
- V set of control points; $i \in V$
- E set of edges in graph; $e \in E$

Parameters

- c_e weight of edge e in graph (a measure of the volume of mail and distance between the two control points that define e)

d_e length of edge e (physical distance between the two control points that define e)

α_i weight of control point i (may be a function of the mail volume originating at control point i)

K number of zones to be created

C^{max} maximum node weight permitted in each cluster

C^{min} minimum node weight required in each cluster

Variables

x_{ek} 1 if edge e has both its endpoints in zone k; 0 otherwise

y_{ik} 1 if control point i is included in zone k; 0 otherwise

Model

$$\text{Maximize} \quad \sum_{k=1}^{K} \sum_{e \in E} c_e x_{ek} \tag{4.1}$$

$$\text{Subject to} \quad \sum_{k=1}^{K} y_{ik} = 1, \quad \forall i \in V \tag{4.2}$$

$$x_{ek} \le y_{ik}, \; x_{ek} \le y_{jk}, \; \forall e = (i, j) \in E, \; k = 1,\ldots,K \tag{4.3}$$

$$x_{ek} \ge y_{ik} + y_{jk} - 1, \; \forall e = (i, j) \in E, \; k = 1,\ldots,K \tag{4.4}$$

$$C^{min} \le \sum_{i \in V} \alpha_i y_{ik} \le C^{max}, \; k = 1,\ldots,K \tag{4.5}$$

$$x_{ek} \in \{0,1\}, \; y_{ik} \in \{0,1\}, \; \forall \; i \in V, \; e = (i,j) \in E, \; k = 1,\ldots,K \tag{4.6}$$

The objective in (4.1) is to maximize the sum of the weights of edges within zones, which is equivalent to minimizing the sum of the weights of edges between zones. If the endpoints of edge e are not in the same zone, then the corresponding weight c_e is not counted. Constraints (4.2) ensure that each control point i is included in exactly one zone, while constraints (4.3) and (4.4) specify that edge $e = (i, j)$ is in zone k if and only if both endpoints i and j are in zone k. Constraints (4.5) sum the weights of the nodes in zone k and limit the total to be between C^{min} and C^{max}. If $\alpha_i = 1$ for all $i \in V$, then (4.5) restricts the number of control points in a zone to a maximum of C^{max} and a minimum of C^{min}. Note that if (4.5) is omitted, it is optimal to create a single zone; i.e., all the control points and hence edges would be in one zone. Binary restrictions are placed on all the

variables in (4.6); however, as long as $y = 0$ or 1, x will also be 0 or 1 so it can be treated as a continuous variable in the range [0, 1].

Constraints (4.4) are actually redundant in light of the objective function. When either y_{ik} or $y_{jk} = 0$, $x_{ek} = 0$, which gives a feasible solution to (4.4). When both y_{ik} and y_{jk} are 1, x_{ek} will be 1 as well since the objective is to maximize the total weight. Again (4.4) is automatically satisfied and so can be omitted.

To make the model operational, c_e must be specified. One possibility is to associate the weight of an edge with the volume of mail that is moved between its two endpoints over the week. Let v_{ij} be the amount of mail that is picked up at control point i and dropped off at j during the 7-day planning horizon. If the primary concern is with the absolute volume of mail that goes between i and j and not the percentage volume in either direction, then it is natural to set the weight as $c_e = v_{ij} + v_{ji}$, where $e = (i, j)$. This has the effect of minimizing the flow of mail between zones. If directional flows are an important consideration, they can be taken into account by letting the fraction of mail that goes from i to j be $f_{ij} = v_{ij}/(v_{ij} + v_{ji})$, and setting $c_e = 1 - |f_{ij} - f_{ji}|$. Note that $0 \le c_e \le 1$. In this case, the closer c_e is to 1, the more balanced the flow, so the more desirable it would be for control points i and j to be in the same zone. Also, it is an easy matter to define c_e in terms of distance rather than volume. In our implementation, we set c_e = mail_flow[i,j] / ((1+ distance[i,j]) × max_distance), where $e = (i,j)$ and max_distance = max{distance[i,j] : $\forall$ $(i,j) \in$ E}.

4.4.2 PIV routing problem

For fixed zones and known demand, we have developed a model to determine the minimum number of PIVs required to satisfy all requests in a facility during a shift. A related objective is to determine the routes for the PIVs such that the total travel distance is minimized. The following assumptions underlie the formulation.

1. Demand is deterministic and known for the week. It is specified by a pickup point, a dropoff point, and a time window that indicates the interval during which service may begin.

2. Each day is divided into three 8- or 8.5-hour shifts, where the latter have a ½ hour overlap.
3. A *zone* is defined as a set of control points and a *segment* is defined as the pair (pickup point, dropoff point). A time window, denoted by $[a,b]$, is associated with each segment, where a is the earliest time and b is the latest time that we can begin moving the mail for the segment under consideration.
4. PIVs are assigned to zones. All segments whose pickup and dropoff points are within the same zone must be serviced by a PIV assigned to that zone. A segment whose pickup point is in zone k and whose dropoff point is in zone l may be serviced by either a PIV assigned to zone k or l. The latter case might arise when a PIV, which is assigned to zone l, first services a segment with origin in l and destination in k, and then picks up a load in k to deliver to l on the return trip. Note that the PIV on its return trip to l can service at most one request with pickup point in k and dropoff point in l. Other types of requests are not allowed on its return trip. For example, the intermediate case in which a PIV assigned to zone l picks up a load in zone k and drops it off in zone k on its return from zone k to zone l is not allowed.
5. A PIV can service only one segment at a time; i.e., it cannot move more than one load at a time.
6. PIVs start and end a shift at a depot or parking area within the facility. A depot may be assigned to one or more zones. Once a PIV leaves its depot to transfer mail, it doesn't return until the shift is over. A post-processor can be used to determine when a PIV will actually spend its idle time at its depot or at a control point waiting for the mail at its next pickup point to be processed.
7. Lunch breaks and rest breaks are not included. However, it should be straightforward to require that each PIV route contain a ½-hour lunch break within some specified interval during a shift.
8. Because demand varies from one shift to another and from one day to another, the minimum number of PIVs is a function of the particular shift and day of the week.

It is management's responsibility to decide the level of demand to plan for. We provide results for the shift with the highest demand as well as several other shifts.

In the development of the model, it is convenient to view each segment as node in a graph. The time windows impose a partial ordering on the nodes, which are represented by directed arcs. If the time windows of, say, nodes *i* and *j* are wide enough so it is possible to visit them in either order, then two arcs are needed; one from *i* to *j* and one from *j* to *i*. For each shift *t*, the objective is to determine v_t^{PIV} such that each node is visited exactly once and all time windows are respected.

The following notation is used in the construction of the model.

Indices and sets

- i, j indices for segments (nodes)
- k, l indices for zones
- 0_k index for a dummy segment that starts and ends at the depot for zone k
- K set of zones
- S_k set of segments whose pickup and dropoff points are both in zone k
- S_{kl} set of segments whose pickup point is in zone k and whose dropoff point is in zone l
- $\overline{S}_k$ set of segments whose pickup point is in zone k and whose dropoff point is not in zone k;

  $\overline{S}_k = \bigcup_{l \in K \setminus \{k\}} S_{kl}$
- $\overline{\overline{S}}_k$ set of segments whose pickup point is not in zone k and whose dropoff point is in zone k;

  $\overline{\overline{S}}_k = \bigcup_{l \in K \setminus \{k\}} S_{lk}$

Parameters

- ε objective function penalty coefficient for travel time ($\varepsilon = 0.01$)
- a_i beginning of time window for segment i
- b_i end of time window for segment i

σ_i time to travel from the pickup point to the dropoff point of segment i (processing time) $\sigma_{0_k} = 0, \forall k \in K$

τ_{ij} time to travel from dropoff point of segment i to pickup point of segment j

M_{ij} large positive constant associated with arc (i,j); $M_{ij} = \sigma_i + \tau_{ij} + b_i - a_j$

Decision variables

x_{ij}^k 1 if segment i is the immediate predecessor of segment j on a route that starts from depot k, 0 otherwise

t_i time at which a PIV arrives at the pickup point of segment i

In homogeneous vehicle routing formulations, a binary variable is typically used to represent flow between two nodes so only two indices (i and j) are needed. In our problem, the rules for cross-zone movement require an additional index to indicate the assigned zone of the PIV that connects i and j. To see this, consider a two-subscript formulation, and let x_{ij} be 1 if segment i is the immediate predecessor of segment j on a route, and 0 otherwise. Constraints (4.7) and (4.8) ensures that each segment is followed by either another segment or the depot.

Segment i has pickup and dropoff points in zone k

$$\sum_{j \in S_k \cup \bar{S}_k \cup \{0_k\}} x_{ij} = 1 \qquad \forall\, k \in K,\ i \in S_k \tag{4.7}$$

Segment i has pickup point in zone k and dropoff point in zone l

$$\sum_{j \in S_k \cup \bar{S}_k \cup S_{lk} \cup \{0_k\}} x_{ij} = 1 \qquad \forall\, k \neq l \in K,\ i \in S_{kl} \tag{4.8}$$

Referring to the example in Figure 4.3, suppose depot 1 connects to segment h which connects to segment i. Then, a PIV from Depot 1 can pick up any load in zone 1 after it services segment i if the time window constraint is satisfied. However, for the case where k is zone 2 and l is zone 1, without any additional information, (4.8) requires segment i to connect with a segment in $S_2 \cup \bar{S}_2 \cup S_{1,2}$. This is incorrect because segment i is really on a route that originates in zone 1 and should connect to a segment in

$S_1 \cup \bar{S}_1 \cup \{0_1\}$. This ambiguity occurs because the two-subscript formulation does not indicate whether segment i is being serviced by a PIV from zone 1 or from zone 2.

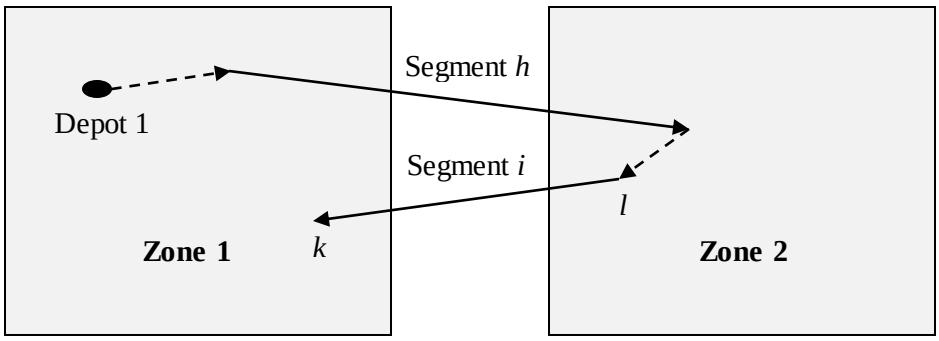


Figure 4.3. Example of ambiguity for two-subscript model

The following formulation relies on a third subscript to indicate the origin of the PIV that connects nodes i and j.

Exact model

$$\text{Minimize} \quad \sum_{k\in K}\sum_{j\in S_k\cup\bar{S}_k} x^k_{0_k,j} + \varepsilon\sum_{k\in K}\left(\sum_{i\in S_k\cup\bar{S}_k\cup\bar{\bar{S}}_k}\sum_{j\in S_k\cup\bar{S}_k\cup\{0_k\}}\tau_{ij}x^k_{ij} + \sum_{l\in K\setminus\{k\}}\sum_{i\in S_{kl}}\sum_{j\in S_{lk}}\tau_{ij}x^k_{ij}\right) \quad (4.9)$$

Subject to

Segment j has pickup and dropoff points in zone k

$$\sum_{i\in S_k\cup\bar{S}_k\cup\bar{\bar{S}}_k\cup\{0_k\}} x^k_{ij} = 1 \qquad \forall k\in K,\ j\in S_k \quad (4.10)$$

Segment j has pickup point in zone k and dropoff point in zone l

$$\sum_{i\in S_k\cup\bar{S}_k\cup\bar{\bar{S}}_k\cup\{0_k\}} x^k_{ij} + \sum_{i\in S_{lk}} x^l_{ij} = 1 \qquad \forall k\neq l\in K,\ j\in S_{kl} \quad (4.11)$$

Flow balance for segment j that has pickup and dropoff points in zone k

$$\sum_{i\in S_k\cup\bar{S}_k\cup\bar{\bar{S}}_k\cup\{0_k\}} x^k_{ij} - \sum_{h\in S_k\cup\bar{S}_k\cup\{0_k\}} x^k_{jh} = 0, \quad \forall k\in K,\ j\in S_k \quad (4.12)$$

Flow balance for segment $j\in S_{kl}$ and a PIV from zone k

$$\sum_{i\in S_k\cup\bar{S}_k\cup\bar{\bar{S}}_k\cup\{0_k\}} x^k_{ij} - \sum_{h\in S_k\cup\bar{S}_k\cup S_{lk}\cup\{0_k\}} x^k_{jh} = 0, \qquad \forall k\neq l\in K,\ j\in S_{kl} \quad (4.13)$$

Flow balance for segment $j \in S_{kl}$ and a PIV from zone l

$$\sum_{i \in S_{lk}} x_{ij}^{l} - \sum_{h \in S_l \cup \bar{S}_l \cup \{0_l\}} x_{jh}^{l} = 0, \qquad \forall k \neq l \in K, j \in S_{kl} \tag{4.14}$$

Elimination of subtours: i has pickup and dropoff points in zone k

$$t_j \geq t_i + \sigma_i + \tau_{ij} - M_{ij}(1 - x_{ij}^{k}), \quad \forall k \in K, i \in S_k, j \in S_k \cup \bar{S}_k \tag{4.15}$$

Elimination of subtours: $i \in S_{kl}$ and a PIV from zone k

$$t_j \geq t_i + \sigma_i + \tau_{ij} - M_{ij}(1 - x_{ij}^{k}), \quad \forall k \neq l \in K, i \in S_{kl}, j \in S_k \cup \bar{S}_k \cup S_{lk} \tag{4.16}$$

Elimination of subtours: $i \in S_{kl}$ and a PIV from zone l

$$t_j \geq t_i + \sigma_i + \tau_{ij} - M_{ij}(1 - x_{ij}^{l}), \quad \forall k \neq l \in K, i \in S_{kl}, j \in S_l \cup \bar{S}_l \tag{4.17}$$

Time windows

$$a_i \leq t_i \leq b_i, \ \forall\ k \in K, i \in S_k \cup \bar{S}_k \tag{4.18}$$

Integrality

$$x_{ij}^{k} \in \{0,1\}, \ \forall\ k \in K \text{ such that } <i,j> \text{ is feasible for zone } k \tag{4.19}$$

The first term in (4.9) represents the total number of PIVs that leave their assigned depot at the beginning of a shift, and is intended to be minimized for a given demand. The second term accounts for the total distance traveled between segments. Because □ is arbitrarily small, a solution will give the minimum total distance among the alternatives that minimize the number of PIVs. Note that the second term separates within zone movements from cross-zone moments.

For consistency of notation, 0_k is defined as a segment of zero length whose origin and destination are the depot associated with zone k. The first two constraints, (4.10) and (4.11), ensure that each segment follows either another segment or the depot. When segment j follows 0_k, the implication is that a new PIV is needed. In the case of (4.10), both control points of segment j are in zone k. Either the pickup point, the dropoff point, or both points of segment i are in zone k. In the case of (4.11), the pickup point of segment j is in zone k but the dropoff point is in zone l. The segment j is served by a PIV either from zone k or l. In the former case, segment j can follow any segment with

pickup or dropoff point in zone k if time windows permit. In the latter case, the PIV services segment j during its return trip back to zone l and segment j can only follow a segment from S_{lk}.

The next three constraints, (4.12) – (4.14), ensure that each segment is preceded by either another segment or the depot. In either case there are four possibilities for segment i: it can be the initial trip of a route, it can have both control points in k, it can have the pickup point in k but not the dropoff point, or it can have the pickup point in l and the dropoff point in k. The last situation reflects the opportunity to return to zone k with a load, rather than as a deadhead.

Constraints (4.15) – (4.17) ensure that time is increasing on any route. They state that if segment i immediately precedes segment j on a route, then the time that the PIV arrives at the pickup point for j must be greater than or equal to the time that it arrives at the pickup point for i, plus the time to service i, plus the time to travel from the dropoff point of i to the pickup point of j; i.e., $t_j \geq t_i + \sigma_i + \tau_{ij}$. Notice that it is always possible to build the service time for segment i into the travel time from i to j and define a single parameter $\hat{\tau}_{ij} \equiv \sigma_i + \tau_{ij}$ for all feasible i and j. Because (4.15) – (4.17) guarantee that if i immediately precedes j on a route, then $t_j > t_i$, so it is not possible to service segment i a second time and hence create a subtour in a solution.

