Drayage Optimization in Truck/Rail Networks

A Thesis
Presented to
The Academic Faculty

by

Yetkin Ileri

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Industrial and Systems Engineering
Georgia Institute of Technology
November 2006

Drayage Optimization in Truck/Rail Networks

Approved by:

Date Approved: November 20, 2006

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Daily drayage operations involve moving loaded or empty equipment between customer locations and rail ramps. Drayage orders are generally pickup and delivery requests with time windows. The repositioning of empty equipment may also be required in order to facilitate loaded movements. The drayage orders are satisfied by a heterogeneous fleet of drivers. Driver routes must satisfy various operational constraints.

In the first part of the dissertation, our goal is to minimize the cost of daily drayage operations in a region on a given day. We present an optimization methodology for finding cost-effective schedules for regional daily drayage operations. The core of the formulation is a set partitioning model whose columns represent routes. Routes are added to the formulation by column generation. We present numerical results for real-world data which demonstrate that our methodology produces low cost solutions in a reasonably short time.

The second part of the dissertation addresses minimizing total empty mileage when driver capacity is not restrictive and new orders are added to the problem in an online fashion. We present a lower bound for the worst case guarantee of any deterministic online algorithm. We develop a solution methodology and provide results for the performance of different scheduling policies and parameters in a simulated environment.

In the third part of the dissertation, we study a system with one rail ramp and one customer location which is served by a single driver. The problem has discrete time periods and at most one new order is released randomly each time period. The objective is to maximize the expected number of orders covered. With this simple problem, we seek to learn more about route planning for a single driver under uncertainty. We prove that carrying out an order ready to be picked up at the driver's current location is optimal for the case with one customer location. We show that the structure of the optimal policies is not simple and depends on various parameters. We devise a simple policy which yields provably near-optimal results and identify a case for which that policy is optimal.

CHAPTER I

INTRODUCTION

1.1 Introduction

Intermodal freight transportation refers to the transportation of freight in a container or vehicle, using multiple modes of transportation (rail, ocean carrier, and truck), without any handling of the freight itself when changing modes [48]. In truck/rail intermodal transportation, the freight is first loaded into a trailer or container which is moved from the shipper to a nearby rail ramp by truck. Later, the freight is conveyed by rail to a ramp close to the freight's destination. Finally, the freight is delivered to the consignee by truck. Drayage refers to the truck portion of truck/rail intermodal transportation. The word "drayage" originates from "dray" which has been used to describe "a low, strong cart without fixed sides, for carrying heavy loads" [15]. Historically, drays were wheeled vehicles usually pulled by animals and used to transport heavy goods [47].

Railroads provide the most cost-efficient transportation on land and trucks provide flexible pickup and delivery services and faster transportation. Truck/rail intermodal transportation aims to combine the best features of both modes. Since the freight itself remains sealed in a trailer or a container when the unit is transferred to another mode, the risk of damage and handling cost and time are reduced.

The number of trailers and containers moved via truck/rail intermodal transportation has been increasing steadily. The Association of American Railroads (AAR) reports that rail intermodal traffic rose from from 3.1 million trailers and containers in 1980 to 11.7 million in 2005 [3]. The container and trailer intermodal traffic on U.S. railroads is given in Figure 1. The container traffic seems to be the driving force in the increase of the intermodal traffic.

Truck/rail intermodal transportation provides a cheaper way than truck alone for moving freight over long distances. The rail part of an intermodal move needs to be long enough

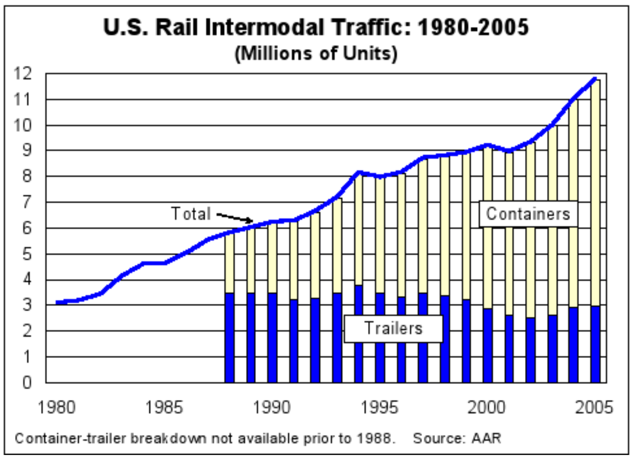


Figure 1: Intermodal traffic from 1980 to 2005.

so that the savings of using rail outweigh the overhead of drayage and terminal cost at both the origin and the destination. Although short compared to the rail portion, drayage constitutes a significant component of the cost of truck/rail intermodal transportation. Morlok and Spasovic [30] estimated that as much as 40% of the total cost of a 900 mile movement is incurred in the drayage portion. Efficient management of drayage operations stands out as an important issue in intermodal transportation. Because of drayage and terminal costs, over-the-road transportation is cheaper than intermodal for short hauls. Reduction in drayage cost will have a positive impact on the intermodal market share since the breakeven haul length will become smaller.

1.2 Drayage Operations

In this thesis, we look at daily drayage operations from a carrier's perspective. Daily drayage operations in a region involve moving loaded containers and trailers between customer locations and rail ramps within that region. A request for moving full equipment from a shipper to a rail ramp or from a rail ramp to a consignee is referred to as a *drayage order* or *dray*. A drayage order is called *inbound* if it originates at a rail ramp and *outbound* if it originates at a shipper. Drayage orders may include intermediate stops between the rail ramp and the customer location. The stop at a customer location has time windows given by the operating hours of the location and the requirements of the drayage order. The latest time for the delivery at a rail ramp is determined by the cut-off time of the train leaving the rail ramp. The earliest time for a pick up at a rail ramp is determined by the available time, i.e., the time when the equipment is removed from the rail and available for pickup.

Sometimes full equipment needs to be moved from one rail ramp to another rail ramp by truck. Another type of drayage movement is the transportation of a full trailer from a shipper location to a staging location, where the trailer is stored temporarily before another driver delivers it to a rail ramp. Such drayage movements, which are not inbound or outbound and take place within the region, are called *staging movements*. We will not consider staging movements in this study.

Freight is moved in trailers and containers. The trailers are owned by the carrier. Trailers and annually-leased containers are used freely by the carrier. Containers which are not leased by the carrier must be taken back to the rail ramps from which they entered the market once unloaded at the consignee location. Containers need to be put on chassis for road transport. Typically, the railroads are responsible for ensuring chassis availability. In this thesis, we limit the equipment type to trailers.

Empty trailers can be located at some customer locations, equipment yards and operating centers. There are two types of stops at a customer location–"live" stops and "drop-and-hook" stops. In a live pickup, the driver picks up an empty trailer before arriving at the shipper location and waits during loading at the shipper. Live stops are sometimes called "stay-with" stops. In a drop-and-hook pickup, the driver drops an empty trailer and picks up a pre-loaded trailer.

Relocation of empty trailers may be necessary to facilitate the movement of loaded trailers. For instance, a driver has to pick up an empty trailer before going to a live pickup at a shipper location following a delivery at a rail ramp. An empty trailer can be obtained from any location with a trailer pool.

There are two types of drivers–company and third party. Company drivers work daily shifts and start and end at their parking locations. Company drivers and third party drivers have different cost structures. Company drivers are employed by the carrier and third party drivers are hired on a job-by-job basis. Planning daily drayage operations involves creating routes for company and third party drivers so that all drayage orders are covered. The schedule must satisfy time windows of the stops, and the restrictions on the daily shifts of the drivers.

1.3 Literature on Drayage Operations

A review of current literature and opportunities for operations research in intermodal transportation is given in [29]. In that paper, research areas are categorized according to the decision maker and the time horizon. The decision makers can be drayage, terminal, network and intermodal operators and the time horizon of the problems are strategic, tactical and operational. The daily dray planning problem described in this thesis can be classified as an operational problem with drayage operator as the decision maker.

1.3.1 Strategic and Tactical Research Topics in Drayage Operations

The earlier works in drayage optimization are on the benefits of cooperation between drayage companies. In [30] and [40], the impact of central planning on cost and service quality is investigated. An integer programming model is solved to plan trailer and tractor movements centrally. Estimated cost savings of 43% to 63% and improvements in service quality can be achieved with centralized planning. A similar problem is addressed in [45].

The selection of rail ramps for intermodal orders is studied in [43]. Two heuristics are tested for performance on 40 scenarios. The first heuristic chooses the origin and the destination ramps for an order so that out-of-route miles are minimized. The second heuristic minimizes total miles which include out-of-route miles and empty miles. The scenarios for which one heuristic outperforms the other are identified.

In [23], the problem of ensuring chassis availability at rail ramps is studied. Due to the flow of containerized freight, chassis shortage may occur in some ramps while chassis availability is high at other ramps. Chassis can be transported by train (and by truck between ramps in the same region) to meet demand. Given the supply and demand for chassis at each ramp over a number of periods, the objective is to find a minimim cost solution for chassis redistribution. The problem is modeled as a time based network transportation problem and a software system is built based on the model.

These examples of research in drayage operations are at the strategic or tactical level. Before we move on to the literature on drayage at the operational level, we present an overview of some routing and scheduling models that appear in the drayage literature.

1.3.2 Related Routing Problems

In the Vehicle Routing Problem with Time Windows (VRPTW), the goal is minimizing the cost of servicing customers located around a central depot, where vehicle tours start and end. The customer demand is known and it must be serviced by only one vehicle. The total demand of the customers on a vehicle's route cannot exceed the capacity of the vehicle. Customers must be serviced within a time window, i.e., after the earliest service time and before the service deadline.

The VRPTW is NP-hard and finding a feasible solution to the VRPTW with a fixed fleet size is a NP-complete problem [36]. The exact solution methods for VRPTW are based on column generation, Lagrangian relaxation or branch-and-cut. In [13], a column generation based method is used for modeling VRPTW as a set partitioning problem. The set of customers is partitioned into subsets, and each subset is serviced by a single vehicle. Each column represents a route and the number of feasible routes can be very large. So, initially the model is solved with a limited number of columns and a shortest path subproblem with time window and capacity constraints is solved to identify routes that can potentially improve the solution. In [24] and [26], a Lagrangian relaxation is used on the requirement that all customers must be serviced. Then, the problem decomposes into capacitated shortest path problems for each vehicle. A bundle method is used to find the optimal Lagrange multipliers. In the branch-and-cut approach, the problem is formulated as a mixed integer program (MIP). The linear programming relaxation of the model is solved and if the solution is fractional, the model is resolved after the addition of valid inequalities. Branching takes place when no more valid inequalities are added to the model. A branch-and-cut procedure for the VRPTW is given in [4]. A 2-path cut for the VRPTW is first used in [25], which is developed further in [11]. A comprehesive overview of the exact and inexact solution methods for the VRPTW can be found in [12]. More recent surveys of local search and metaheuristics for the VRPTW can be found in [8] and [9].

The Multiple Traveling Salesman Problem with Time Windows is a relaxation of the VRPTW. Vehicle capacity is unrestricted. Variations of column generation approaches have been used for this problem as well.

The Pickup and Delivery Problem with Time Windows (PDPTW) is a generalization of the VRPTW. In the PDPTW, the customer demand is a transportation request from a known origin to a known destination. The other attributes of the problem remain the same. Each vehicle has a certain capacity, there are time windows at each stop, and the fleet of vehicles use the same location as the base. The VRPTW is a special case wherein all transportation requests originate at customer locations and end at the central depot. A comprehensive review of the general pickup and delivery problem is presented in [37]. Colum generation is generally used for solving the PDPTW and similar problems in [49], [16], and [38].

1.3.3 Related Literature on Drayage Optimization at Operational Level

An iterative solution procedure for solving the asymmetric multiple traveling salesman problem with time window constraints (m-TSPTW) is described in [46]. The solution method is developed with local truckload pickup and delivery problems in mind and designed for problem instances which have a small number of tasks assigned to each server, which is the case in drayage operations. Only pickup time windows are considered. So, when an order is converted into a node by identifying its origin and destination, the node has a single time window. The distance matrix for the nodes is asymmetric. The time windows are discretized and at each iteration of the solution method, two versions of the problem, over-constrained and under-constrained versions are solved. The solution to the over-constrained problem provides a feasible solution, while the optimality gap provided by the two solutions informs the decision maker whether to continue searching or to implement the best solution found so far. A specific time window partitioning scheme is used to ensure that the cost of solutions found are monotonically non-increasing.

In [22], container movements by trucks with time windows at origins and destinations are modeled as an asymmetric m-TSPTW with a limit on the work shift of a driver. A two-phase exact algorithm based on dynamic programming (DP) is used to find an optimal solution. In the first phase, feasible routes are enumerated using DP and a set covering problem is solved to find the best selection of routes. For larger problems, a hybrid methodology consisting of DP with genetic algorithms is used. Both methods are compared with an insertion heuristic.