Constraints (4.18) state that the time at which a PIV arrives at the pickup point of segment i must be within its time window. The last constraint, (4.19), places logical restrictions on the routing variables.

Model (4.9) – (4.19) is a mixed-integer linear program (MILP) and has proven to be extremely difficult to solve for instances of practical size. Much research has been done over the last 25 years on trying to find efficient solution methodologies; a review given in Berbeglia et al. (2007). The tighter the time windows the larger the problem instances that can be solved to a guaranteed minimum. At least without cross-zone movement restrictions, experience has shown that instances with up to 100 segments per shift can be solved without too much difficulty as long as the time windows are tight (e.g., see Bard et

al. 2002, Li and Lim 2003, Bent and Van Hentenryck 2006). In addition, there are many good heuristics that often give optimal solutions (e.g., see Nanry and Barnes 2000).

4.5. Solution Strategies

To find solutions to the clustering problem, we used the greedy randomized adaptive search procedure (GRASP) proposed by Deng and Bard (2011). In phase I of the GRASP, both a heaviest weight edge algorithm and a constrained minimum cut algorithm are called on to select seeds for initializing the $|K|$ clusters. Feasible solutions are obtained with the help of a self-adjusting restricted candidate list that sequentially guides the assignment of the remaining nodes. In phase II, three neighborhoods, each defined by common edge and node swaps, are explored to attain local optima. Exploration strategies included cyclic neighborhood search, variable neighborhood descent, and randomized variable neighborhood descent. The best solutions found are stored in what is called an *elite* pool. In a post-processing step, path relinking is applied to the pool members to cyclically generate paths between each pair with the expectation of uncovering improved solutions.

This paper mainly focuses on determining the minimum number of PIVs needed and the optimal route for each for a given set of clusters. Our first attempt to solve model (4.9) – (4.19) with CPLEX showed that only small instances with no more than 30 requests were tractable. For larger instances, feasible solutions were found quickly but the optimality gap remained in double and triple digits. In most cases, instances with 200 or more requests exceeded the memory of our server. As an alternative, we developed a column generation (CG) scheme and two versions of a rolling horizon heuristic. Our column generation formulation is given in Appendix A. Using (4.10) and (4.11) to form the master problem, our original thought was that the $|K|$ subproblems formed from the remaining constraints would be easy to solve at each iteration due to tight time windows and reduced connections imposed by the cross-zone movement constraints. This turned out not to be the case. With a 300-sec time limit for each subproblem, we tested instances with 30, 60, 120 and 240 requests. For the smaller instances with 30 and 60 requests, the CG algorithm converges within 4 to 13 hours of CPU time, and provides

good lower bounds. However, when problem instances with 120 requests are considered, CG cannot solve the subproblems, or even find a feasible solution with negative objective function value within the given time limit (300 sec). Computational results are highlighted in Appendix A. In light of our inability to solve the relaxed master problem at the root node of the search tree, even for medium size problems, we abandoned column generation in favor of heuristic approaches.

4.5.1 Rolling horizon heuristic I – fixed timespan

We are interested in solving model (4.9) – (4.19) for a single shift which could include more than 1400 requests on a busy day. To make the problem more manageable, we developed a rolling horizon heuristic in which a shift is divided into fixed-length periods and a subproblem is solved for each period sequentially. Although the period length L is a parameter, after extensive testing we settled on a value of two hours as a compromise between runtime and solution quality. Our first heuristic begins by solving the subproblem for all segments whose earliest pickup time is within the first period (i.e., the first two hours of the shift). All requests that are on a route whose earliest pickup time is in the first hour are fixed whether they are completed or remain to be completed in the second hour. This means that if a segment has a time window that crosses from the first hour into the second hour it is also fixed regardless of its pickup time. Next, the time horizon is extended by half a period (one hour) and the next subproblem is solved (i.e., the subproblem associated with hours two and three is solved).

When the second subproblem is solved, all routes starting from the depots and ending at the end point of the requests whose earliest start times are contained within the first half of the first period are fixed. Thus, the size of the second subproblem is roughly the same size as the first. For an 8-hour shift and a 2-hour period, a total of 7 subproblems must be solved. Figure 4.4 illustrates the major steps in the algorithm referred to as *rolling horizon heuristic I*. In the figure, the timespan of each subproblem is denoted by L, the time increment is $L/2$, and the end time of each subproblem is denoted by *endTime*.

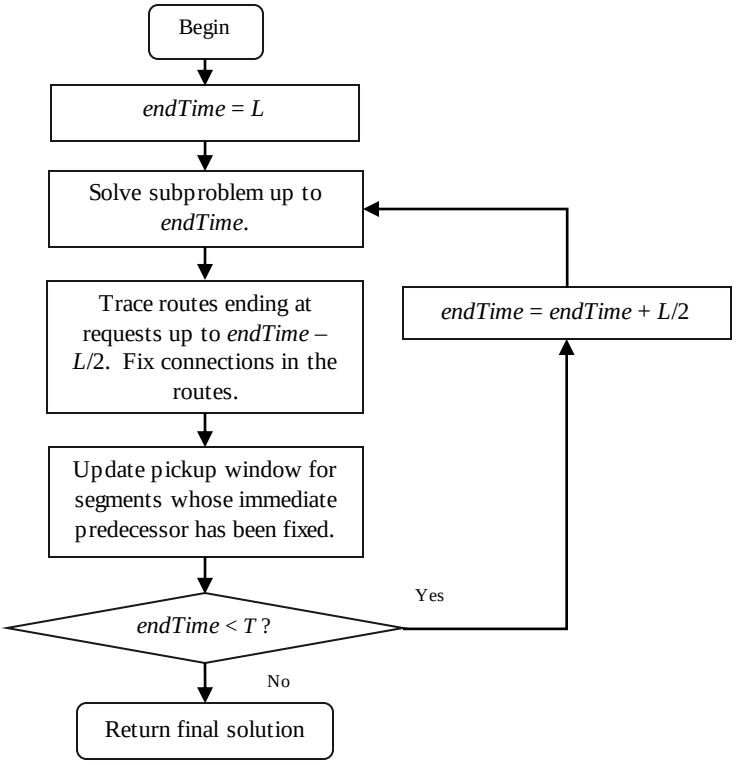


Figure 4.4. Rolling horizon heuristic I for fixed timespan

The pseudocode below highlights the steps in rolling horizon heuristic I. This is followed by an explanation of Procdedure_FixConnections in which the segments are fixed after a subproblem is solved.

Rolling_Horizon_Heuristic_I

Notation

T length of a shift in (1-hour) periods

L length of subproblem timespan (2 hours)

endTime end time of current subproblem

[*endTime–L, endTime*] timespan of current subproblem

FI set of segments (nodes) whose predecessor and successor have been fixed

FJ set of segments (nodes) whose predecessor has been fixed but successor has not

J set of segments whose earliest pickup time is in [*endTime–L, endTime–L*/2]

T_i value of decision variable t_i for segment i determined by solving current subproblem

Input: segments in a shift

Output: number of PIVs needed and their routes for a shift

Begin

Set $FI = \phi$, $FJ = \phi$

$endTime = L$

Solve subproblem for segments in interval [$endTime–L$, $endTime$]

Let J = set of segments whose earliest pickup time is in [$endTime–L$, $endTime–L/2$]

Call Procdedure_FixConnections to update FI and FJ

while ($endTime < T$){

$endTime \leftarrow endTime + L/2$

Solve subproblem with segments in interval [$endTime–L$, $endTime$]

Let J = set of segments whose earliest pickup time is in [$endTime–L$, $endTime–L/2$]

Call Procdedure_FixConnections to update sets FI and FJ

for every segment $i \in FJ$

{

$a_i \leftarrow \max(a_i, T_i)$

}

End

The procedure starts with the sets FI and FJ empty. The first subproblem includes all segments whose earliest pickup time is in [0, L]. After solving, we call *Procdedure_FixConnections* to update FI and FJ. That is, we fix all connections from the depot and from segments in the first subproblem to segments whose earliest pickup time is in [0, $L/2$]. The next subproblem is then solved over the interval [$L/2$, $3L/2$]. Before solving this subproblem, though, we need to update the pickup time windows for segments whose immediate predecessor has been fixed. If a PIV arrives at segment i for a pick up at T_i which is later than the earliest pickup time a_i, then its earliest pickup time for the following subproblem should be T_i. We only need to update the pickup time

windows for segments in FJ because those segments will connect with segments in subsequent subproblems. Successors of segments in FI have been fixed. Note that solutions with different T_i values could have the same objective function value. To find the smallest value of t_i when there are multiple optimal solutions, we add t_i to the objective function and multiply by an extremely small coefficient; e.g., 0.00001[see (4a) below]. The subproblem formulation is presented at the end of this subsection.

As time rolls forward, the next subproblem is solved and a subset of segments is fixed. The procedure continues until the ending time of a subproblem exceeds the time horizon T of the original problem. At termination, each segment is assigned to exactly one route and all time windows are satisfied.

Figure 4.5 illustrates the timespans over which the "current" subproblem, the "previous" subproblem, and the "next" subproblem are solved. It also indicates the connections that are fixed after solving the current subproblem. From the figure, we can see that the end time of the previous subproblem is equal to the start time of the next subproblem.

Choosing the value of L, the length of a subproblem, requires careful consideration. If L is too large, the subproblems become intractable. If L is too small, the quality of the solution to the original problem is likely to be poor. To provide a rationale for choosing L, we consider a worst case scenario in which the required number of PIVs determined in each subproblem may be greater than the optimal value because the planning period is too short.

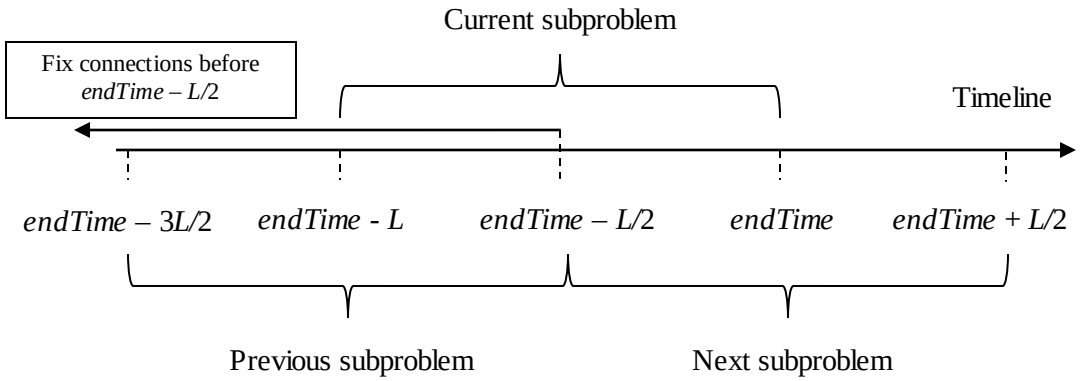


Figure 4.5. Timespans of subproblems

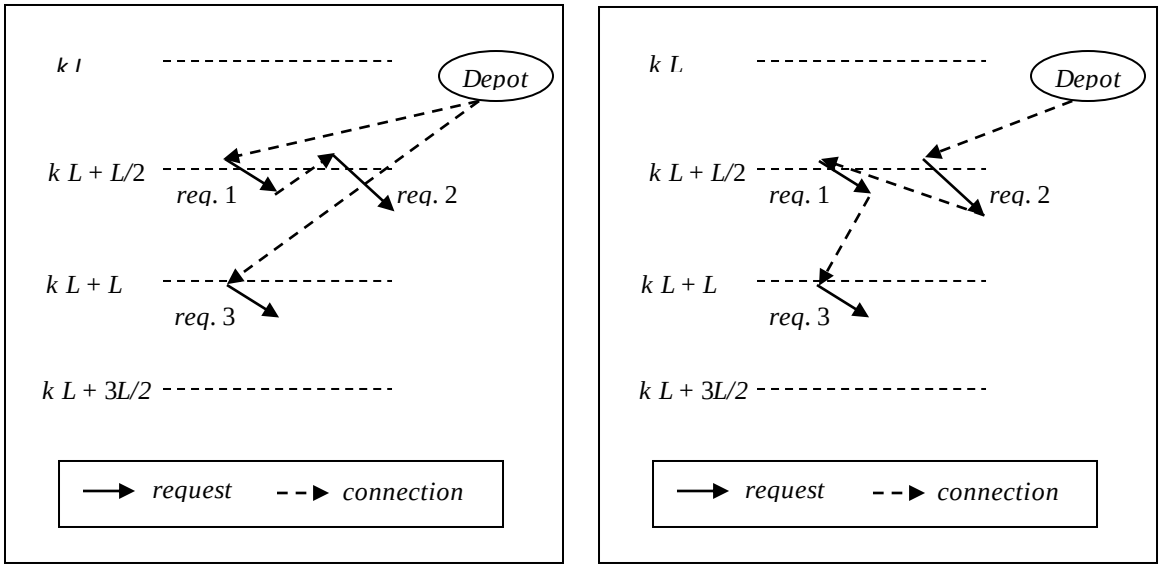
Definition. *Worst case scenario*. Suppose that two consecutive subproblems are defined over the time intervals $[k \cdot L, (k+1) \cdot L]$ and $[k \cdot L + L/2, (k+1) \cdot L + L/2]$, respectively, for any $k = 0, 1, 2\ldots$ Then a segment j whose earliest pickup is $(k+1) \cdot L$ would belong to the latter subproblem. Now assume that there is an existing PIV h whose most recently serviced segment g has earliest pickup time in $[k \cdot L, k \cdot L + L/2]$. After solving the first subproblem, PIV h's route has been fixed up to the segment g. In the worst case, segment j cannot be serviced by PIV h because the arrival time at the dropoff point of segment g plus the transit time from that last dropoff point to segment j's pickup point exceeds the time window of segment j. In this situation, segment j must be serviced by either a new PIV or an existing PIV other than h. ■

The underlying difficulty that we face is that each subproblem only considers segments within a timespan of L hours. Therefore, the subproblem defined over the increment $[k \cdot L, (k+1) \cdot L]$ does not consider requests that start at or after $(k+1) \cdot L$ (e.g., request j in the above definition). As a consequence, the routes that have been fixed up to time $k \cdot L + L/2$ (halfway through the previous subproblem) may not be good starting points for requests after $(k+1) \cdot L$. As such, additional PIVs may be required that wouldn't necessarily be required if the fraction of the timespan L in which segments are fixed after a subproblem is solved is less than $L/2$. This would allow more time for the current PIVs to arrive at the pickup points of the segments in the next subproblem. We denote the length of the first period as FP and the length of second period at UP, where

$FP + UP = L$. Since our algorithm by design fixes segments in the first half of the horizon, we have $FP = UP = L/2$.

A worst case scenario is illustrated in Figure 4.6 for an instance with three requests. The first subproblem includes requests 1 and 2 and the second subproblem includes request 3 [see panel (a)]. When the problem is solved exactly, all three requests are taken together [see panel (b)]. Assume that the transit time from the dropoff point of request 1 to request 3's pickup point is much shorter than the time from the dropoff point of request 2 to request 3's pickup point.

In Figure 4.6a, the route in the first subproblem solution starts from the depot and services request 1 and request 2 in sequence. Assume this route has the shortest total travel time. Since the earliest pickup time of requests 1 and 2 are before $k \cdot L + L/2$, this route is fixed and the corresponding PIV is available at the start of the next subproblem. For that PIV to be able to service request 3, the arrival time at the dropoff point of request 2 plus the transit time to request 3's pickup point cannot exceed the time window of request 3. If this condition is not met, then a second PIV is needed to service request 3 in the second subproblem. In contrast, when all three requests can be considered at once, Figure 4.6b shows that only a single PIV is needed.



(a) Worst case scenario (b) Optimal solution

Figure 4.6. Illustration of a worst case scenario

For rolling horizon heuristic I to avoid encountering the worst case scenario, the timespan during which segments are not fixed in each subproblem [e.g., $(kL + L) - (kL + L/2) = L/2$ in Figure 4.6] must be sufficiently large. The following proposition provides a lower bound on this value.

Proposition 1. Let α be the maximum transit time between all pairs of control points in a P&DC. Then, to avoid the *worst case scenario*, the minimum length of the period, *UP*, during which segments are not fixed for any subproblem is 2α.