An application of column generation for drayage can be found in [39], where empty trailer repositioning to facilitate loaded movements is modeled using flexible orders. A flexible order is a pickup and delivery request that specifies the origin or the destination, but not both. Different from our assumptions on drayage operations, "live" and "drop-and-hook" stops are allowed to be served by two drivers in this work. For instance, a driver can bring an empty trailer to a shipper for a "live" pickup and then leave, while a second driver arrives at the shipper later to pickup the loaded trailer. The model is a multi-resource routing problem with flexible tasks and the solution method is a column generation-based scheme with an insertion heuristic. The solution value of the set partitioning problem is compared with a lower bound obtained by solving an assignment problem. Various column generation schemes are tested on 20 instances from the Chicago area.

1.4 Uncertainty in Drayage Operations

An important part of daily drayage operations is uncertainty. New orders are called in and some orders are cancelled or changed. The availability of drivers is also dynamic; drivers may run into some mechanical problems or other obstacles which may make them unavailable. Unexpected delays in transit and loading/unloading may require changes to the original plan.

Dispatchers can follow a driver's status via onboard communication systems. Dispatchers do not give full day assignments to drivers. Instead, they plan 3-4 hours into the future, so that the drivers know where to go next. As the day progresses, dispatchers update driver assignments. Dispatchers try to balance the workload among company drivers, and make sure that company drivers get enough tasks assigned before third party companies are utilized. A fast decision support system which can handle the dynamic changes and account for uncertainties can be very helpful to dispatchers in making suggestions and comparing different solutions.

1.5 Literature on Routing With Uncertainty

In [27], a method for real-time dispatching of automobile service units with soft time windows is presented. The underlying VRPTW is solved repeatedly by a highly tuned column generation-based method. The convergence of the method is accelerated by dynamic pricing control. Due to the nature of the problem, routes that cover a limited number of orders are favored by optimal solutions. This fact enables the use of a set partitioning formulation with column generation in a real-time environment.

A stochastic and dynamic model for the pickup and delivery problem is studied in [42]. The demand follows a Poisson process. The pickup and delivery locations are distributed uniformly and are independent. The objective is to minimize the expected time in system for demands. Three policies are compared and the nearest neighbor policy outperforms the other policies.

A generic real-time multivehicle truckload pickup and delivery problem is studied in [50]. Various cost types such as cost of empty travel, delays and order rejections are included. An MIP formulation is developed for the offline problem. Two of the policies are based on reoptimization and outperform the local heuristic rules in simulation.

Probabilistic information on future customer demands is used in [6] for solving a dynamic vehicle routing problem with time windows. In the Multiple Plan Approach (MPA), routing plans that are compatible with current decisions are continously generated. At each event, a list of plans and a distinguished plan is updated in MPA. The distinguished plan is chosen according to a consensus function such that the plan selected has the most similar routing to other plans in the list. The idea is based on a least commitment strategy, which is used in the artificial intelligence community. In the Multiple Scenario Approach (MSA), routing plans for multiple future scenarios are generated by sampling potential orders from probability distributions. MSA and MPA are compared to a greedy approach and the best performance is given by MSA in general. Both the consensus function and utilization of stochastic information seem to be important in getting good solutions.

Anticipatory route selection is investigated in [44]. The problem setup is representative of less-then-truckload operations. The goal is to construct a route with minimum expected total cost between two given nodes while servicing random pickup requests at intermediate nodes. The total cost is reduced by the reward gained from servicing customers. Each customer node can have at most one reward, and the likelihood of a pickup request is known as a function of time. The problem is modeled as a Markov decision process. The optimal policy utilizes the stochastic information to choose the route that can accommodate potential future pickups. The benefits of anticipation are demonstrated by comparing the optimal anticipatory policy with a reactive policy that does not take into account potential future orders.

In [41], the dynamic vehicle routing problem is simplified with the assumption that a resource (container, vehicle, or driver) can serve only one task in one period. Under this assumption, an adaptive, nonmyopic algorithm that involves iteratively solving sequences of assignment problems is developed. The value of advance information is tested in this study as well.

Random customer demands, travel times and user noncompliance in dynamic routing and scheduling of truckload transportation are studied in [32]. The impact of each type of uncertainty on the value of optimal myopic solutions is investigated separately. The problem is modeled as a dynamic assignment problem in which all known orders are assigned to drivers. The degree of dual variable utilization is controlled by the dual variable discounting factor. When the discounting factor is 1.0, the solution of the model is an exact optimal solution for covering known orders. When the discounting factor is 0, the solution of the model is equivalent to the greedy solution. The simulation runs suggest that the most robust value for dual discount factor is 0.75 for all three types of uncertainty. The results also suggest that uncertainty reduces the value of exact optimal solutions in a dynamic setting.

In [10], the home delivery problem is defined to model grocery delivery services. The solution should specify which deliveries to accept or reject as well as the time slot for the accepted deliveries while maximizing expected total profit. Delivery requests are revealed dynamically to the decision maker and the probability distribution of the latest time to request a delivery is known for each customer. An insertion heuristic with various criteria is tested on randomly generated instances.

In [17], a dynamic dispatching system that handles random arrivals of customers, who require a transport from a pickup location to a destination, is considered. The dispatching system processes online traffic information dynamically to calculate shortest paths. An assignment problem based resolution strategy gives the best performance. The cost matrix of the assignment problem includes factors such as unattractiveness of a location.

In [18], the problem is vehicle routing with stochastic demand and customers and in [28], the problem is vehicle routing with stochastic travel times. The two papers are similar in the sense that a two-stage stochastic formulation with a simple recourse function is used in both. More information on dynamic vehice routing and dispatching research can be found in [19], [33], and [7].

The literature on routing problems can be grouped into two classes depending on availability of stochastic information on future changes. For problems that are dynamic with no stochastic information, the solution approaches typically choose to reoptimize when new information becomes available using exact optimization or heuristics. If stochastic information on future changes is available, the solution methods try to take advantage of the extra information in various ways. Due to the short time horizon in daily drayage operations and high variability from day-to-day, we think an online model with no prior information on future changes is more appropriate for our problem. Our solution method differs from other methods of solving dynamic routing problems in trying to plan for future changes in the absence of stochastic information. In order to gain more insights that can be useful in the dynamic problem with multiple drivers, we study the theoretical properties of a stochastic drayage problem with a single driver and one customer. The analysis for the problem with a single driver and one customer location is similar to the analysis in [44], which models less-than-truckload pickups instead of pickup and delivery requests.

1.6 Contributions

In the next chapter, we address the modeling of daily drayage operations with static data, and provide a formulation which can be heuristically solved to near-optimality in reasonable time. The model captures important aspects of daily drayage operations such as empty relocation requirements. In the remainder of the thesis, we present and study two abstract models of drayage operations with dynamic addition of new orders, in order to build a foundation for larger and more comprehensive drayage models with uncertainty.

In Chapter 2, we study the Static Daily Drayage Problem. We present a column-generation based formulation for the problem. We show that instances based on the real-world data can be solved in reasonable time by caching calculations effectively. We devise an enumeration-based solution method that can solve smaller instances and we convert the enumeration-based method into a heuristic column generation-based method that can be used to solve larger instances.

The details of the problem and the model are given in Section 2.2. Enumeration-based and column generation-based solution methods and techniques used to improve running time are described in Section 2.3. Numerical results using historical real-world data are given in Section 2.4. A discussion of implementation challenges in Section 2.5 is followed by the conclusions in Section 2.6.

In Chapter 3, we study our first model with uncertainty, namely the Online Daily Drayage Problem. This chapter addresses minimizing total empty mileage when driver capacity is not restrictive and new orders are added to the problem in an online fashion. We present a lower bound for the worst case guarantee of any deterministic online algorithm. We develop a solution methodology based on the solution of the Static Daily Drayage Problem and provide results for the performance of different scheduling policies and parameters in a simulated environment.

The introduction and the motivation for the Online Daily Drayage Problem are presented in Sections 3.1 and 3.2, respectively. We describe the problem in detail in Section 3.3, which includes the simplifying assumptions in Section 3.3.1, the online nature of the problem in Section 3.3.2, feasibility issues in Section 3.3.3 and an example in Section 3.3.4. The first solution methodology and scheduling policies are described in Section 3.4. In Section 3.5, we define an ideal solution which we use to evaluate the performance of the solution method. In Section 3.6, we prove a lower bound for the performance guarantee of any deterministic online algorithm. The simulated performances of various scheduling policies are given in Section 3.7. We modify the solution methodology to reward solutions which allow for insertion of new orders into their schedule and introduce a new scheduling policy in Section 3.8. The new computational results are given in Section 3.9. The assumption of no prior knowledge of future orders is discussed in Section 3.10, followed by conclusions in Section 3.11.

In Chapter 4, we study a system with one rail ramp and one customer location which is served by a single driver. The problem has discrete time periods and at most one new order is released randomly each time period. The objective is to maximize the expected number of orders covered. With this simple problem, we seek to learn more about route planning for a single driver. We prove that carrying out an order ready to be picked up at the driver's current location is optimal for the case with one customer location. We show that the structure of the optimal policies is not simple. We devise a simple policy which yields provably near-optimal results. We identify a case for which this simple policy is optimal.

After the introduction and the description of the problem, we present a finite-horizon Markov decision process formulation for the problem in Section 4.3. We explain our method of calculating the optimal expected value and the observations we use to speed up the calculations in Section 4.4. In Section 4.5, we analytically study conditions under which the driver should wait or move to the other location when there are no orders in the system. We give analytical answers up to five periods and show that the answer depends on parameter values for five or more periods. The numerical calculations in Section 4.6 demonstrate that the parameter and the probabilities determine optimal policies in a complicated way. We present a provably near-optimal and simple-to-state policy in Section 4.7. In Section 4.8, we show that the policy can be optimal in a special case. The extension to two customer locations and other generalizations are given in Sections 4.9 and 4.10, respectively. The conclusions for the chapter are given in Section 4.11. We present our conclusions for the thesis and thoughts for future research directions in Chapter 5.

CHAPTER II

STATIC DAILY DRAYAGE PROBLEM

2.1 Introduction

In the Static Daily Drayage Problem, our goal is to minimize the cost of daily drayage operations in a region on a given day. Drayage operations involve moving loaded and empty equipment between rail ramps, shippers, consignees, and equipment drop lots. While drayage equipment can include containers as well as trailers, only trailers are considered in this study.

There are two general types of driver activities in the Static Daily Drayage Problem– carrying out drayage orders and repositioning empty trailers. Drayage orders involve moving loaded trailers and may contain requests for intermediate stops as well. The set of drayage orders is given and all orders must be satisfied, i.e., order rejection is not allowed. Repositioning of empty trailers may be required depending on the sequence and the types of the orders on a route. In some cases, a driver may have to pick up an empty trailer or drop an empty trailer before carrying out an order. Since the need for repositioning an empty trailer changes with the routing decisions, two routing plans covering the same set of orders may have a different set of empty trailer repositioning activities.

The drayage orders are covered by a heterogeneous fleet which includes company drivers and third party drivers. Driver routes must satisfy various operational constraints such as maximum work hours. The cost of a route depends on the type of driver executing the route. The cost of a driver may depend on various factors such as driver type, working hours, mileage, detention and minimum daily cost.

The details of the Static Daily Drayage Problem and our model are explained in Section 2.2. In Section 2.3, we present our solution methods. Numerical results for the performance of the methods on real-world data are given in Section 2.4. Section 2.6 gives our conclusions for this chapter.

2.2 Modeling Daily Drayage Operations

We create daily work schedules for a heterogeneous set of drivers so that all drayage orders are satisfied. Each order is composed of a pre-determined sequence of stops with disjoint time windows. The first stop of an order is a pickup and the last stop of an order is a delivery. Let O denote the set of orders.

Each driver has a designated starting location and ending location. Other driver attributes include maximum work hours, and earliest and latest start times. Drivers with the same attributes are grouped together. Let D denote the set of driver groups, and let d_j denote the number of drivers in group $j \in D$. Grouping drivers helps reduce the symmetry in the model.

A route describes a driver's daily work schedule from start to finish. Each route assignment pairs a route with a driver group. Let R denote the set of all feasible route assignments. Let R^j denote all feasible route assignments that can be carried out by driver group j. Since each route assignment is carried out by a unique driver group, R^j partitions R. Let R_i denote the set of route assignments that cover order i.

For each driver, a minimum daily cost is incurred regardless of the actual work carried out by the driver. We normalize the cost of a route assignment by subtracting the minimum daily cost, which is independent of the planning decisions. A plan is a collection of route assignments such that all drayage orders are covered exactly once without violating driver capacity. The cost of a plan is the total normalized cost of route assignments selected.

In the remainder of Section 2, we explain in detail various aspects of daily drayage operations, leading up to presenting our model. These aspects include drayage orders, drivers, trailer pools, constraints on route assignments, and costs.

2.2.1 Drayage Orders

A drayage order specifies a given sequence of stops. The stops are well defined; all of the attributes of a stop including location and duration are given as part of the order description. Stops may have multiple time windows.