Proof. For PIV h that services a request with earliest pickup time just before $k \cdot L + L/2$ (e.g., request 2 in Figure 4.6), it takes at most α minutes to pick up and drop off the request. Assume that the pickup time window for all requests is β minutes. To avoid the worst case scenario, the PIV must be able to service a request j with earliest pickup time at $k \cdot L + L$ (e.g., request 3 in Figure 4.6). To accomplish this, the latest time that PIV h can arrive at its most recent dropoff point (say, $k \cdot L + L/2 + \alpha + \beta$) plus the transit time from the last dropoff point to request j's pickup point (say, α) should be less than the latest pickup time of request j (say, $k \cdot L + L + \beta$). Therefore, $k \cdot L + L/2 + \alpha + \beta + \alpha < k \cdot L + L + \beta$, which is equivalent to $2\alpha < L - L/2$. By design, $UP = L - L/2$ in each subproblem so we conclude that $2\alpha < UP$. ■

Noting that 2000 feet is a practical upper bound on the maximum distance between all pairs of pickup and dropoff points in a P&DC, and that the travel speed of a PIV is 5 feet per second, $\alpha = 6.6$ min. Therefore, *UP* should be greater than 13.2 min and L should be greater than 26.4 min. Choosing the value of parameter L is critical in our heuristics. However, CPU time grows exponentially with subproblem size so for practical purposes, the timespan L must be limited to a fraction of a shift.

In the computational section, we discuss our choice of L. Ordinarily, one might conjecture that the number of PIVs required would be a non-increasing function of L, especially when all the subproblems were solved to optimality. This turns out not to be the case as we now show.

Proposition 2. Due to cross-zone movement restrictions, increasing values of the subproblem timespan L do not necessarily lead to solutions that require fewer PIVs to service all demand.

Proof. The result will be confirmed by way of example. Let case I correspond to $L = 30$ min and let case II correspond to $L = 60$ min. Assume that the facility is divided into two zones, all pickup time windows are 5 min, every request has 1 min processing time, and that the planning horizon is 2 hours. The first two requests have $a_1 = a_2 = 0$. The third and the fourth requests have $a_3 = a_4 = 35$ min, which means they belong to the second subproblem in the case I but the first subproblem in the case II. The segments associated with the first and fourth requests have pickup point in zone 1 and dropoff point in zone 2, while the second and third segments have pickup point in zone 2 and end point in zone 1. Segments and their control points are shown in Figure 4.7.

In the first subproblem of case I, only the first two requests are considered. Assume that the optimal solution is to use a new PIV from zone 1 to service these requests. Suppose that the travel time from the zone 1 depot to the first request is 1 min and the travel time from the dropoff point of request 1 (i.e., control point 2) to the pickup point of request 2 (i.e., control point 3) is 1 minute. The total travel time is then 1 + 1 + 1 + 1 = 4 min.

In the first subproblem of case II, segments 1 through 4 are considered. Assume that the optimal solution is to use a new PIV from zone 2 to service request 2, request 1, request 3, and request 4 in this order. Suppose that the travel time from the zone 2 depot to the request 2 is 2 min, the travel time from the dropoff point of request 2 (i.e., control point 4) to the pickup point of request 1 (i.e., control point 1) is 1 min, the travel time from the dropoff point of request 1 (i.e., control point 2) to the pickup point of request 3 (i.e., control point 5) is 1 min, and the transit time from control point 6 to 7 is 1 min. On the path starting at depot 2, the travel time between segments is 2 + 1 + 1 + 1 = 5 min, and the travel time along the four segments is 4 min.

Now consider a second path for case II starting at depot 1 that connects requests 1 through 4. Suppose that the travel time from the dropoff point of request 2 (i.e., control

point 4) to the pickup point of request 3 (i.e., control point 5) is 5 min. The travel time between segments on the path from depot 1 takes 1 + 1 + 5 + 1 = 8 min, and the travel time for serving the four requests again takes 4 min. Thus, the second path is suboptimal.

Returning to case I, in the second subproblem, the third and fourth segments are considered. The existing PIV from depot 1 is assigned to service request 3 first and then request 4. Now we add a fifth request with $a_5 = 65$ min that has pickup point and dropoff point in zone 1. This request belongs to the second subproblem in case II and requires a second PIV from zone 1 to serve it. For case I, however, an additional PIV is not needed. Thus, we have shown that case II with a larger L requires two PIVs while case I with smaller L requires only one PIV. ■

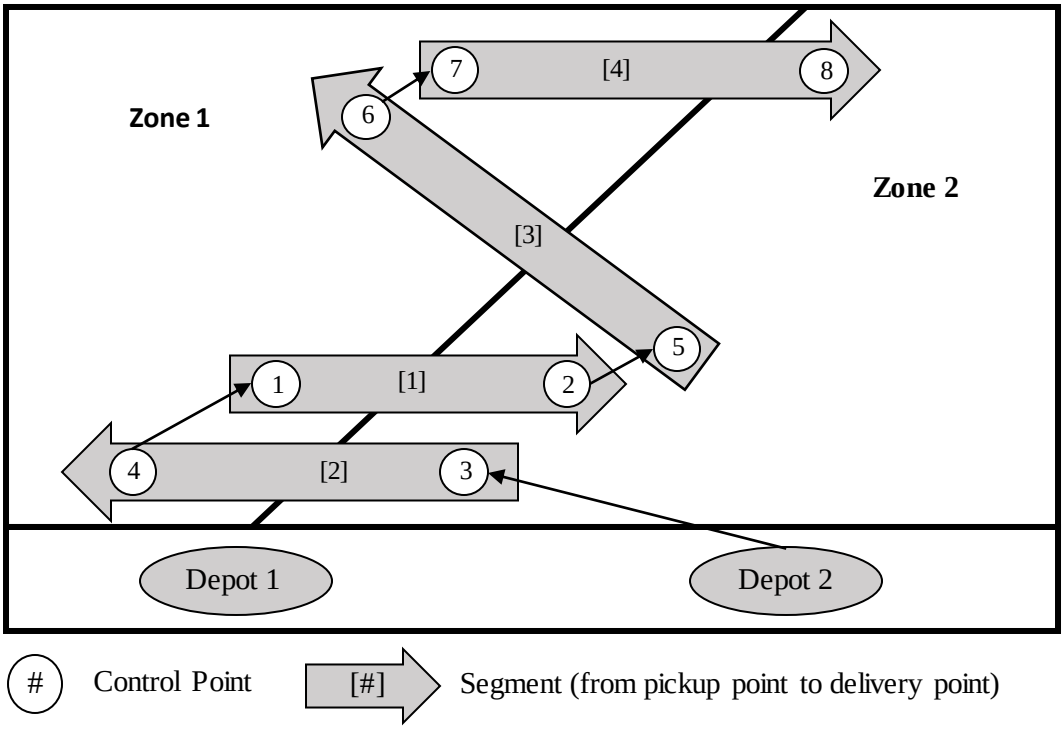


Figure 4.7. Optimal solution for the first subproblem for case II

We also have the following result.

Proposition 3. Due to cross-zone movement restrictions, when the subproblems have multiple optimal solutions, there is no guarantee that rolling horizon heuristic I will find the best solution for a given set of parameter values.

Proof. Again, the result will be confirmed by way of example. Assume that the facility is divided into two zones, $L = 60$ min, all time windows are 5 min, three requests must be serviced, and that the planning horizon is 2 hours. The first two requests have $a_1 = a_2 = 0$, and the third request has $a_3 = 84$ min. The segment associated with the first request has pickup point in zone 1 and dropoff point in zone 2, while the second segment has pickup point in zone 2 and end point in zone 1. Assume that it takes 1 min to service the first two segments, and that the travel time from one segment's dropoff point to the other segment's pickup point is 2 min.

For case I, assume that the solution to the first subproblem requires 1 PIV from zone 1, which services the segment from zone 1 to zone 2 first and then the segment from zone 2 to zone 1. This takes 1+2+1 = 4 min in total.

For case II, we have a symmetric solution that now uses 1 PIV from zone 2, which services the segment from zone 2 to zone 1 first and then the segment from zone 1 to zone 2. This also takes 1+2+1 = 4 min. Thus, both solutions are optimal. Now suppose that the segment with opening time at 84 starts and ends in zone 1, which means that it can only be serviced by a PIV from zone 1. Therefore, for the corresponding subproblem, case II requires a second PIV from zone 1 while only a single PIV is required for case I. ■

The examples in these proofs indicate that optimal subproblem solutions in the early periods of a shift do not guarantee the best possible solution that the heuristic can provide to the original problem. As we have shown, decisions in the earlier periods of a shift may restrict the options available in each zone in the latter periods of a shift. This can lead to a "suboptimal" overall solution.

Fix connections

Due to the pickup time windows, we cannot simply fix all connections $<i,j>$ such that the earliest pickup time of segment j is no later than $endTime - L/2$ (see Figure 4.4). Figure 4.8 gives an example of what can go wrong. Here, the route starts at the depot and connects segment 1, 2 and 3. Even though it looks like the route backtracks in time, this is not the case because the time window of segment 3 is assumed to be late enough to

allow a pickup after segment 2 is serviced. If we only fix connections $<0_k,1>$ and <2,3>, where the earliest pickup times of segments 1 and 3 are no later than *endTime* – *L*/2, but not connection <1,2> because the pickup time of segment 2 is greater than *endTime* – *L*/2, we would have two issues. First, segment 3 would not be a valid route because it has no predecessor. Second, in the next subproblem, segment 1 would be free to connect to segments other than segment 2. This would be suboptimal because the solution to the first subproblem indicates that segments 2 and 3 should be serviced by the same PIV that services segment 1.

To avoid these problems, we start from segment *j* whose earliest pickup time is no later than *endTime* – *L*/2, trace its route back to the depot, and fix all connections along the route. For the example in Figure 4.8, we start with segment 3 and fix connections <2,3>, <1,2> and $<0_k,1>$. More generally, Procedure_FixConnections indicates how to fix connections and update the sets *FI* and *FJ*, which provide the initial conditions or starting routes for the next subproblem.

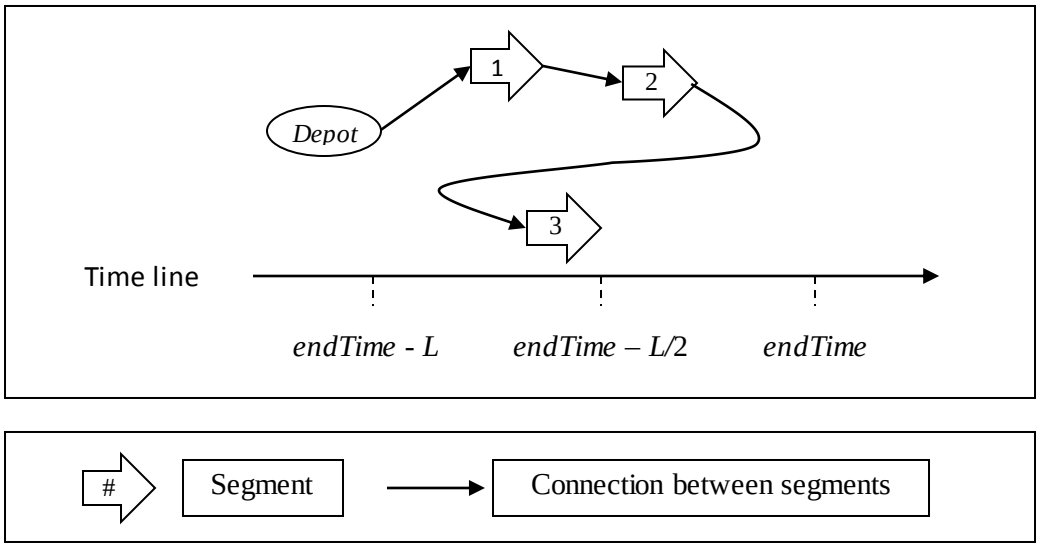


Figure 4.8. Example of fixing connections in current subproblem

Procedure_FixConnections

Input: Value of connection variables in the current subproblem; sets *FI*, *FJ* and *J*

Output: Updated sets FI and FJ containing segments that have been fixed in the current subproblem

Begin

Sort the elements of J in descending order of their earliest start time

for every segment $j \in J$

{

if ($j \notin FI \cup FJ$)

{

$FJ \leftarrow FJ \cup \{j\}$

Find the predecessor of j in current route; denoted as i

while ($i \notin FI \cup FJ$)

{

$FI \leftarrow FI \cup \{i\}$

Put $i \leftarrow$ predecessor of i in current route

If (i = depot), go to next $j \in J$

}

if ($i \in FJ$)

{

$FJ \leftarrow FJ \setminus \{i\}$

$FI \leftarrow FI \cup \{i\}$

}

}

}

End

For each segment $j \in J$, the algorithm takes constant time to check if j is in the sets FI or FJ. Then, up to N operations are required to find the immediate predecessor of j, where N is the total number of segments. Since the algorithm will not visit the same segment more than once, the complexity of the algorithm is $O(M \cdot N)$, where M is the number of segments in J.

Formulation of the subproblem for rolling horizon heuristic I

As opposed to model (2), there is no requirement when solving all but the last subproblem that a route return to its depot. Moreover, if connections have been fixed in previous subproblems, then the corresponding variables remain fixed as the iterations progress. In rolling horizon heuristic I, *FI* is the set of segments whose predecessor and successor are both fixed, and *FJ* is the set of segments in the previous subproblem whose predecessor is fixed but whose successor is not. Therefore, the number of segments in *FJ* is equal to the number of existing PIVs.

After we solve subproblem (4a) – (4k) we only fix the connections that end at segments whose earliest pickup time is in [*endTime* – *L*, *endTime* – *L*/2]. However, because the time window of some segment *j* may cross *endTime* – *L*/2 (i.e., we may have $a_j < endTime - L/2 < b_j$) the predecessor of segment *j* may be some segment *i* with earliest pickup time later than *endTime* – *L*/2. In this case, we also need to fix the connection ending at segment *i* if we want to fix the route from the depot to segment *j*. This is the reason why the earliest pickup time of some segments in *FI* may fall within the timespan of the current subproblem.

In Procedure_FixConnections, we search backwards from a segment with the earliest pickup time in [*endTime–L*, *endTime–L/2*] to either a depot or a segment whose predecessor has been fixed. Segments in $FI \cup FJ$ are those whose predecessor has been fixed. Therefore, when solving the current subproblem, we only need to service segments whose earliest pickup times are within [*endTime–L*, *endTime*] but not in *FI*.

In defining each subproblem, the first component of the objective function is aimed at minimizing the number of additional PIVs needed for each shift, while existing PIVs are free to use. This represents a modification of (3a). The second component in the objective function minimizes travel time in the subproblem timespan [*endTime–L*, *endTime*].

The following definitions are needed to formulate a subproblem.

π objective function penalty value for arrival time t_i ($\pi = 0.00001$)

f_i lower bound on pickup time for segment i; if the predecessor of segment i has been fixed, then f_i is the pickup time from the previous subproblem solution, otherwise $f_i = a_i$

ST set of segments whose earliest pickup time is within [*endTime–L, endTime*]

SI set of segments whose immediate successor has not been fixed when solving the current subproblem; $SI = FJ \cup \{ST \setminus FI\}$

SJ set of segments that are in [*endTime–L, endTime*] and whose immediate predecessor needs to be determined when solving the current subproblem; $SJ = ST \setminus FI$

The sets SI and SJ restrict the set of predecessors and successors that need to be consider in the current subproblem, respectively. When solving the current subproblem, a segment's predecessor is either a depot or a segment in SI while its successor is either free (i.e., to be determined in the next subproblem) or a segment in SJ. Note that SI includes segments whose earliest start time is before *endTime–L* and whose immediate successor has not been fixed (i.e., FJ). It also includes segments in [*endTime–L, endTime*] and segments whose immediate successor has not been fixed (i.e., $ST \setminus FI$). Segments that are in FJ cannot be in ST.

Subproblem (4a) – (4k) differs from the exact model (3a) – (3k) in that the former places greater restricts on predecessors and successors. In addition, only the last subproblem imposes the requirement that all PIVs return to the depot.

Subproblem

$$\text{Minimize} \sum_{k\in K} \sum_{j\in\{S_k \cup \bar{S}_k\}\cap SJ} x^k_{0_k j} + \varepsilon \sum_{k\in K} \left(\sum_{i\in\{S_k \cup \bar{S}_k \cup \bar{\bar{S}}_k\}\cap SI} \sum_{j\in\{S_k \cup \bar{S}_k\}\cap SJ} \tau_{ij} x^k_{ij} + \sum_{l\in K\setminus\{k\}} \sum_{i\in S_{kl}\cap SI} \sum_{j\in S_{lk}\cap SJ} \tau_{ij} x^k_{ij} \right)$$

$$+\pi \sum_{k\in K} \sum_{i\in\{S_k \cup \bar{S}_k\}\cap SJ} t_i \qquad (4.20)$$

Subject to

Segment j has pickup and dropoff points in zone k

$$\sum_{i\in\{S_k \cup \bar{S}_k \cup \bar{\bar{S}}_k\}\cap SI} x^k_{ij} + x^k_{0_k j} = 1 \qquad \forall k \in K,\ j \in S_k \cap SJ \qquad (4.21)$$

Segment j has pickup point in zone k and dropoff point in zone l

$$\sum_{i\in\{S_k\cup\bar{S}_k\cup\bar{\bar{S}}_k\}\cap SI} x_{ij}^k + x_{0_k j}^k + \sum_{i\in S_{lk}\cap SI} x_{ij}^l = 1 \quad \forall k\neq l\in K,\ j\in S_{kl}\cap SJ \tag{4.22}$$

Flow balance for segment j that has pickup and dropoff points in zone k

$$\sum_{i\in\{S_k\cup\bar{S}_k\cup\bar{\bar{S}}_k\}\cap SI} x_{ij}^k + x_{0_k j}^k - \sum_{h\in\{S_k\cup\bar{S}_k\}\cap SJ} x_{jh}^k \geq 0, \qquad \forall k\in K,\ j\in S_k\cap SJ \tag{4.23}$$

Flow balance for segment $j\in S_{kl}$ and a PIV from zone k

$$\sum_{i\in\{S_k\cup\bar{S}_k\cup\bar{\bar{S}}_k\}\cap SI} x_{ij}^k + x_{0_k j}^k - \sum_{h\in\{S_k\cup\bar{S}_k\cup S_{lk}\}\cap SJ} x_{jh}^k \geq 0,\ \forall k\neq l\in K,\ j\in S_{kl}\cap SJ \tag{4.24}$$

Flow balance for segment $j\in S_{kl}$ and a PIV from zone l

$$\sum_{i\in S_{lk}\cap SI} x_{ij}^l - \sum_{h\in\{S_l\cup\bar{S}_l\}\cap SJ} x_{jh}^l \geq 0, \quad \forall k\neq l\in K,\ j\in S_{kl}\cap SJ \tag{4.25}$$

Elimination of subtours: i has pickup and dropoff points in zone k

$$\begin{aligned} t_j &\geq t_i + \sigma_i + \tau_{ij} - M_{ij}(1-x_{ij}^k), \\ &\forall k\in K, i\in S_k\cap SI,\ j\in\{S_k\cup\bar{S}_k\}\cap SJ \end{aligned} \tag{4.26}$$

Elimination of subtours: $i\in S_{kl}$ and a PIV from zone k

$$\begin{aligned} t_j &\geq t_i + \sigma_i + \tau_{ij} - M_{ij}(1-x_{ij}^k), \\ &\forall k\neq l\in K, i\in S_{kl}\cap SI,\ j\in\{S_k\cup\bar{S}_k\cup S_{lk}\}\cap SJ \end{aligned} \tag{4.27}$$

Elimination of subtours: $i\in S_{kl}$ and a PIV from zone l

$$\begin{aligned} t_j &\geq t_i + \sigma_i + \tau_{ij} - M_{ij}(1-x_{ij}^l), \\ &\forall k\neq l\in K, i\in S_{kl}\cap SI,\ j\in\{S_l\cup\bar{S}_l\}\cap SJ \end{aligned} \tag{4.28}$$

Time windows

$$f_i\leq t_i\leq b_i,\ ,\ \forall\ k\in K,\ i\in\ \{S_k\cup\bar{S}_k\}\cap\{SI\cup SJ\} \tag{4.29}$$

Integrality

$$x_{ij}^k\ \in\{0,1\},\ \forall\ k\in K \text{ such that } <i,j> \text{ is feasible for zone } k \tag{4.30}$$

The objective function (4.20) penalizes each new PIV introduced in the current subproblem as well as the additional travel time. Existing PIVs are taken into account indirectly by the set *SI*. Constraints (4.21) and (4.22) ensure that each segment in *SJ* is connected with an immediate predecessor which is either a depot or a segment in *SI*.

They also enforce the cross-zone movement rules. Constraints (4.23) – (4.25) guarantee that the number of predecessors that a segment connects with is no less than the number of its successors. For the last subproblem, these constraints are rewritten as equalities where the set of successors includes the depots. Subtour elimination constraints (4.26) – (4.28) parallel (4.15) – (4.17) and restrict the sets of predecessors and successors of a segment. Time windows constraints (4.29) apply to segments in both *SI* and *SJ*. Integrality requirements are defined in (4.30) for each zone *k* and corresponding valid connection in the current subproblem.

4.5.2 Rolling horizon heuristic II with dynamic timespan

The difficulty of each subproblem in heuristic I is directly related to the number of requests that are considered over the *L*-hour planning horizon. Given that pickup and dropoff requests occur randomly during a shift, extending the time horizon by a constant amount does not keep the problem size constant. Our second heuristic tries to maintain a uniform problem size, and hence difficulty, by extending the time horizon up to the point at which the number of unfixed requests is roughly the same in each subproblem. Figure 4.9 identifies the basic components of the algorithm, which we refer to as *rolling horizon heuristic II*. Additional detail is provided in the pseudocode below. As in rolling horizon heuristic I, Procedure_FixConnections remains the same, but we now define the time horizon of each subproblem dynamically.

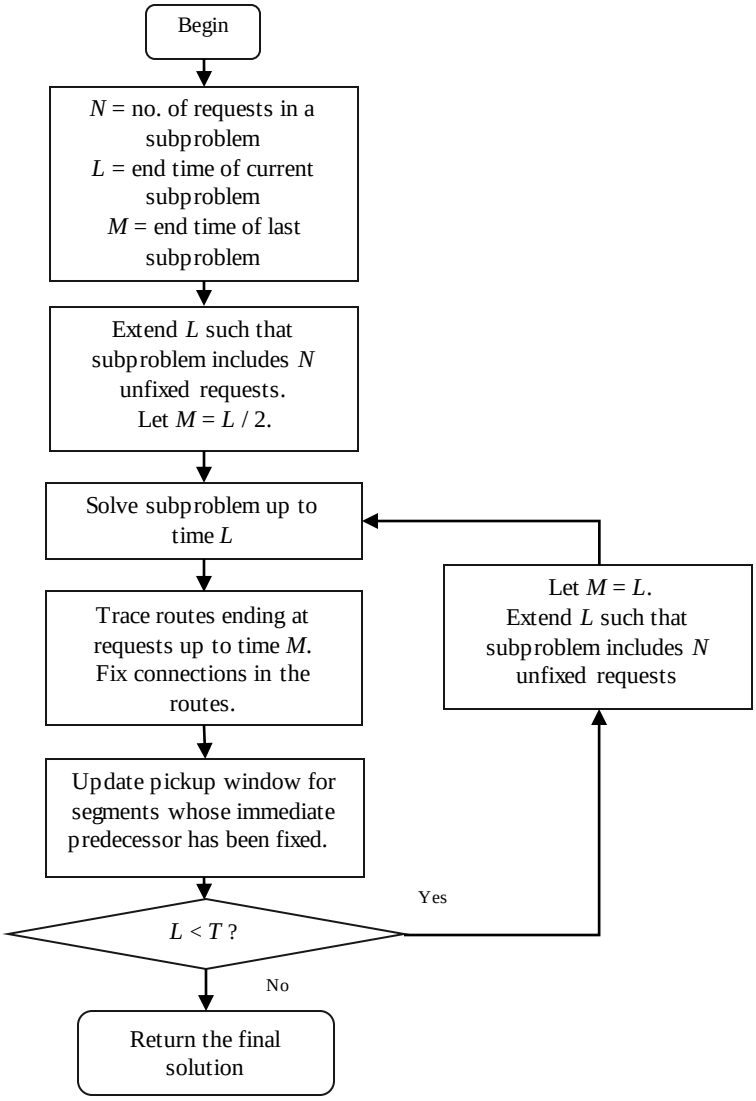


Figure 4.9. Rolling horizon heuristic II with dynamic timespan

Rolling_Horizon_Heuristic_II

Notation

T length of a shift

N number of requests (segments) in current subproblem

Input: Segments in the shift under consideration

Output: Number of PIVs needed and their routes for shift under consideration

Begin

Set $FI = \phi, FJ = \phi$

Sort all segments in increasing order of the earliest pickup time

Let M = earliest pickup time of the first $N/2$ segment

Let *endTime* = earliest pickup time of the first N segment

Solve subproblem containing segments in [0, *endTime*]

Let J = set of segments whose earliest pickup time is in [0, M]

Call Procedure_FixConnections to update FI and FJ

while ($endTime < T$){

$N \leftarrow N + N/2$

Define $temp = endTime$

endTime = earliest pickup time of the first N segments

Solve subproblem satisfying segments in [M, *endTime*]

Let J = set of segments whose earliest pickup time is in [M, *temp*]

Call Procedure_FixConnections to update FI and FJ

$M = temp$ (end time of the previous subproblem)

for every segment $i \in FJ$

{

$a_i \leftarrow \max(a_i, T_i)$

}

}

End

The first subproblem extends the horizon from 0 to *endTime*, which is defined to be the earliest pickup time of the first N segments. The connections of the first $N/2$ segments are then fixed and the time horizon is extended to include the next $N/2$ segments. This procedure iterates until the planning horizon of the original problem is reached.

The value of N determines the size of each subproblem. Similar to the analysis of choosing L for rolling horizon heuristic I, we chose N as 150 while noting a larger N does not necessarily mean better solutions, as stated in the proposition below. Table 4.5 in the

next section indicates that the length of a subproblem is about 45 min, which is larger than the lower bound derived in *Proposition* 1 on a subproblem's time horizon, L.

Proposition 4. Due to cross-zone movement restrictions, increasing the number of segments in each subproblem in rolling horizon II does not necessarily lead to solutions that require fewer PIVs to service all demand.

Proof. Let N denote the number of segments in each subproblem. Case I in the proof of the *Proposition* 2 contains 2 segments in each subproblem, which is equivalent here to $N = 2$. Case II contains 4 segments in each subproblem so $N = 4$. The same example proves the result. ■

Proposition 5. Due to cross-zone movement restrictions, when one or more subproblems have multiple optimal solutions, there is no guarantee that rolling horizon heuristic II will find the best solution for a given set of parameter values.

Proof. The example in the proof of the Proposition 3 contains 2 segments in each subproblem, which is equivalent to $N = 2$. The same example proves the result. ■

4.6. Computation Results

P&DCs operate continuously during the week. Given pickup and dropoff requests with time windows for each shift, we now try to determine the minimum number of PIVs needed to meet the maximum demand under different clustering configurations, while also minimizing total travel time. All algorithms were implemented in Java and used CPLEX 12.5 with its default settings to solve the optimization models. The computations were performed on a Linux server with a 2.3 GHz Xeon processor and 13 GB RAM.

The statistical characteristics of the input data are described Section 6.1. This is followed in Section 6.2 with a presentation of the facility layout and clustering results under different values of K. Section 6.3 provides computation results of the exact MIP model. Computational results for the rolling horizon heuristics are presented in Section 6.4 and 6.5. In the last section, we examine the sensitivity of the solution to several input parameters.

4.6.1 Input data

The data for the problem were provided by the Cardiss Collins P&DC in Chicago for a typical week. Empirically, the pickup and dropoff requests appear to be random. Figure 4.10 summarizes the number of segments in an hour during the week starting with the midnight shift on Monday, and shows a large variation in demand. Figure 4.11 consolidates the input data and identifies the number of segments in each shift, again starting on Monday morning. From the latter figure we see that number of segments on Monday is the smallest while Tuesday (shifts 4 – 6), Wednesday (shifts 7 – 9) and Friday (shifts 13 – 15) are roughly the busiest time.

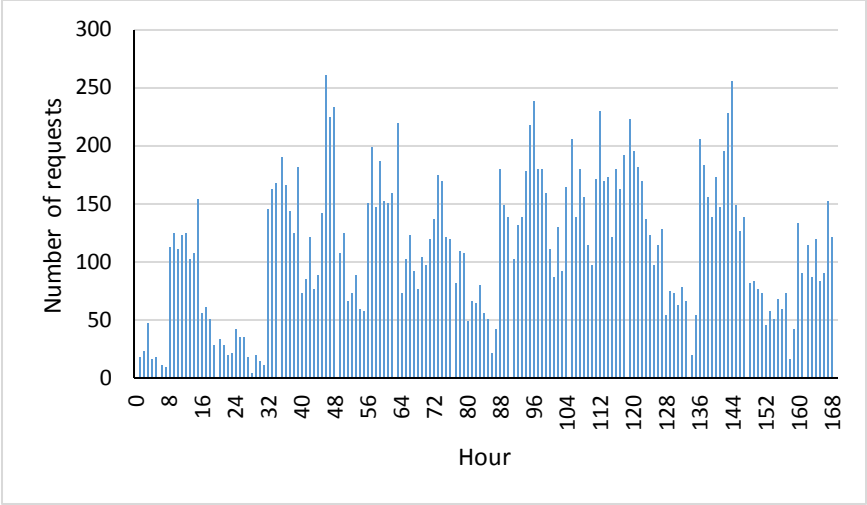


Figure 4.10. Demand for each hour in the week

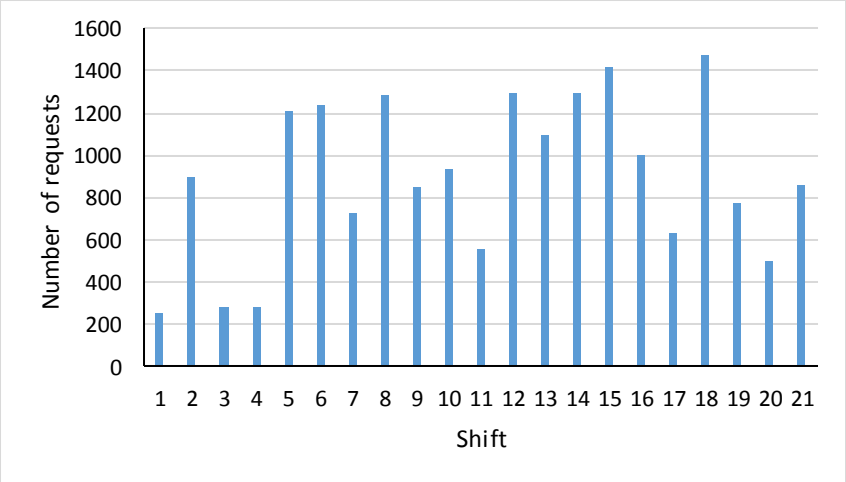


Figure 4.11. Demand for each shift in the week

4.6.2 Clustering results

Figure 4.12 depicts a map of the control points in the P&DC on a rectangular grid whose *x*- and *y*-coordinates are given in feet. In total, there are 97 control points, most being machines, operations, and workstations. Considering the vast area of the facility, node distance is critical when determining the edge weight in the clustering model, as confirmed in Figure 4.13. Nevertheless, mail volume is the second criteria used to determine edge weight. In Figure 4.13, control points in zone 3 are spread over the top and the bottom of the facility. This noncontiguous arrangement resulted from the fact that there is a large volume of mail being transported between points in that zone. Figures 4.13, 4.14 and 4.15 present the clustering results for three, five and seven zones, respectively.