The types of drayage stops are described in Table 1. "In tow pre-condition" describes what equipment, if any, the driver *must have* in tow upon arrival at the stop location. "In tow post-condition" describes what equipment, if any, the driver *will have* in tow after completing the stop. "Empty" means an empty trailer, "Loaded" means a loaded trailer and "None" means no trailer in tow.

Table 1: Drayage stop types

Type	Description	In Tow Condition	
		Pre	Post
DE&PL	Drop empty trailer & pick up preloaded trailer	Empty	Loaded
DL&PE	Drop loaded trailer & pick up empty trailer	Loaded	Empty
PL_W	Live loading	Empty	Loaded
DL_W	Live unloading	Loaded	Empty
PE	Pick up empty trailer	None	Empty
DE	Drop empty trailer	Empty	None
PL	Pick up loaded trailer	None	Loaded
DL	Drop loaded trailer	Loaded	None

The first four stop types in Table 1 can be grouped into two general classes called *drop-and-hook* and *live*. At a *drop-and-hook* stop (i.e., DE&PL or DL&PE), the driver drops one trailer and hooks to another. Drop-and-hook stops occur at customer locations with trailer pools. In general, the duration of a drop-and-hook stop is shorter than the duration of a live stop. At a *live* stop (i.e., PL_W or DL_W), the driver must wait while the driver's trailer is loaded or unloaded. Live stops usually occur at customer locations without trailer pools. We assume that rail ramps do not have trailer pools. Hence, only loaded trailers are picked up or dropped at rail ramps. Pickup (or delivery) at a rail ramp is denoted by PL (or DL). PE (pick up empty trailer) and DE (drop empty trailer) type stops can be at customer locations with trailer pools or any other location (e.g., a drop yard) with a trailer pool. At a drop-and-hook stop, the total number of trailers at the customer location remains unchanged, since one trailer is left and one is removed.

A drayage order is composed of a sequence of stops. Admissible sequences are given in Table 2. The first two types of orders are called *inbound* and the second two are called *outbound*. Inbound (outbound) orders start at a rail ramp (customer location) and end at a customer location (rail ramp). The last type of drayage order is an empty trailer relocation request. Drayage orders typically have two stops (one for pickup and one for delivery), but in some cases, also include one or more intermediate stops. For instance, a driver may have to visit two locations to pick up freight. By necessity, all intermediate stops are live stops. In Table 2, $[X]^*$ means stop type X can repeat 0 or more times.

Table 2: Admissible sequences for drayage orders

1:	PL	→	[DL_W]*	→	DL_W
2:	PL	→	[DL_W]*	→	DL&PE
3:	PL_W	→	[PL_W]*	→	DL
4:	DE&PL	→	[PL_W]*	→	DL
5:	PE	→	DE		

2.2.2 Drivers

All drivers with the same attributes are put in the same driver group. The attributes of a driver in group j are:

1. Earliest starting time, t_s^j
2. Latest starting time, $t_{s'}^j$
3. Starting location, l_s^j
4. Ending location, l_e^j
5. Maximum work hours, w_j
6. Driver type, g_j
7. Cost function, $f_j(.)$
8. Minimum daily cost, μ_j

Drivers can be one of two types–company driver (CD) and third party (TP). Each type can contain several driver groups. Minimum daily cost represents the fixed cost of having a driver on duty, whether or not that driver actually drives. For company drivers, $\mu_j > 0$ while for third parties, $\mu_j = 0$ since company drivers are carrier employees and third party drivers are paid on a job-by-job basis. The cost functions are explained in Section 2.2.5.

The starting location and the ending location of a driver is typically an operating center where the driver parks his tractor overnight. If a company driver is covering at least one order, then he has to start his route assignment between his earliest and latest starting times. The earliest and the latest starting times for third party drivers are set to the beginning of the time horizon and infinity, respectively, so that a third party driver can start a route assignment any time of the day. The work hours start accumulating when a driver leaves his starting location and stop when he parks at the ending location. For all drivers, there is a limit on the total work hours.

For simplicity, we assume that drivers start and end the day without a trailer. However, it is possible to relax this assumption by extending driver attributes to include starting and required ending trailer state.

2.2.3 Empty Trailers

Trailers are shared by a number of operations besides drayage including over-the-road and regional movements. In its current status, accurately modeling the trailer capacity requires stepping outside the scope of drayage operations. Therefore, we chose to assume that empty trailers are readily available at specified locations. Our model takes into account additional time and distance required to pick up or drop empty trailers, but not limitations on trailer availability.

2.2.4 Feasible Route Assignments

Each route is constructed for a specific driver group. Let D denote the set of driver groups. Let r be a route assignment for driver group j, i.e. $r \in R^j$. Let τ_s^r denote the starting time and τ_e^r denote the end time for the route. Then, the route must

- start at location l_s^j at time $\tau_s^r \in [t_s^j, t_{s'}^j]$, and
- end at location l_e^j at time τ_e^r such that $\tau_e^r - \tau_s^r \leq w_j$.

In addition to the constraints for the starting time and work hours, the schedule of r must also satisfy the time windows for the stops on the route.

A driver has to complete one order before starting another. In order to satisfy empty trailer requirements, the driver may have to make extra stops which are not part of any order. The driver may have to drop or pick up an empty trailer before executing the first stop of an order or before parking at the ending location. For instance, if the driver has no empty trailer and the next stop on the route is a DE&PL, the driver must pick up an empty trailer (PE) before deadheading (i.e., driving with an empty trailer in tow) to the next stop. Alternatively, if the driver has an empty trailer in tow and the next stop is a pickup at a rail ramp, the driver must drop the empty trailer and bobtail (i.e., drive with no trailer in tow) to the rail ramp. All possible cases are summarized in Table 3, where ℓ denotes the location for the next stop.

Table 3: Rules for adding extra stops to a route

In Tow	Next Stop		Extra Stop Type
None	DE&PL	$\Rightarrow$	PE at any pool location except ℓ
Empty	DE&PL	$\Rightarrow$	-
None	PL	$\Rightarrow$	-
Empty	PL	$\Rightarrow$	DE at any pool location
None	PL_W	$\Rightarrow$	PE at any pool location except ℓ
Empty	PL_W	$\Rightarrow$	-
None	PARK	$\Rightarrow$	-
Empty	PARK	$\Rightarrow$	DE at any pool location

Although all stops of an order have pre-determined locations, the locations of extra stops are flexible. Suppose that the next stop's type is DE&PL. Then, ℓ must be a pool location (refer to the discussion of drop-and-hook vs. live pickup in Section 2.2.1). Hence, the extra stop can be at any trailer pool location except ℓ. If the next stop's type is PL, then location ℓ does not have a trailer pool and the driver can go to any trailer pool to perform the extra stop. We assume that extra stops do not have time windows. In the "Extra Stop Type" column of Table 3, we use "-" when an extra stop is not required.

2.2.5 Objective

The type of a driver determines that driver's cost structure. The cost of route assignment r for a company driver in group j is given by the formula

$$\max\Big(\mu_j, h \cdot (\tau_e^r - \tau_s^r)\Big),$$

where h is the cost per hour for a company driver.

The cost of route assignment r for a third party driver from group j, denoted by u_{rj}, is the sum of mileage, placement, detention and bobtail costs. Placement and detention costs depend on the stops. Each stop of the type DE&PL, DE, DL&PE or DL_W is considered a placement and incurs a fixed cost. Let n_r be the number of placements in r and p_j be the cost of each placement. The placement cost is given by $u^1_{rj} = n_r \cdot p_j$. Carriers' mileage charges include an allowance to cover the time required to load and unload. When drivers are detained longer than two hours, carriers charge an additional detention charge. Detention cost is a non-decreasing function of stop duration. Let Δ_j denote the cost of detention per 15-minute period and δ_k denote the duration of stop k in minutes. If there are K^r stops in r, the detention cost is given by

$$u^2_{rj} = \sum_{k=1}^{K^r} \eta_k, \text{ where } \eta_k = \begin{cases} 0 & \text{if } \delta_k < 120, \\ \Delta_j \cdot \left\lfloor \frac{\delta_k - 120}{15} + 0.5 \right\rfloor & \text{otherwise.} \end{cases}$$

Let θ_j denote the cost per mile and m_r denote the total mileage excluding the first and last legs of the route. The mileage cost is given by $u^3_{rj} = m_r \cdot \theta_j$. In the last leg of the route, the third party driver bobtails back to a designated end location. The cost for this leg is calculated separately as bobtail cost. Let b_r denote the bobtail mileage. The bobtail cost is given by

$$u^4_{rj} = \begin{cases} 0 & \text{if } b_r < 50, \\ \theta_j \cdot b_r & \text{if } 50 \le b_r < 100, \\ \theta_j \cdot b_r + p_j & \text{if } b_r \ge 100. \end{cases}$$

The total cost for r is $u_{rj} = u^1_{rj} + u^2_{rj} + u^3_{rj} + u^4_{rj}$.

In summary, the cost function for driver group j is given by the formula:

$$f_j(r) = \begin{cases} \max\Big(\mu_j, h \cdot (\tau^r_e - \tau^r_s)\Big) & \text{if } g_j = CD \\ u_{rj} & \text{if } g_j = TP. \end{cases}$$

For each route assignment $r \in R^j$, we define the normalized cost of the route assignment,

c_r, to be $c_r = f_j(r) - \mu_j$. Hence,

$$c_r = \begin{cases} \max\Big(0, h \cdot (\tau_e^r - \tau_s^r) - \mu_j\Big) & \text{if } g_j = CD \\ u_{rj} & \text{if } g_j = TP. \end{cases}$$

The objective is to minimize the total normalized cost.

2.2.6 Formulation

We formulate the following integer program with 0-1 variables for each feasible route assignment. The output of the model is a minimum cost plan.

$$DRAYAGE(O, D, R): \quad \min \sum_{r \in R} c_r x_r \tag{1}$$

$$\sum_{r \in R_i} x_r = 1 \quad \forall i \in O \tag{2}$$

$$\sum_{r \in R^j} x_r \leq d_j \quad \forall j \in D \tag{3}$$

$$x_r \in \{0, 1\} \tag{4}$$

The constraints (2) ensure that all orders are satisfied. The constraints (3) enforce capacity for each driver group. Solutions to this model represent plans, where a route assignment r is in the plan if and only if x_r equals 1.

The main advantage of this formulation is that the complicated route feasibility constraints and cost functions are stated explicitly in the route assignments thereby avoiding the need to include complex side constraints. The main disadvantage is that the number of route assignments can be very large. However, it is not necessary to enumerate all of the variables beforehand.

2.3 Solution Method

A branch-and-bound method is used to solve this integer programming formulation. At the first stage, a subset of feasible routes is used to create an initial linear programming (LP) relaxation of the model. The LP relaxation is solved to optimality and dual values are obtained. The dual values are used to calculate the reduced costs for feasible route assignments (columns) which are generated by a column generator. Columns with negative reduced costs are collected for each driver group and are added to the formulation. After adding the new columns to the model, the resultant LP relaxation is re-solved and new dual values are obtained. This column generation cycle is repeated until no columns with negative reduced cost can be found. Then, integrality constraints are enforced and the integer programming model is solved by standard branch-and-bound techniques [31].

When the column generation is completed (i.e., no columns with negative reduced costs are found), the objective value of the final LP relaxation is the same as the optimal value of the LP relaxation with the full set of columns. On the other hand, the resulting solution for the integer program may not be optimal, since columns are not generated at every node of the branch-and-bound tree [5].

An alternative approach is to enumerate all of the columns up front and to solve the resultant integer program. This approach guarantees an optimal solution. However, enumeration becomes impractical as the number of orders grows. Another possibility is to generate columns throughout the tree; implementing such an approach is problematic, though, since commercial software such as CPLEX 9.0 does not support this option.

The description of our route enumeration algorithm is given in Section 2.3.1. In Section 2.3.2, we modify the enumeration algorithm to obtain a column generation scheme.

2.3.1 Enumerating Route Assignments

For every driver group, a search tree for finding feasible route assignments is created. The root node of the search tree corresponds to a dummy route covering no orders. The children of the root node represent route assignments that cover single orders. Branching continues by appending orders to the end of each route. Each node of the tree represents a unique subsequence of orders. We check for the existence of a feasible route by carrying out the orders in the given sequence, inserting extra PE or DE stops when necessary. We add feasible routes to the route list and continue branching on feasible routes that do not cover all orders. We prune the search tree at nodes corresponding to infeasible routes since any route beginning with the same sequence of orders must also be infeasible.

Given a fixed sequence of orders s_r, we check the feasibility of the route assignment r by first inserting any necessary empty trailer relocation stops into the route (see Section 2.2.4) and then searching for a feasible schedule that minimizes route duration. Whenever an empty trailer needs to be picked up or dropped to be able to continue the route, we choose the trailer pool that minimizes the time added to the route. Since we assume unlimited capacity for trailer pools and no time windows for extra stops, the additional time is simply the total time required for driving from the current stop to the pool location, picking up or dropping an empty trailer and driving from the pool location to the next stop.