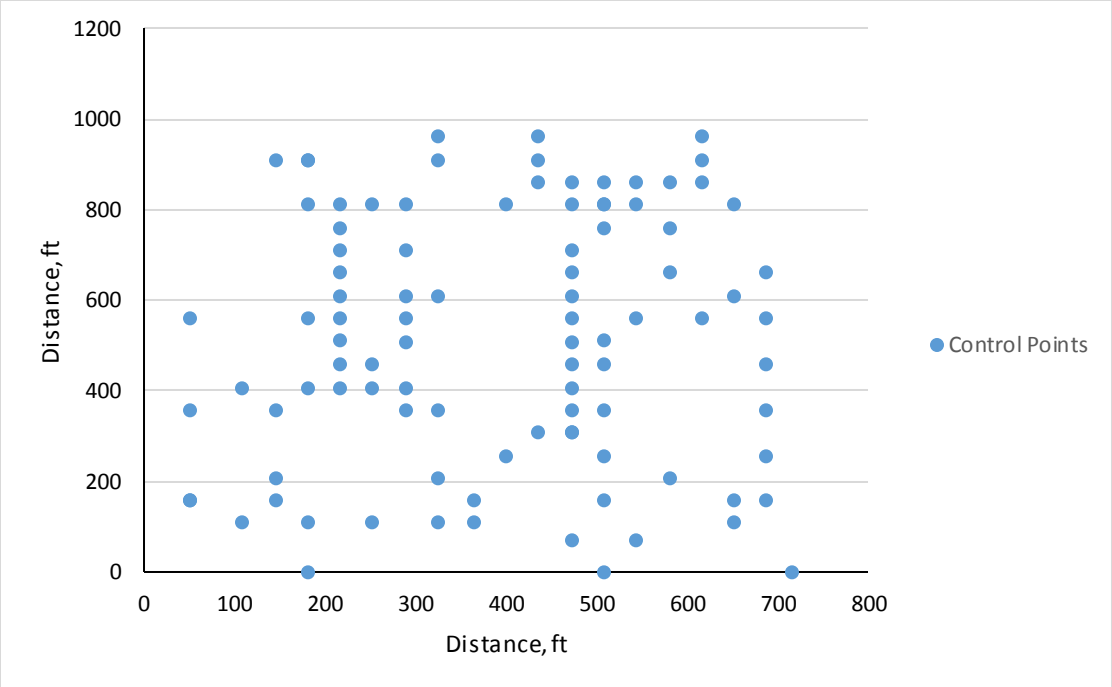


Figure 4.12. Facility layout with control points

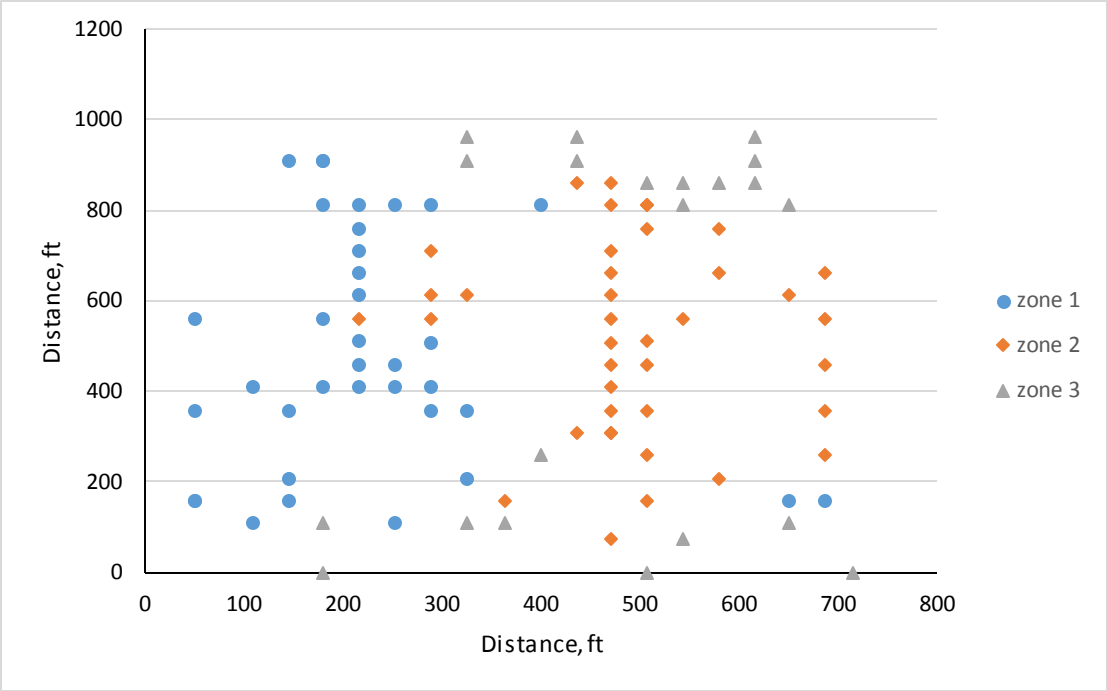


Figure 4.13. Facility divided into three clusters

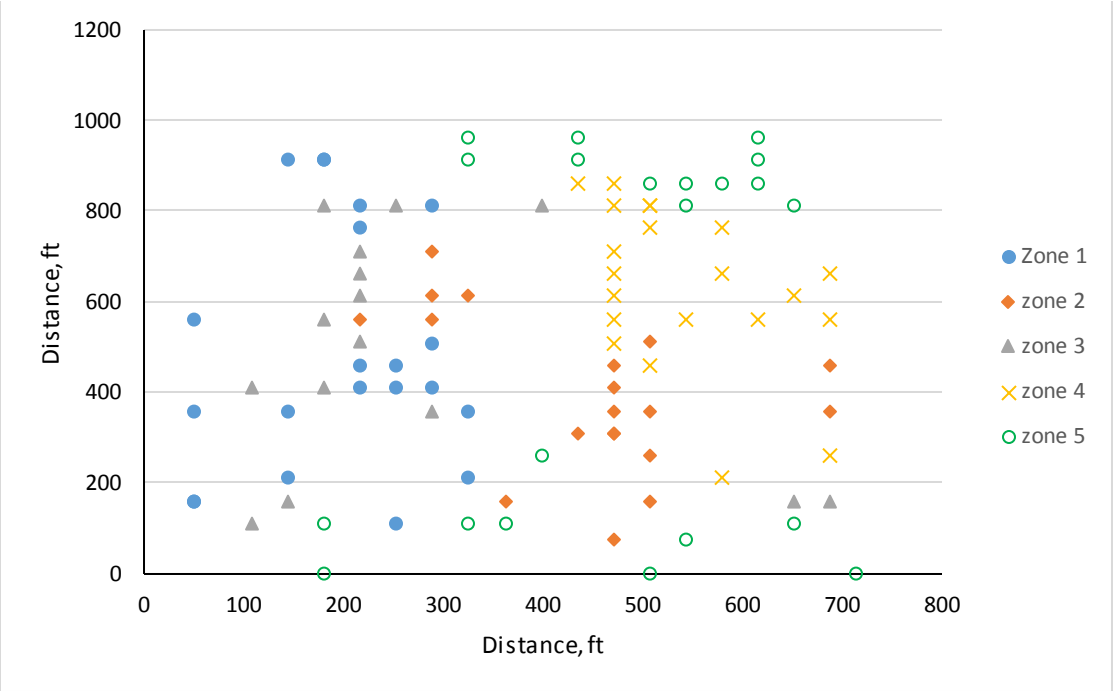


Figure 4.14. Solution for five clusters

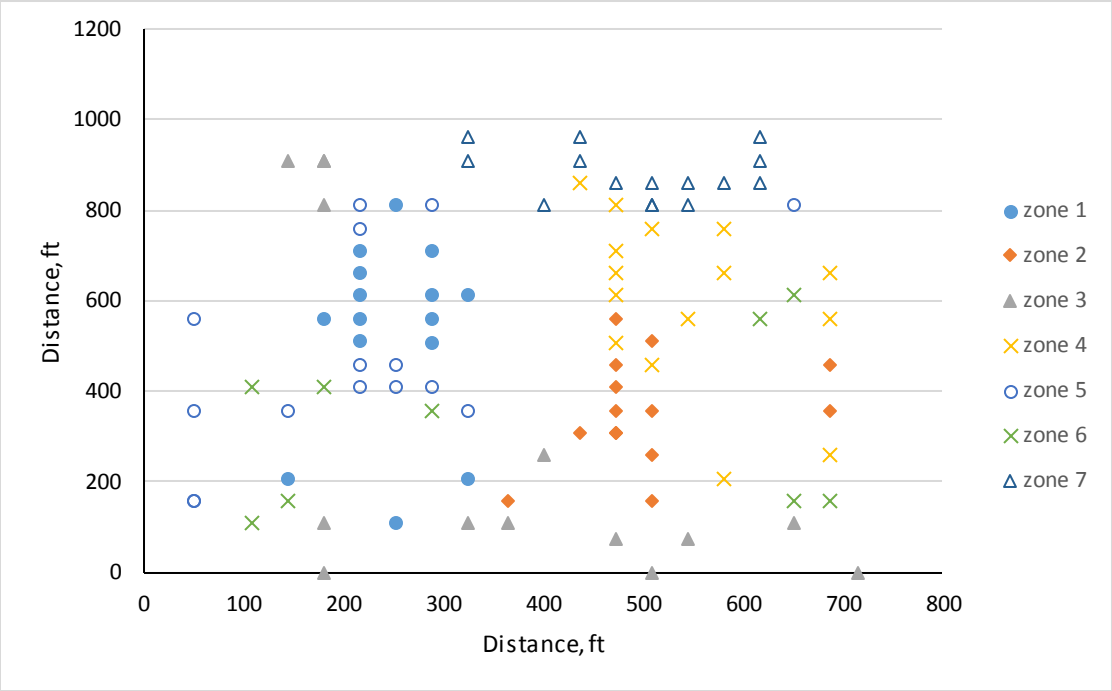


Figure 4.15. Solution for seven clusters

Comparing Figures 4.13 and 4.14, the algorithm creates five clusters by dividing zones 1 and 2 in Figure 4.13 into two subzones, respectively. When seven clusters are desired, zone 3 in Figure 4.14 is split into two smaller zones, as shown in Figure 4.15. Moreover, the control points in zones 1 - 4 in Figure 4.14 are now divided into five zones in Figure 4.15.

Given the clustering results for 3, 5 and 7 zones, we now present our computations for model (2). We start with our experience with CPLEX.

4.6.3 MIP model

Initially, we implemented the exact model presented in Section 4.2 and used CPLEX to try to solve five instances corresponding to five different shifts during the week. The first three shifts represent a typical weekday while the fourth and fifth instances are the first two busiest shifts in the sample week. In all cases, the control points were divided into 3 zones, the travel speed was set to 5 feet per second, the length of pickup time windows was fixed at 5 min, the weight of travel hours was set to 0.01 [which guarantees that the total travel time term in (3a) is less than 1], and the computations were terminated at 5 hours. The characteristics of each instance and the computational results

are given Table 4.1. The first column identifies the shift star and end times, and the day of the week. The next three columns give the number of segments/requests and model size. At termination, the optimality gap reported by CPLEX is given in column 6. The next column reports the number of PIVs needed in the best solutions returned. Given a constant velocity, the total travel time during which the PIVs move mail from pickup points to dropoff points is constant. We call this value the "travel time within segments" and distinguish it from the total "travel time between segments" where empty PIVs travel from the dropoff point of one request to the pickup point of the next request.

The results in Table 4.1 indicate that there are large optimality gaps in the final solutions after 5 hours of computations. Further analysis will show that these values are far from optimal.

Table 4.1. Exact model (2) size and results for 3 zones

Instance		CPLEX statistics						
Timespan	# of requests	# of variable	# of constraints	CPU time (sec)	Optimality gap	# of PIVs	Travel time within segments (hours)	Travel time between segments (hours)
0 am – 8:30 am Wed.	819	186,049	186,880	18000	46%	17	16.98	14.11
8 am – 4:30 pm Wed.	1331	509,075	509,529	18000	60%	18	38.54	19.25
4 pm – 12:59 pm Wed.	849	191,451	191,553	18000	42%	14	16.98	15.58
4 pm – 12:59 pm Sat.	1476	561,257	562,034	18000	47%	21	34.38	19.28
4 pm – 12:59 pm Sun.	862	189,028	189,227	18000	20%	15	20.49	11.23

4.6.4 Rolling horizon heuristic I

Before testing our algorithm, we need to determine an appropriate subproblem size L for heuristic I. In Section 5.1 we developed the heuristic and discussed L with respect to the *worst case scenario*. Given PIV velocity and facility size, the theoretical lower bound on L that avoids potential *worst case scenarios* is 26.4 min. In this section, we test our algorithm with L equal to 15 min, 30 min and 45 min, respectively. Table 4.2 highlights the results when the facility is divided into 3 zones, pickup time windows are 3 (not 5) min and the optimality gap for CPLEX is 5%. If there is only one zone ($K = 1$), then the larger the subproblem size the better the results are likely to be when each subproblem

can be solved to optimality. However, due to *Proposition* 2, this was not our experience for $K \geq 3$, even for values of L less than the lower bound of 26.4 min. For our instances, larger subproblem sizes often led to worse heuristic solutions.

In this phase of the experiments, we have made the original problem somewhat easier so that the subproblems can be solved to optimality in most instances in order to allow us to choose the "best" L. Specifically, we use 3-min pickup time windows instead of 5 min. Also, we set the maximum runtime for the first subproblem to two hours and then one hour for the remaining. The rationale is that the first subproblem has no free PIVs to use so we have to solve a full VRPTW. Empirically we observed that subsequent subproblems are easier to solve since an increasing number of PIVs are available at fixed locations and at no "cost." After the first subproblem, less effort is required for branch and bound. Testing showed that solving the first subproblem to optimality produces better overall results.

Table 4.2. Results for different values of parameter L for rolling horizon heuristic I for 3 zones

Instance	L = 15 min			L = 30 min			L = 45 min		
	CPU (sec)	# of PIVs	Travel time between segments (hr)	CPU (sec)	# of PIVs	Travel time between segments (hr)	CPU (sec)	# of PIVs	Travel time between segments (hr)
0 am – 8:30 am Wed.	269	17	14.57	418	18	14.59	5511	18	14.57
8 am – 4:30 pm Wed.	11801	19	22.19	16347	16	21.52	16956	18	21.95
4 pm – 12:59 pm Wed.	86	16	16.43	217	16	16.25	86	18	16.03
4 pm – 12:59 pm Sat.	146	26	21.60	218	26	20.82	5328	29	20.44
4 pm – 12:59 pm Sun.	7202	23	13.44	3636	24	12.90	3661	22	12.64

In Table 4.2, if an instance takes no more than one hour in total, then its subproblems have been solved to optimality (within 5%). In light of *Proposition* 1, though, values of L that are too small may lead to poor solutions. Nevertheless, this is not always the case as can be seen in the table. For L = 15, instances 1 and 5 have a smaller fleet size than when L = 30. Since not all subproblems in instance 5 are solved to optimality when L = 30, a larger subproblem size cannot lead to a better solution to the original problem.

However, all subproblems in the first instance are solved to optimality for both $L = 15$ and $L = 30$ but the number of PIVs required is greater when $L = 30$. The reason for this anomaly is a direct consequence of *Proposition* 2, that is, it relates to the cross-zone movement rules and not to suboptimal solutions. Retesting the first instance with just one zone gives 11 PIVs and 14.19 hours of travel time for $L = 15$. For $L = 30$, we similarly get 11 PIVs but travel time is reduced by 0.01 hours, a marginal improvement.

When L is relatively large, the subproblems are difficult to solve optimally so the solutions may be poor for the allotted runtime. For the first and the fourth instances, the subproblems can be solved optimally when $L = 30$ but not when $L = 45$. The third instance is a special case because subproblems are also solved to optimality when $L = 45$ but the solution is worse (18 PIVs vs. 16 PIVs). The reason again is due to cross-zone movement under the conditions in *Proposition* 2. Retesting with just 1 zone gave similar results. As the best compromise, we chose L to be 30 min, which is just above the theoretical lower bound for avoiding the *worst case scenario*.

To gauge the performance of our heuristics, we started by solving the model for the same set of instances and input parameters as those in Section 6.3. We then analyzed the solution for the third shift on Saturday (4 pm to 12:59 pm.), which had the most requests and hence was the most difficult. Again, the facility was divided into 3 zones, the traveling speed was set to 5 feet per second, and the length of the pickup time windows was fixed at 5 min. A limit of two hours was placed on the runtime for the first subproblem while all subsequent subproblems had a maximum runtime of 30 min. The results are presented in Table 4.3. In the last column, PIV utilization is equal to 100% × (total travel time) [i.e., sum of the fourth and fifth columns] divided by [(number of PIVs used) × (time horizon of each subproblem)] (i.e., 8 or 8.5 hours). Compared with the results in Table 4.1, the heuristic requires a greater number of hours to service all requests for all instances but uses no more than the number of PIVs indicated in the CPLEX solution except for the fourth instance which is analyzed in Table 4.4.

Table 4.3. Results for rolling horizon heuristic I for 3 zones

Instance	CPU (sec)	# of PIVs	Travel time within segments (hours)	Travel time between segments (hours)	PIV utilization
0 am – 8:30 am Wed.	13250	14	16.98	14.60	27%
8 am – 4:30 pm Wed.	17442	18	38.54	20.67	39%
4 pm – 12:59 pm Wed.	3959	12	16.98	16.07	34%
4 pm – 12:59 pm Sat.	13180	22	34.38	20.66	31%
4 pm – 12:59 pm Sun.	4084	15	20.49	12.78	28%

The first column in Table 4.4 lists the time increment of each subproblem followed by the number of requests. The second column presents the number of segments in each 30-min subproblem. Note the half hour overlap on the Wednesday instances. The third column indicates the number of segments in each subproblem. The fourth column gives the number of additional PIVs needed in each subproblem, while the fifth column indicates the number of existing PIVs used in each subproblem. Because the value of □□ << 1 in (3a), the algorithm tries to minimize the number of additional PIVs used in each subproblem before it tries to minimize the total travel time. The sixth column gives the cumulative travel hours within and between segments through the end time of each subproblem.