After selecting pool locations for extra stops, we schedule the route such that its duration, $\tau_e^r - \tau_s^r$, is minimized. If the assignment duration is more than w_j, then the assignment is infeasible. When $g_j = CD$, minimizing route assignment duration allows us to choose the schedule that minimizes c_r for a fixed order sequence s_r.

We now present a small example to illustrate our enumeration scheme. After the example, we describe the elimination of dominated route assignments. We introduce the precedence feasibility matrix we use to speed up the enumeration. Finally, we present pseudocode for the enumeration-based method.

2.3.1.1 An Example

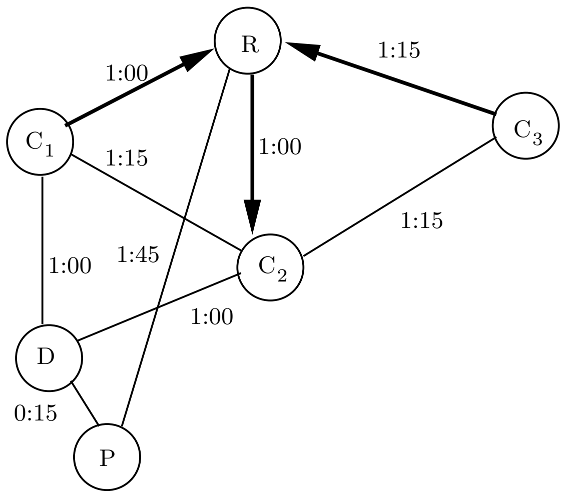


Figure 2: Example with three orders. The transit times are given in HH:MM format.

We use the example problem in Figure 2 to illustrate the construction of a search tree. The locations and the transit times are given in the figure. R denotes a rail ramp. None of the customer locations (C_1, C_2, and C_3) have trailer pools. Location D, a drop yard, has the only trailer pool. The problem includes three orders, which are represented by directed arcs. Details of these orders are given in Table 4. The duration for a PE or DE stop at location D is 15 minutes. Let location P be the starting and ending location for driver group 1. Drivers in this group start at 8:00 AM and can work at most 12 hours.

Table 4: Orders in the example problem

Order	Stop	Stop Type	Location	Time Win.	Duration (HH:MM)
Order 1	Stop 1	PL_W	C_1	-	1:30
	Stop 2	DL	R	-	0:30
Order 2	Stop 1	PL	R	[8:00 AM, 11:00 AM]	0:30
	Stop 2	DL_W	C_2	-	1:30
Order 3	Stop 1	PL_W	C_3	-	0:45
	Stop 2	DL	R	-	0:30

Suppose that we start branching with order 1. We check for the existence of a feasible route for r with $s_r = (1)$. The route for r starts at location P. In order to satisfy the in tow pre-condition for order 1, the driver must first go to location D to pick up an empty trailer (PE stop). After transporting order 1 from location A to the rail ramp, the driver leaves the ramp and travels to the park location P. The minimum total duration of the route is 6 hours and 15 minutes, which is less than the maximum work hours. There are no time windows for the stops of order 1. Therefore, r is feasible and we add it to the feasible route assignments list. Suppose that we continue branching by appending order 2 after order 1. The new route for q with $s_q = (1, 2)$ is the same as the route for r until the driver completes order 1. The earliest time that the driver can complete order 1 is 12:30 PM, but the latest pickup time for order 2 is 11:00 AM. Therefore, q is infeasible. We prune the search tree at the node associated with $s_q = (1, 2)$, and backtrack to the parent node with $s_r = (1)$. The search continues until all route assignments corresponding to the nodes of the tree are evaluated or pruned.

For our small example, the search tree for driver group 1 is given in Figure 3. Pruned nodes are outlined with dashed lines. Nodes corresponding to feasible route assignments are outlined with solid lines. We showed that the node labeled with (1,2) is pruned because the pickup time window of order 2 is violated. In most of the other pruned nodes, the maximum

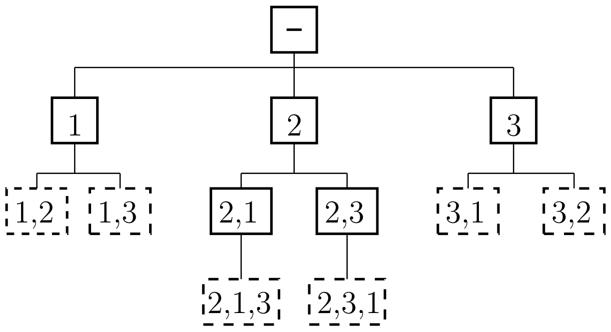


Figure 3: Search tree for driver group 1

route duration is violated. The depth of the search tree is bounded by the number of orders, which is 3 in this example. There are 5 feasible route assignments for group 1.

2.3.1.2 Dominated route assignments

We collect feasible route assignments for all driver groups into a single list and use this list to create the integer programming model. In order to reduce the number of columns in the model, dominated columns are deleted from the list.

Definition 1. *Route assignment r dominates route assignment q if*

- *r and q cover the same set of orders,*
- *r and q belong to the same driver group, and*
- *$c_r < c_q$.*

Eliminating dominated columns helps reduce memory usage and improves solution time.

Proposition 1. *Let $r, q \in R^{\alpha}$ for some $\alpha \in D$ with r dominating q. Let $OPT(O, D, R)$ denote the optimum cost for $DRAYAGE(O, D, R)$. Let R^* denote the set of route assignments in an optimal solution. Then, $OPT(O, D, R^*) = OPT(O, D, R \setminus \{q\})$.*

Proof. Suppose for a contradiction that $q \in R^*$. Then, $r \notin R^*$ since in a feasible solution each order is covered exactly once. Consider a new solution $\tilde{R} = (R^* \cup \{r\}) \setminus \{q\}$. The set of constraints (2) is still satisfied by $\tilde{R}$ since q and r cover the same set of orders. In the driver capacity constraint for group α, $x_r + x_q$ remains equal to one. We have

$$OPT(O, D, \tilde{R}) = OPT(O, D, R^*) + \underbrace{c_r - c_q}_{<0} < OPT(O, D, R^*),$$

which contradicts the fact that R^* is an optimal solution. Therefore, $q \notin R^*$. If $q \notin R^*$, then we have $R^* \subseteq R \setminus \{q\}$, which implies $OPT(O, D, R \setminus \{q\}) = OPT(O, D, R^*)$. $\square$

2.3.1.3 The precedence feasibility matrix

The feasibility check is the most time-consuming part of the enumeration step. The precedence feasibility matrix speeds up the enumeration process by allowing us to prune many infeasible combinations.