The heuristic took 13,180 sec (221 min) in total to converge. As we can see from Table 4.4, the first subproblem required 12 PIVs, as does the second and third since none were added. In the fourth subproblem (4:45 pm – 5:15 pm), two PIVs are added. For the subproblems that span 5:00 pm to 7:45 pm, either 13 or 14 PIVs are used and none are added. In the next subproblem corresponding to the interval [7:45 pm, 8:15 pm], 1 PIV is added; however, between 10:30 pm and 11:15 pm, 6 new PIVs are required. The reason for this sharp increase can be attributed to the spike in demand starting at 10:30 pm. The average number of segments per half hour jumps from 82 before 10:30 pm to 126 during the interval [10:30 pm, 11:15 pm]. In all, 22 PIVs and 55.38 hours of travel time are required to meet all requests.

The last two columns give insight into PIV utilization. For the next to last column, PIV utilization is equal to the total travel time during the subproblem divided by the

number of PIVs used times the timespan of each subproblem (i.e., 30 min) as a percent. The first subproblem has a high utilization (71%) because the number of PIVs used is being minimized for the first time. In the second subproblem, no additional PIVs are required but the utilization drops to 58% because there is less work to do. The last column reports PIV utilization through the end of current subproblem. As defined above, it is equal to the total travel time so far divided by the number of PIVs required through the end of current subproblem multiplied by the elapsed time from the start of the shift through the end of the current subproblem. This value decreases as the time horizon increases because each additional PIV is considered to be "idle" prior to its introduction.

Figures 4.15 – 4.17 depict the cumulative PIV travel hours in zones 1, 2 and 3, respectively. In Figure 4.16 we see that 5 PIVs in total are used; the corresponding graphs represent their cumulative travel hours up to the time indicated on the horizontal axis. If the cumulative travel hours in a particular period is the same as that in the previous period, then the PIV has not been assigned any requests in the later period. Four PIVs are introduced at the beginning of the shift (i.e., between 4:00 pm and 4:30 pm) and only one is added subsequently in the period from 7:00 pm to 7:30 pm. However, Figure 4.17 indicates that four new PIVs are needed in the period from 10:30 pm to 11:00 pm in zone 2. Their introduction towards the end of the shift is due to a sudden increase of demand. Similarly, for zone 3 in Figure 4.18; that is, 1 PIV is added in the period starting at 10:30 pm and one in the period starting at 11:00 pm.

Table 4.4. Subproblem solutions of the fourth instance in Table 3.3 for heuristic I for 3 zones

Subproblem timespan	# of requests	Optimality gap	# of new PIVs required	# of existing PIVs used	Total travel time (hours)	PIV utilization for each subproblem	PIV utilization through end of subproblem
4:00 pm – 4:30 pm	102	25%	12	0	4.27	71%	71%
4:15 pm – 4:45 pm	82	5%	0	12	6.15	58%	68%
4:30 pm – 5:00 pm	84	5%	0	12	7.75	58%	65%
4:45 pm – 5:15 pm	81	5%	2	12	9.72	51%	56%
5:00 pm – 5:30 pm	82	4%	0	14	11.61	55%	55%
5:15 pm – 5:45 pm	84	4%	0	14	13.41	53%	55%
5:30 pm – 6:00 pm	74	5%	0	14	14.73	45%	53%
5:45 pm – 6:15 pm	67	4%	0	14	15.95	36%	51%
6:00 pm – 6:30 pm	82	5%	0	13	17.67	45%	50%
6:15 pm – 6:45 pm	65	1%	0	14	18.46	36%	48%
6:30 pm – 7:00 pm	56	3%	0	14	20.03	34%	48%
6:45 pm – 7:15 pm	69	5%	0	13	21.33	44%	47%
7:00 pm – 7:30 pm	68	3%	0	13	23.35	51%	48%
7:15 pm – 7:45 pm	85	4%	0	14	25.31	57%	48%
7:30 pm – 8:00 pm	105	3%	0	14	27.05	53%	48%
7:45 pm – 8:15 pm	112	98%	1	14	28.95	48%	45%
8:00 pm – 8:30 pm	92	5%	0	15	30.31	43%	45%
8:15 pm – 8:45 pm	61	5%	0	13	30.93	30%	43%
8:30 pm – 9:00 pm	55	3%	0	12	31.86	26%	42%
8:45 pm – 9:15 pm	62	5%	0	10	33.01	42%	42%
9:00 pm – 9:30 pm	90	5%	0	14	34.68	40%	42%
9:15 pm – 9:45 pm	106	5%	0	14	36.46	49%	42%
9:30 pm – 10:00 pm	105	4%	0	13	38.31	56%	43%
9:45 pm – 10:15 pm	96	4%	0	15	39.74	44%	42%
10:00 pm – 10:30 pm	100	3%	0	15	42.13	51%	43%
10:15 pm – 10:45 pm	114	4%	0	15	44.06	58%	44%
10:30 pm – 11:00 pm	128	75%	3	15	46.46	48%	37%
10:45 pm – 11:15 pm	138	66%	3	18	49.03	47%	32%
11:00 pm – 11:30 pm	123	5%	0	21	51.10	44%	32%
11:15 pm – 11:45 pm	127	5%	1	21	53.34	39%	31%
11:30 pm – 12:00 pm	130	3%	0	22	55.38	39%	31%

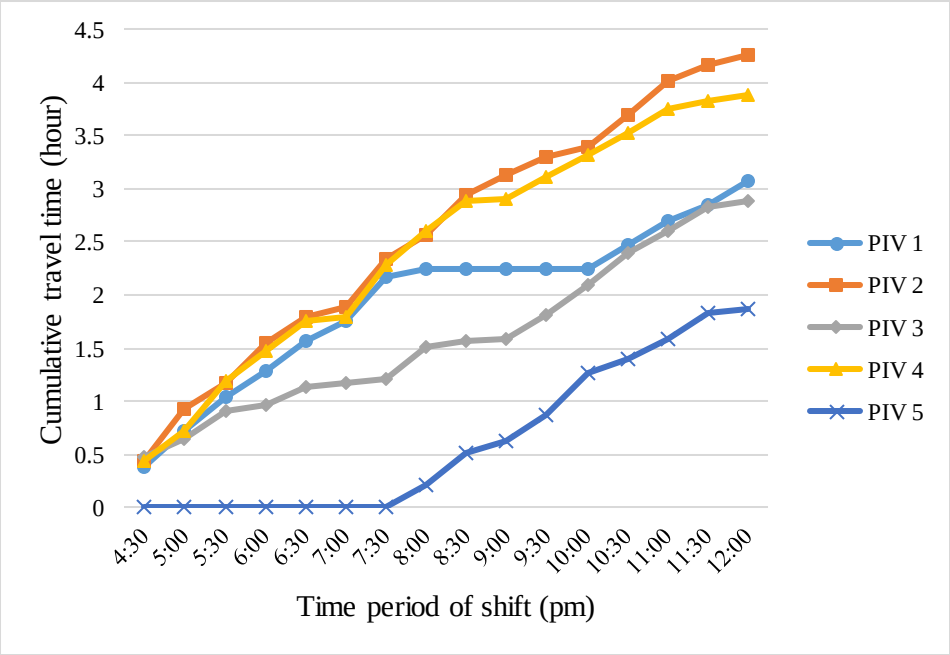


Figure 4.16. Travel time for PIVs in zone 1 for Saturday PM shift

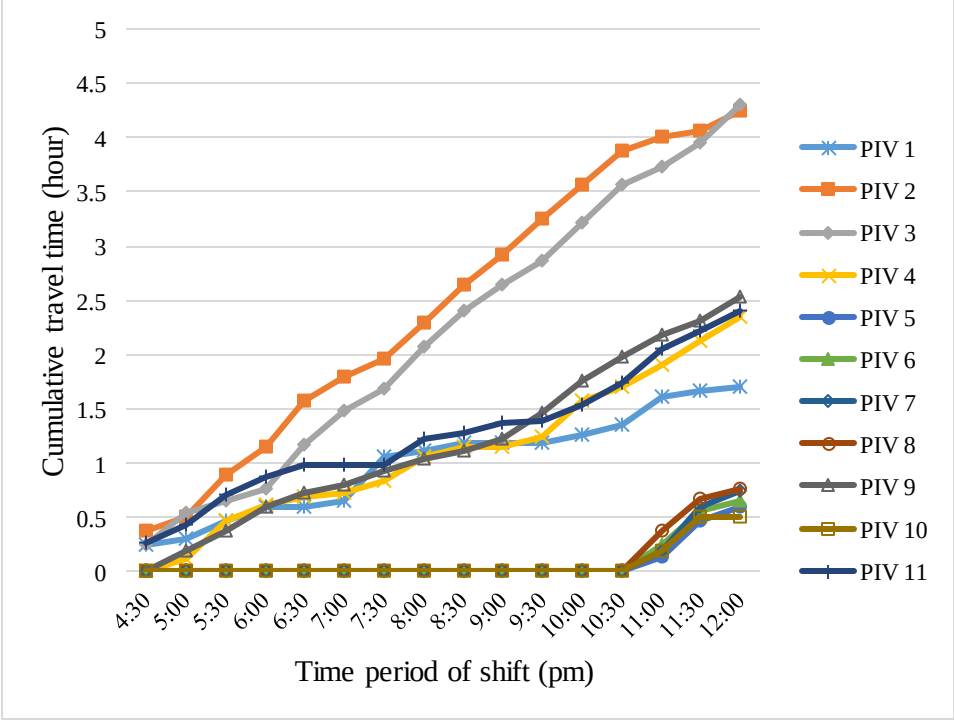


Figure 4.17. Travel time for PIVs in zone 2 for Saturday PM shift

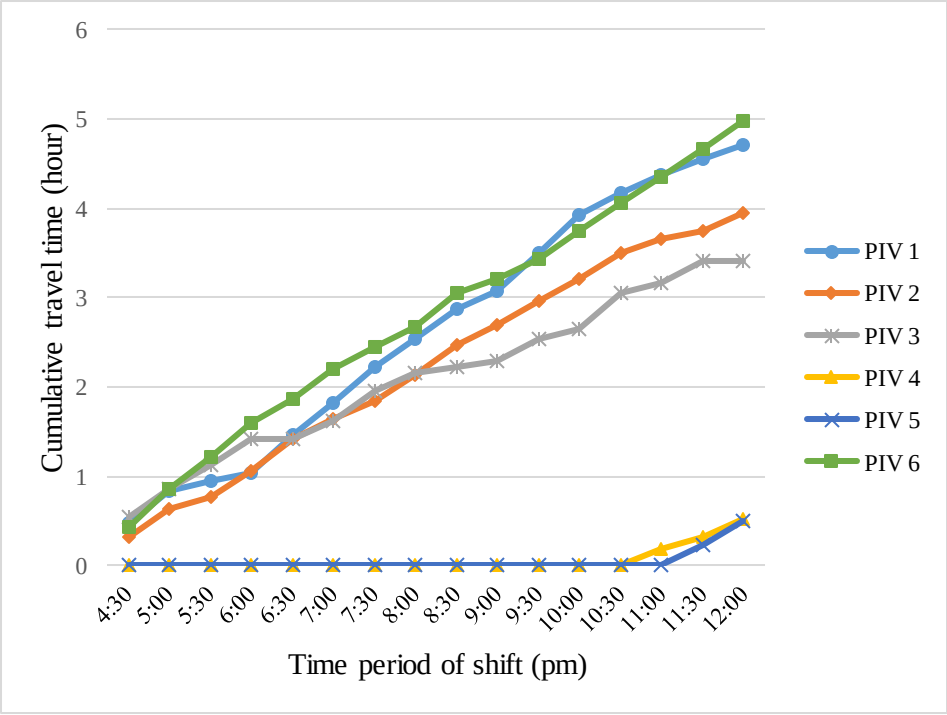


Figure 4.18. Travel time for PIVs in zone 3 for Saturday PM shift

4.6.5 Rolling horizon heuristic II

Our second heuristic dynamically adjusts the timespan of each subproblem in an effort to keep its size constant. Like rolling horizon heuristic I, we also need to determine an appropriate subproblem size. According to the input data for the P&DC application, there are an average of 60 requests per ½-hour period. Recalling that N is the number of segments per subproblem, testing was done for N equal to 30, 60 and 90 for each of the five instances. We still use 3 zones, 5 feet per second PIV speed, and 5 min pickup time windows. The CPLEX parameters, such as maximum runtime and optimality gap, are the same as those used in Section 6.4. A limit of two hours was placed on the runtime of the first subproblem and 30 min for all subsequent subproblems.

The results are shown in Table 4.5. In the second and third instances, the number of PIVs needed is non-increasing as subproblem size grows, as might be expected. The results for the first, fourth and fifth instances are not as consistent but strongly suggest that 60-segment subproblems provide the best solutions for our hierarchical objective function. When $N = 30$, the third and fifth instances are solved within 3600 sec which

indicates that optimal solutions to all subproblems are obtained. However, they are worse than the solutions obtained with $N = 60$ because solving for only 30 segments at a time sharply limits feasible transitions due to the occurrence of the *worst case scenario* in some subproblems. From Table 4.5, we conclude that $N = 60$ is a good compromise for heuristic II, and is used in all of the remaining experiments.

Table 4.5. Results for different values of parameter N for rolling horizon heuristic II for 3 zones

Instance	$N = 30$			$N = 60$			$N = 90$		
	CPU (sec)	# of PIVs	Travel time between segments (hours)	CPU (sec)	# of PIVs	Travel time between segments (hours)	CPU (sec)	# of PIVs	Travel time between segments (hours)
0 am – 8:30 am Wed.	9450	16	14.63	11612	14	14.60	11251	14	14.69
8 am – 4:30 pm Wed.	4111	18	22.15	16141	18	21.93	15998	16	21.21
4 pm – 12:59 pm Wed.	248	14	16.38	5503	13	15.95	11952	12	16.02
4 pm – 12:59 pm Sat.	4472	23	21.59	16447	20	21.03	21616	23	20.43
4 pm – 12:59 pm Sun.	1828	15	12.47	1939	14	12.43	9458	16	12.17

Intuitively, heuristic II should be more stable than heuristic I and hence produce better results. This follows because sharp increases in demand in a single period can make the corresponding heuristic I subproblem very difficult to solve leading to a non-optimal solution with an excessive number of PIVs. This situation does not arise when the subproblems are constant in size. Table 4.6 gives the subproblem solutions obtained from heuristic II for the fourth test instance. In all, 48 subproblems were solved. Compared to the results in Table 4.4 for the same instance, we see that heuristic II is better able to handle surges in demand. For example, compare the results for the interval 10:30 pm to 11:15 pm in both cases.

After we solve the initial subproblem, we fix the routes associated with the first 30 segments and then create the second subproblem by extending the timespan to contain the next 30 segments. In all, heuristic II provided a solution with 20 PIVs and total travel time of 55.67 hours. Comparing the results in Table 4.6 with those in Table 4.4, we see a

decrease of 2 PIVs, although the total travel time increases by 0.29 hours, or 0.5%. The reason for the PIV decrease is that heuristic II only adds 3 new PIVs over the interval from 10:30 pm to 11:15 pm when demand surges while heuristic I adds 6 in the same period. With the decrease in fleet size with get an increase in overall PIV utilization from 31% to 35%.

The computational results in Tables 4.1, 4.3 and 4.5 for the exact model, rolling horizon heuristic I, and rolling horizon heuristic II, respectively, are summarized in Table 4.7. As mentioned, the subproblem size for heuristic I is 30 min and the subproblem size for heuristic II is 60 segments. The pickup time windows are all 5 min and PIV travel speed remains at 5 feet per second. Control points in the facility are classified into the three zones. For the exact model, the runtime limit was a maximum of five hours.

Table 4.6. Subproblem solutions for the fourth instance in Table 4.3 for heuristic II for 3 zones

Subproblem timespan	Optimality gap	# of new PIVs required	# of existing PIVs used	Total travel time (hours)	PIV utilization in each subproblem	PIV utilization up to the end of subproblem
4:00 pm – 4:14 pm	21%	12	0	2.52	90%	90%
4:07 pm – 4:27 pm	25%	0	12	3.83	64%	71%
4:14 pm – 4:38 pm	4%	0	12	5.12	54%	67%
4:27 pm – 4:47 pm	5%	0	12	6.27	61%	67%
4:38 pm – 4:59 pm	5%	0	12	7.86	65%	67%
4:47 pm – 5:10 pm	5%	1	13	9.09	52%	60%
4:59 pm – 5:19 pm	5%	1	11	10.29	61%	56%
5:10 pm – 5:31 pm	3%	0	14	11.73	54%	55%
5:19 pm – 5:43 pm	4%	0	13	13.14	55%	55%
5:31 pm – 5:53 pm	5%	0	13	14.35	55%	54%
5:43 pm – 6:09 pm	3%	0	13	15.37	40%	51%
5:53 pm – 6:20 pm	5%	0	13	16.70	40%	51%
6:09 pm – 6:30 pm	5%	0	13	17.57	48%	50%
6:20 pm – 6:49 pm	3%	0	12	18.90	38%	48%
6:30 pm – 7:01 pm	3%	0	12	20.07	40%	48%
6:49 pm – 7:15 pm	5%	0	13	21.46	45%	47%
7:01 pm – 7:28 pm	3%	0	14	22.93	45%	47%
7:15 pm – 7:38 pm	2%	0	14	24.03	48%	47%
7:28 pm – 7:47 pm	5%	0	14	25.32	54%	48%
7:38 pm – 7:55 pm	5%	0	13	26.01	54%	47%
7:47 pm – 8:02 pm	99%	1	13	27.31	57%	45%
7:55 pm – 8:11 pm	99%	1	13	28.54	68%	43%
8:02 pm – 8:20 pm	5%	0	13	29.48	56%	43%
8:11 pm – 8:34 pm	5%	0	13	30.52	40%	42%
8:20 pm – 8:51 pm	1%	0	12	31.35	30%	40%
8:34 pm – 9:05 pm	0%	0	12	32.20	27%	40%
8:51 pm – 9:18 pm	4%	0	13	33.31	33%	39%
9:05 pm – 9:23 pm	5%	0	14	34.21	48%	40%
9:18 pm – 9:33 pm	5%	0	14	35.35	58%	40%
9:23 pm – 9:43 pm	5%	0	14	36.34	46%	40%
9:33 pm – 9:52 pm	5%	0	15	37.48	45%	40%
9:43 pm – 10:00 pm	3%	0	13	38.46	58%	40%
9:52 pm – 10:09 pm	5%	0	16	39.62	47%	40%
10:00 pm – 10:21 pm	1%	0	16	40.46	36%	40%
10:09 pm – 10:27 pm	5%	0	15	41.81	49%	41%
10:21 pm – 10:35 pm	5%	0	16	43.03	69%	41%
10:27 pm – 10:44 pm	5%	0	16	44.16	52%	41%
10:35 pm – 10:47 pm	2%	0	16	44.55	48%	41%
10:44 pm – 10:54 pm	100%	3	15	45.91	59%	35%
10:47 pm – 11:02 pm	2%	0	16	46.98	61%	35%
10:54 pm – 11:08 pm	5%	0	17	48.01	53%	35%
11:02 pm – 11:17 pm	99%	1	17	49.60	58%	34%
11:08 pm – 11:26 pm	5%	0	20	50.74	45%	34%
11:17 pm – 11:31 pm	5%	0	20	51.73	46%	34%
11:26 pm – 11:39 pm	1%	0	20	52.86	49%	35%
11:31 pm – 11:45 pm	1%	0	19	53.85	48%	35%
11:39 pm – 11:51 pm	5%	0	16	54.70	57%	35%
11:45 pm – 11:59 pm	5%	0	15	55.67	52%	35%

Table 4.7. Comparative summary of computation results for 3 zones

Instances	Exact model				Rolling horizon heuristic I				Rolling horizon heuristic II			
	CPU hour	# of PIVs	Travel time within requests (hours)	PIV util.	CPU hour	# of PIVs	Travel time within requests (hours)	PIV util.	CPU hour	# of PIVs	Travel time within requests (hours)	PIV util.
0 am – 8:30 am Wed.	5	17	14.11	22%	3.7	14	14.60	27%	3.2	14	14.60	27%
8 am– 4:30 pm Wed.	5	18	19.25	38%	4.8	18	20.67	39%	4.5	18	21.93	40%
4 pm – 12:59 pm Wed.	5	14	15.58	29%	1.1	12	16.07	34%	1.5	13	15.95	32%
4 pm – 12:59 pm Sat.	5	21	19.28	32%	3.7	22	20.66	31%	4.6	20	21.03	35%
4 pm – 12:59 pm Sun.	5	15	11.23	26%	1.1	15	12.78	28%	0.5	14	12.43	29%

Recall that in Table 4.1, the average optimal gap is 43% which indicates that there is potentially room for significant improvement. Rolling horizon heuristic I provides better solutions than the exact model in most instances expect for the fourth, while rolling horizon heuristic II finds better solutions for all instances except for the second where the results are the same. Also, PIV utilization rates for heuristic II solutions are greater than those for the exact model solutions in all instances and runtimes are up to an order of magnitude less.

4.6.6 Sensitivity analysis

In the final set of experiments we explored the sensitivity of rolling horizon heuristic II to changes in the following parameter values: number of zones, traveling speed, and length of the pickup time windows. All the computations were conducted on the fourth instance, which is the busiest shift in the sample week. We used the same CPLEX parameters as indicated in Section 6.5.

Table 4.8 highlights the impact of changing the number of zones. The asterisk indicates the value used in the baseline testing. From the table, we first see a significant savings with one zone, as expected, and that the number of PIVs required increases with the number of zones. This is intuitive because the cross-zone movement rules limit the feasible transition for the PIVs. For the same reason, travel time between segments

increases when the number of zones increases. To a large extent, facility design is a managerial decision. The results presented in Table 4.8 provide insights into the costs that come with tighter controls.

The results in Table 4.9 highlight the impact of changing vehicle speed. As expected, as speed increases, both the travel time within segments and the travel time between segments decrease. Also, greater speed means fewer PIVs and hence lower costs; however, more safety measures may be required.

The size of the pickup time windows is mainly an operations decision. Setting it too wide may lead to a solution with fewer PIVs than are really needed as well as large delays in the mail flow. In contrast, if the time window is too small, a greater number of PIVs than is really needed will be the consequence. Choosing a reasonable size time window is critical to meeting P&DC service standards at minimum cost. The results in Table 4.10 demonstrate the available tradeoffs.

Table 4.8. Parametric analysis for number of zones

# of zones	CPU (sec)	# of PIVs	Travel time within segments (hour)	Travel time between segments (hour)	PIV utilization rate
1	11664	11	34.38	18.56	60%
3*	16447	20	34.38	21.03	35%
5	12471	27	34.38	21.66	26%
7	2838	29	34.38	24.77	25%

Table 4.9. Parametric analysis for PIV speed

Speed (feet/sec)	CPU (sec)	# of PIVs	Travel time within segments (hour)	Travel time between segments (hour)	PIV utilization rate
3	8157	30	57.02	34.89	38%
4	15111	27	43.03	26.30	32%
5*	16447	20	34.38	21.03	35%
6	14466	17	28.86	17.48	34%

Table 4.10. Parametric analysis for pickup time windows

Pickup time window (min)	CPU (sec)	# of PIVs	Travel time within segments (hour)	Travel time between segments (hour)	PIV utilization rate
3	1990	26	34.38	21.37	27%
5*	16447	20	34.38	21.03	35%
10	18633	19	34.38	20.38	36%
12	21234	18	34.38	20.29	38%

4.7. Summary and Conclusions

This paper addresses a VRPTW in which thousands of mail transfer requests within a P&DC must be satisfied each day. Primarily for supervisory reasons, it was first necessary cluster the pickup and dropoff points into zones before determining the optimal feet size and routes. Because the number and composition of the zones is not input to the problem, we took a parametric approach and examined the relationship between the number of zones and PIV requirements.

For a given facility configuration, we developed a mixed-integer programming model with the objective of minimizing a weighted sum of the number of PIVs required plus the total travel time over a shift to meet all requests. A unique feature of the model is the need to limit PIV cross-zone movement which mades the problem exceptionally difficult to solve with CPLEX. This led to the development of a column generation algorithm that similarly failed to converge and so was abandoned in favor of two optimization-based, rolling horizon heuristics. Extensive testing with data provided by the Chicago P&DC showed that the second heuristic in which a fixed number of requests are assigned to each subproblem yielded the best results. High quality solutions were obtained for the most difficult instances in approximately 274 min.

In our final experiments, we studied the relationship between solution quality and variations in the number of zones, PIV speed, and pickup time window length. The results showed travel time between segments increased by 33% as the number of zones went from 1 to 7. When the PIV speed doubled from 3 ft/sec to 6 ft/sec, the number of PIVs required decreased from 30 to 17 while the total travel time decreases by 49.5%.

Larger pickup time windows made instances harder to solve but gave route planners more flexibility. When pickup time windows increased from 3 min to 12 min, the number of PIVs required decreased from 26 to 18 and travel time between segments decreased by 5%.

In addition to total cost, our results provide another factor for shop floor supervisors to consider when deciding on values for these parameters. In particular, PIV utilization is an important metric that indicates the efficiency of mail transport operations. There is always a tradeoff between utilization and service levels that underlies the long-term decision on the number of PIVs to actually acquire. Because demand changes from shift to shift over the week, it may be cost-efficient to configure a facility to meet, say, 80% of the demand on the busiest shift rather than 100%.

Looking at the results, we see that there are several areas where improvement is possible. The first would be to develop a more specialized approach to solving the subproblems. One such possibility would be to implement a branch-and-cut algorithm rather than rely on a commercial solver. A more efficient approach would allow for larger, and hence fewer, subproblems to be solved. The second area for improvement comes from the observation that the number of PIVs required for a shift is mainly determined by the number of PIVs needed in the busiest period of that shift. Instead of starting the rolling horizon algorithm from the beginning of a shift, it may be more effective to start it from the busiest period and extend the subproblems both forwards and backwards.

Appendix A: Column Generation

An efficient way to find a tight lower bound on a mixed-integer linear program with a minimization objective is to use column generation (CG) rather than solve the LP relaxation directly. Once the bound is found, branch and bound can be used to find an integer solution. For VRPs, the first step is to create a restricted linear master problem from the demand constraints. In the resultant formulation, each column represents a feasible route for a vehicle. New routes are generated as needed by solving a series of subproblems created from the remaining constraints. Each new route is placed in the master problem as a column. By design, only columns that have the potential to reduce the master problem objective function are included. These are identified by minimizing a generic representation of the reduced cost in the subproblems, one for each zone in our case.

To start the computations, it is necessary to initialize the restricted master problem with a set of columns that correspond to a feasible solution. For every request j with pickup point in zone k, we define a route r (i.e., a column in the master problem) that starts from depot k, connects to segment j and returns to depot k.

The objective function of the subproblem is the reduced cost of the new variable with respect to the current dual variables for the master problem. For each zone k, we have a separate subproblem that aims to find a route starting from and returning to depot k with a negative reduce cost.

We first construct the master problem and then provide the formulation of the subproblem.

Notation

N set of segments (nodes)

E set of valid connections (edges)

$R(k)$ set of feasible routes (columns) for zone k

R set of all feasible routes (columns), $R = \bigcup_{k \in K} R(k)$

X_i^{kr} (parameter) 1 if segment i is visited in the route r that starts from depot k [depends on the subproblem solution associated with route/column $r \in R(k)$], 0 otherwise;

$$X_i^{kr} = \sum_{j:<i,j>\in E} x_{ij}^{kr}$$

a_i the beginning of time window for segment i

b_i the end of time window for segment i

σ_i the amount of time to travel from the pickup point to the drop off point of segment i

$$\sigma_{0_k} = 0, \forall k \in K$$

τ_{ij} time to travel from the drop off point of segment i to the pickup point of segment j

ε arbitrarily small positive constant

c_{ij} 1 if i is depot; otherwise $c_{ij} = \varepsilon\tau_{ij}$

c^{kr} cost of a feasible schedule that associates with route/column $r \in R(k)$;

$$c^{kr} = \sum_{<i,j>\in E} c_{ij} x_{ij}^{kr} \quad \forall\, k \in K, r \in R(k)$$

Decision variables

λ^{kr} continuous variable in [0, 1]. It represents the fractional of the route $r \in R(k)$ that is selected.

Master Problem (*MP*)

$$\Phi_{IP} = \min \sum_{k\in K} \sum_{r\in R(k)} c^{kr}\lambda^{kr} \tag{5.1}$$

Subject to

$$\sum_{k\in K} \sum_{r\in R(k)} X_i^{kr}\lambda^{kr} = 1, \ \forall\, i \in N \qquad \pi_i \tag{5.2}$$

$$0 \le \lambda^{kr} \le 1, \ \forall\, k \in K, \ r \in R(k) \tag{5.3}$$

Subproblem k (*SP*$_k$)

Reduce cost $\bar{c}^{kr}$ where $r \in R(k)$.

$$\bar{c}^{kr} = c^{kr} - \sum_{l\in K} \sum_{i\in S_l \cup \bar{S}_l} \pi_i X_i^{kr}$$

When $i \in S_l$ for all $l \in K$, the necessary condition that a PIV from depot k will visit segment i is $l = k$. When $i \in \bar{S}_l$ for all $l \in K$, the necessary condition that a PIV from depot k will visit segment i is either $i \in S_{lk}, l \neq k$ or $i \in S_{kl}, l \neq k$. In the former case, a PIV from depot k services segment i on its way from zone l back to zone k. In the latter case, a PIV from depot k begins to service segment i at a control point in zone k. In all other cases, $X_i^{kr} = 0$. Thus, the reduced cost for a generic column kr in model (5.1) – (5.3) is

$$\bar{c}^{kr} = c^{kr} - \sum_{l \in K} \left\{ \sum_{i \in S_l, l=k} \pi_i X_i^{kr} + \sum_{i \in S_{kl}, l \neq k} \pi_i X_i^{kr} + \sum_{i \in S_{lk}, l \neq k} \pi_i X_i^{kr} \right\} \tag{5.4}$$

$$= \sum_{<i,j> \in E} c_{ij} x_{ij}^{kr} - \sum_{l \in K} \left\{ \sum_{i \in S_l, l=k} \pi_i \sum_{j:<i,j> \in E} x_{ij}^{kr} + \sum_{i \in S_{kl}, l \neq k} \pi_i \sum_{j:<i,j> \in E} x_{ij}^{kr} + \sum_{i \in S_{lk}, l \neq k} \pi_i \sum_{j:<i,j> \in E} x_{ij}^{kr} \right\} \tag{5.5}$$

where the second term in (5.5) represents the transformation of the parameter X_i^{kr} into the subproblem variable x_{ij}^{kr}. Taking into account the rules for cross zone movements and feasible transitions, (5.5) can be written as

$$\bar{c}^{kr} = \sum_{<i,j> \in E} c_{ij} x_{ij}^{kr} - \sum_{i \in S_k} \pi_i \sum_{j \in S_k \cup \bar{S}_k \cup \{0_k\}} x_{ij}^{kr} - \sum_{l \in K \setminus \{k\}} \sum_{i \in S_{lk}} \pi_i \sum_{j \in S_k \cup \bar{S}_k \cup \{0_k\}} x_{ij}^{kr} - \sum_{l \in K \setminus \{k\}} \sum_{i \in S_{kl}} \pi_i \sum_{j \in S_k \cup \bar{S}_k \cup S_{lk} \cup \{0_k\}} x_{ij}^{kr}$$

Now, according to the definition of c_{ij}, $\bar{c}^{kr}$ is equivalent to the following:

$$\bar{c}^{kr} = \sum_{j \in S_k \cup \bar{S}_k} x_{0_k j}^{kr} + \sum_{i \in S_k} \sum_{j \in S_k \cup \bar{S}_k \cup \{0_k\}} (\varepsilon \tau_{ij} - \pi_i) x_{ij}^{kr} + \sum_{l \in K \setminus \{k\}} \sum_{i \in S_{lk}} \sum_{j \in S_k \cup \bar{S}_k \cup \{0_k\}} (\varepsilon \tau_{ij} - \pi_i) x_{ij}^{kr} + \sum_{l \in K \setminus \{k\}} \sum_{i \in S_{kl}} \sum_{j \in S_k \cup \bar{S}_k \cup S_{lk} \cup \{0_k\}} (\varepsilon \tau_{ij} - \pi_i) x_{ij}^{kr}$$

Objective function

$$\min \sum_{k \in K} \left\{ \sum_{j \in S_k \cup \bar{S}_k} x_{0_k j}^{k} + \sum_{i \in S_k} \sum_{j \in S_k \cup \bar{S}_k \cup \{0_k\}} (\varepsilon \tau_{ij} - \pi_i) x_{ij}^{k} + \sum_{l \neq k} \sum_{i \in S_{kl}} \sum_{j \in S_k \cup \bar{S}_k \cup S_{lk} \cup \{0_k\}} (\varepsilon \tau_{ij} - \pi_i) x_{ij}^{k} + \sum_{l \neq k} \sum_{i \in S_{lk}} \sum_{j \in S_k \cup \bar{S}_k \cup \{0_k\}} (\varepsilon \tau_{ij} - \pi_i) x_{ij}^{k} \right\} \tag{5.6}$$

Force to choose at most one route from depot k

$$\sum_{j \in S_k \cup \bar{S}_k} x_{0_k j} \leq 1 \tag{5.7}$$

Forbid routes from other depots

$$\sum_{k' \in K \setminus \{k\}} \sum_{j \in S_{k'} \cup \bar{S}_{k'}} x_{0_{k'} j} = 0 \tag{5.8}$$

Flow balance for segment j that has pickup and drop off points in zone k

$$\sum_{i \in S_k \cup \bar{S}_k \cup \bar{\bar{S}}_k \cup \{0_k\}} x_{ij}^k - \sum_{h \in S_k \cup \bar{S}_k \cup \{0_k\}} x_{jh}^k = 0, \qquad \forall k \in K, j \in S_k \tag{5.9}$$

Flow balance for segment $j \in S_{kl}$, when j is transferred by a PIV from zone k

$$\sum_{i \in S_k \cup \bar{S}_k \cup \bar{\bar{S}}_k \cup \{0_k\}} x_{ij}^k - \sum_{h \in S_k \cup \bar{S}_k \cup S_{lk} \cup \{0_k\}} x_{jh}^k = 0, \qquad \forall k \in K, j \in S_{kl} \tag{5.10}$$

Flow balance for segment $j \in S_{kl}$, when j is transferred by a PIV from zone l

$$\sum_{i \in S_{kl}} x_{ij}^k - \sum_{h \in S_k \cup \bar{S}_k \cup \{0_k\}} x_{jh}^k = 0, \qquad \forall k \in K, j \in S_{lk} \tag{5.11}$$

Elimination of sub-tours: i has pickup and drop off points in zone k

$$t_j \geq t_i + \sigma_i + \tau_{ij} - M_{ij}(1 - x_{ij}^k), \quad \forall k \in K, i \in S_k, j \in S_k \cup \bar{S}_k \tag{5.12}$$

Elimination of sub-tours: $i \in S_{kl}$, when i is transported by a PIV from zone k

$$t_j \geq t_i + \sigma_i + \tau_{ij} - M_{ij}(1 - x_{ij}^k), \quad \forall k \in K, i \in S_{kl}, j \in S_k \cup \bar{S}_k \cup S_{ik} \tag{5.13}$$

Elimination of sub-tours: $i \in S_{kl}$, when i is transported by a PIV from zone l

$$t_j \geq t_i + \sigma_i + \tau_{ij} - M_{ij}(1 - x_{ij}^k), \quad \forall k \in K, i \in S_{lk}, j \in S_k \cup \bar{S}_k \tag{5.14}$$

Time windows for every segment except depots

$$a_i \leq t_i \leq b_i, \ \forall \ i \in S \setminus \{0_k\} \tag{5.15}$$

$$x_{ij}^k \in \{0,1\}, \ \forall < i, j > \in E \tag{5.16}$$

The following pseudocode describes our column generation algorithm. It outlines how to find initial routes, how to add columns, and when to terminate the computations.

Procedure_Column_Generation

Input: S; maximum number of iterations, MAX

Output: number of PIVs needed and their routes for the shift under consideration

Begin

for every segment $j \in S$

{

Let k be the cluster in which the pickup point of segment j is located

Generate a route r that starts from depot k, connects to segment j and returns to the depot.

Add route r to $R(k)$

}

set $i = 0$

while ($i < MAX$) {

Solve master problem MP with $R(k)$ for all k and get optimal dual variables π

set counter = 0

for k from 1 to K

{

Solve subproblem SP_k to get route r with negative reduced cost that starts from and ends at depot k

if (r has negative reduce cost) {

add r to $R(k)$

put counter $\leftarrow$ counter + 1

}

}

if (counter = 0) {

Exit

}

put $i \leftarrow i + 1$

}

End

In the initialization stage, the CG algorithm generates a set of routes that cover all requests in the problem. Each route starts from a depot, connects with a segment whose pickup point belongs to the same depot and then returns to the depot. In each iteration, the algorithm first solves the master problem as an LP and obtains optimal values of □ and the dual variables π. Next, it tries to find negative reduced cost route for each subproblem. If such routes are found, they are added to the master problem, the counter is incremented, and the computations continue. If no such route/solution can be found within a given time limit by solving the subproblems, then the program stops. In the implementation, a maximum of 300 sec was allotted for each subproblem. Alternatively, when the total number of iterations exceeds the maximum limit, where $MAX = 5000$, the program stops as well.

Table A.1 highlights the fractional results for the CG algorithm as well as the integer (non-optimal) solutions obtained with CPLEX. For the smaller instances 1 and 2, the CG results are reasonable compared with the results CPLEX. The CG objective function value for instance 1 is 6.62 which is lower than the objective value of the exact model; that is, $7 + 1.34 \times 0.01 = 7.0134$. CG provides a good lower bound for the first two instances. For the larger instances 3 and 4, the CG results are poor. Curiously, though, the second instance required less runtime than the third instance. This apparent contradiction was due to the inability of CPLEX to find a feasible solution with a negative objective function value within the given runtime limit of 300 sec for each subproblem. Therefore, the CG algorithm terminated after only a few iterations. The fourth instance exhibited the same behavior and ran for only seven CG iterations. At termination for instances 3 and 4, many of the initial routes remained in the final solution.

Table A.1. Column generation computational results

Instance	# of segments	Column Generation		CPLEX		
		Objective value	Runtime (sec)	# of PIVs needed	Total travel time (hours)	Runtime (sec)
4 pm – 4:05 pm Sat.	30	6.62	16157	7	1.34	6400
4 pm – 4:12 pm Sat.	60	8.41	46894	10	2.56	6400
4 pm – 4:37 pm Sat.	120	65.25	14770	11	5.28	6400
4 pm – 5:19 pm Sat.	240	149.60	3670	14	10.05	6400

The computational results in Table A.1 indicate that the CG-based algorithm is not able to solve our problem instances due to their size. Consequently, we abandoned this approach.

References

Ahuja, R. K., Cunha, C. B., & Sahin, G. (2005). Network models in railroad planning and scheduling. *TutORials in operations research*, *1*, 54-101.

Arabeyre, J. P., Fearnley, J., Steiger, F. C., & Teather, W. (1969). The airline crew scheduling problem: A survey. *Transportation Science*, *3*(2), 140-163.

Araque, J.R., Kudva, G., Morin, T. L., & Pekny, J. F. (1994). A branch-and-cut algorithm for vehicle routing problems. *Annals of Operations Research*, *50*(1), 37-59.

Balakrishnan, A., Kuo, A., & Si, X. (2014). Real-time crew assignment in double-ended districts with Primary-Secondary queues, *working paper*, *The University of Texas, Austin.*

Balakrishnan, & Si, X. (2016). Crew Planning with Uncertainty in Train Arrival and Departure Times, *working paper*, *The University of Texas, Austin.*

Bard, J. F., Kontoravdis, G., & Yu, G. (2002). A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, *36*(2), 250-269.

Barlow, R. E., & Heidtmann, K. D. (1984). Computing k-out-of-n system reliability. *IEEE Transactions on Reliability*, 33, 322–323.

Bent, R., & Van Hentenryck, P. (2006). A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research*, *33*(4), 875-893.

Ben-Tal, A., & Nemirovski, A. (2002). Robust optimization–methodology and applications. *Mathematical Programming*, *92*(3), 453-480.

Berbeglia, G., Cordeau, J. F., Gribkovskaia, I., & Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *Top*, *15*(1), 1-31.

Berbeglia, G., Cordeau, J. F., & Laporte, G. (2012). A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. *INFORMS Journal on Computing*, *24*(3), 343-355.

Bodin, L., Golden, B.L., Assad, A., & Ball, M. (1983). Routing and scheduling of vehicles and crews: The state of the art. *Computers & Operations Research*, *10*(2), 63-211.

Cappanera, P., & Gallo, G. (2004). A multicommodity flow approach to the crew rostering problem. *Operations Research*, 52(4), 583-596.

Caprara, A., Fischetti, M., Toth, P., Vigo, D., & Guida, P. L. (1997). Algorithms for railway crew management. *Mathematical programming*, *79*(1-3), 125-141.

Caprara, A., Fischetti, M., Guida, P. L., Toth, P., & Vigo, D. (1999). Solution of large-scale railway crew planning problems: The Italian experience. *Computer-aided transit scheduling* (pp. 1-18). Springer Berlin Heidelberg.

Caprara, A., Kroon, L., Monaci, M., Peeters, M., & Toth, P. (2007). Passenger railway optimization. *Handbooks in operations research and management science*, *14*, 129-187.

Chen, S. X., & Liu, J. S. (1997). Statistical applications of the Poisson-binomial and conditional Bernoulli distributions. *Statistica Sinica*, 875-892.

Cherkesly, M., Desaulniers, G., Irnich, S., & Laporte, G. (2015). Branch-Price-and-Cut Algorithms for the Pickup and Delivery Problem with Time Windows and Multiple Stacks. *European Journal of Operational Research*.

Cordeau, J. F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, *54*(3), 573-586.

Deng, Y., & Bard, J. F. (2011). A reactive GRASP with path relinking for capacitated clustering. *Journal of Heuristics*, *17*(2), 119-152.

Derigs, U., & Döhmer, T. (2008). Indirect search for the vehicle routing problem with pickup and delivery and time windows. *OR Spectrum*, *30*(1), 149-165.

Desaulniers, G., Lessard, F., & Hadjar, A. (2008). Tabu search, partial elementarity, and generalized *k*-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, *42*(3), 387-404.

Desrochers, M., Desrosiers, J., & Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, *40*(2), 342-354.

Dumas, Y., Desrosiers, J., & Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, *54*(1), 7-22.

Feo, T. A., & Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, *6*(2), 109-133.

Flier, H., Gaurav, A., & Nunkesser, M. (2007). Combinatorial aspects of move-up crews. *Operations Research Proceedings 2008*: 569.

Fu, M. C. (1994). Optimization via simulation: A review. *Annals of Operations Research*, *53*(1), 199-247.

Gehring, H., & Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of EUROGEN99*, Vol. 2, 57-64. Springer Berlin.

Gopalakrishnan, B., & Johnson, E. L. (2005). Airline crew scheduling: state-of-the-art. *Annals of Operations Research*, *140*(1), 305-337.

Gorman, M. F., & Sarrafzadeh, M. (2000). An application of dynamic programming to crew balancing at Burlington Northern Santa Fe Railway. *International Journal of Services Technology and Management*, *1*(2), 174-187.

Gutiérrez-Jarpa, G., Desaulniers, G., Laporte, G., & Marianov, V. (2010). A branch-and-price algorithm for the vehicle routing problem with deliveries, selective pickups and time windows. *European Journal of Operational Research*, *206*(2), 341-349.

Hasama, T., Kokubugata, H., & Kawashima, H. (1998). A heuristic approach based on the string model to solve vehicle routing problem with backhauls. In Proceedings of the *5th World Congress on Intelligent Transport Systems* (*ITS*), Seoul, 1998.

Hong, Y. (2011). On computing the distribution function for the sum of independent and non-identical random indicators. *Dep. Statit., Virginia Tech, Blacksburg, VA, USA, Tech. Rep. 11_2*.

Hong, L. J., & Nelson, B. L. (2009, December). A brief introduction to optimization via simulation. In *Winter Simulation Conference* (pp. 75-85). Winter Simulation Conference.

Ioannou, G. (2007). An integrated model and a decomposition-based approach for concurrent layout and material handling system design. *Computers & Industrial Engineering*, *52*(4), 459-485.

Jespersen-Groth, J., Potthoff, D., Clausen, J., Huisman, D., Kroon, L., Maróti, G., & Nielsen, M. N. (2009). *Disruption management in passenger railway transportation* (pp. 399-421). Springer Berlin Heidelberg.

Jütte, S., Albers, M., Thonemann, U. W., & Haase, K. (2011). Optimizing railway crew scheduling at DB Schenker. *Interfaces*, *41*(2), 109-122.

Kliewer, N., Mellouli, T., & Suhl, L. (2006). A time–space network based exact optimization model for multi-depot bus scheduling. *European journal of operational research*, *175*(3), 1616-1627.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, *59*(3), 345-358.

Lau, H. C., & Liang, Z. (2002). Pickup and delivery with time windows: Algorithms and test case generation. *International Journal on Artificial Intelligence Tools*, *11*(03), 455-472.

Lenstra, J. K., & Rinooy Kan, A. H. G. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, *11*(2), 221-227.

Li, H., & Lim, A. (2003). A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, *12*(02), 173-186.

Liu, R., Xie, X., Augusto, V., & Rodriguez, C. (2013). Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. *European Journal of Operational Research*, *230*(3), 475-486.

Mulvey, J. M., Vanderbei, R. J., & Zenios, S. A. (1995). Robust optimization of large-scale systems. *Operations research*, *43*(2), 264-281.

Nanry, W.P., & Barnes, J.W. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, *34*(2), 107-121.

Parragh, S.N. (2011). Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C: Emerging Technologies*, *19*(5), 912-930.

Parragh, S.N., Doerner, K.F., & Hartl, R.F. (2008). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, *58*(1), 21-51.

Potthoff, D., Huisman, D., & Desaulniers, G. (2010). Column generation with dynamic duty selection for railway crew rescheduling. *Transportation Science*, *44*(4), 493-505.

Ropke, S., & Cordeau, J.F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, *43*(3), 267-286.

Rosenberger, J. M., Schaefer, A. J., Goldsman, D., Johnson, E. L., Kleywegt, A. J., & Nemhauser, G. L. (2002). A stochastic model of airline operations. *Transportation science*, *36*(4), 357-377.

Şahin, G., & Yüceoğlu, B. (2011). Tactical crew planning in railways. *Transportation Research Part E: Logistics and Transportation Review*, *47*(6), 1221-1243.

Savelsbergh, M. W., & Sol, M. (1995). The general pickup and delivery problem. *Transportation science*, *29*(1), 17-29.

Schneider, M., & Henning, K. F. (2014). *The vehicle routing problem with time windows and driver-specific times* (No. 65941). Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies, Darmstadt, Germany.

Schaefer, A. J., Johnson, E. L., Kleywegt, A. J., & Nemhauser, G. L. (2005). Airline crew scheduling under uncertainty. *Transportation science*, *39*(3), 340-348.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming—CP98* (pp. 417-431). Springer, Berlin.

Shebalov, S., & Klabjan, D. (2006). Robust airline crew pairing: Move-up crews. *Transportation science*, *40*(3), 300-312.

Sohoni, M., Lee, Y. C., & Klabjan, D. (2011). Robust airline scheduling under block-time uncertainty. *Transportation Science*, *45*(4), 451-464.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, *35*(2), 254-265.

Solomon, M. M., & Desrosiers, J. (1988). Survey Paper-Time Window Constrained Routing and Scheduling Problems. *Transportation science*, *22*(1), 1-13.

USPS report (2013), *Lessons Learned from Mail Processing Network Rationalization Initiatives*. Retrieved Nov. 28, 2015, from https://www.uspsoig.gov/sites/default/files/document-library-files/2015/no-ma-13-004.pdf

Vaidyanathan, B., Jha, K.C., & Ahuja, R.K. (2007). Multicommodity network flow approach to the railroad crew scheduling problem, *IBM J. Res. Development 51(3)*, 325-344.

Van der Bruggen, L. J. J., Lenstra, J. K., & Schuur, P. C. (1993). Variable-depth search for the single-vehicle pickup and delivery problem with time windows. *Transportation Science*, *27*(3), 298-311.

Vance, P. H., Barnhart, C., Johnson, E. L., & Nemhauser, G. L. (1997). Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, *45*(2), 188-200.

Veelenturf, L. P., Potthoff, D., Huisman, D., & Kroon, L. G. (2012). Railway crew rescheduling with retiming. *Transportation research part C: emerging technologies*, *20*(1), 95-110.

Xiang, Z., Chu, C., & Chen, H. (2006). A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *European Journal of Operational Research*, *174*(2), 1117-1139.

Yen, J. W., & Birge, J. R. (2006). A stochastic programming approach to the airline crew scheduling problem. *Transportation Science*, *40*(1), 3-14.

Yu, G., Argüello, M., Song, G., McCowan, S. M., & White, A. (2003). A new era for crew recovery at continental airlines. *Interfaces*, *33*(1), 5-22.

Zhang, X., & Bard, J.F. (2005). Equipment scheduling at mail processing and distribution centers. *IIE Transactions on Scheduling & Logistics 37*(2), 175-187.