Definition 2. *For each driver group j, we define a precedence feasibility matrix F^j, which is a binary $|O| \times |O|$ matrix, where F^j_{ab} denotes the entry in the a^{th} row and the b^{th} column. F^j_{ab} is 0 if order b cannot follow order a on any feasible route assignment for group j and 1 otherwise.*

We initialize all entries of F^j to 1 except for F^j_{aa}, $\forall a \in O$ (since an order cannot appear on a route twice). We relax the empty relocation requirements and enumerate all routes that cover two orders. We set $F^j_{ab} = 0$ for routes r with $s_r = (a, b)$ that are infeasible even under relaxed assumptions. This procedure creates a valid precedence feasibility matrix under the assumption that transit times satisfy the triangle inequality. The precedence feasibility matrix is not necessarily symmetric since, with time windows, the fact that one order can precede another does not imply the reverse.

We relax the empty repositioning requirements for populating F^j because inserting an order between two orders may decrease the total time required to cover both orders in some cases. This unlikely case is illustrated in Figure 4 with an example. Suppose that we are considering route r with $s_r = (a, b)$ such that order a has a DL&PE which is followed by a PL for order b. Now, the driver must drop the empty trailer before picking up order b at the rail ramp. Route r may become infeasible if the closest trailer pool is too far away. Suppose that we insert an order c that must be moved from a location close to the delivery location of order a to the rail ramp from which order b originates. If the stop durations of order c do not take much time, the insertion of order c may create a feasible route assignment from an infeasible one. Though possible in theory, inserting such an order is not likely to improve chances of feasibility, since live loading is typically time consuming.

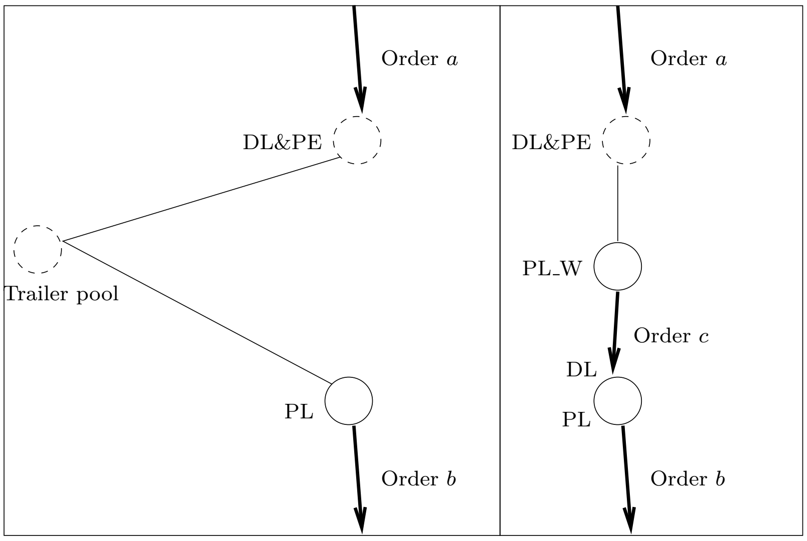


Figure 4: The insertion of an order may reduce the time required to complete a route assignment. The dashed circles denote locations with a trailer pool.

The precedence feasibility matrix can be used for pairs of orders which do not appear consecutively on a route. For example, consider a route assignment $r \in R^j$ with order sequence $s_r = (2, 1, 3)$. If the precedence feasibility matrix shows that order 3 cannot be carried out after order 2 (i.e., $F^j_{23} = 0$), then r and all route assignments formed by appending orders to the end of r must be infeasible. A more formal statement is in Proposition 2.

Proposition 2. *Given $r \in R^j$ for some $j \in D$ with $s_r = (a_1, a_2, \ldots, a_K)$, r is infeasible if there exists α and β where $1 \leq \alpha < \beta \leq K$, and $F^j_{a_\alpha a_\beta} = 0$.*

Proof. For a contradiction, suppose that route r with $s_r = (a_1, a_2, \ldots, a_K)$ is feasible. If we remove the repositioning requirements, the driver can simply skip empty repositioning stops. Route r is still feasible because transit times satisfy triangle inequality. Similarly, we can remove all of the orders of s_r except a_α and a_β while preserving feasibility. The resultant route assignment with $s_r = (a_\alpha, a_\beta)$ is feasible which contradicts the way $F^j_{a_\alpha a_\beta}$ is initialized. $\square$

2.3.1.4 Pseudocode for The Enumeration-based Method

In this section, we provide the general framework for our implementation of the enumeration-based algorithm. The pseudocode briefly describes the flow of our algorithm and how we use the techniques described in Section 2.3.1. We omit the implementation details for the sake of simplicity. Algorithm 1 outlines the depth first search (dfs) which uses cached results for best pool choices, the precedence feasibility matrix and the route assignment domination. We call the dfs algorithm for each driver group j with the corresponding precedence feasibility matrix, F^j. The pool cache, P, keeps the best pool choices. The non-dominated route assignments for driver group j are in R^j. The algorithm is recursive. We update the driver status (w), the sequence of orders (S) and the corresponding partial route (r) after appending an order to the route assignment and recursively call Algorithm 1 with updated parameters.

Algorithm 1 $dfs(S, F^j, P, R^j, w, r)$

```
1: for $a \in O$ do
2:   $r' \leftarrow r$, $w' \leftarrow w$
3:   for $s \in S$ do
4:     if $F^j_{sa} = 0$ then
5:       Go to line 1
6:     end if
7:   end for
8:   if An empty relocation is needed before moving order $a$ then
9:     Get the best trailer pool for $w'$ and $a$ from $P$
10:    if Adding the extra stop for the empty relocation is feasible then
11:      Add the new stop to $r'$, update driver status $w'$
12:    else
13:      Go to line 1
14:    end if
15:  end if
16:  for Each stop of order $a$ do
17:    if Adding the extra stop is feasible then
18:      Add the new stop to $r'$, update driver status $w'$
19:    else
20:      Go to line 1
21:    end if
22:  end for
23:  $s' \leftarrow (s, \{a\})$
24:  if An empty relocation is needed before going back to park location then
25:    Get the best trailer pool for $w'$ and the park location from $P$
26:    if Adding the extra stop for the empty relocation is feasible then
27:      Add the new stop to $r'$, update driver status $w'$
28:    else
29:      Go to line 1
30:    end if
31:  end if
32:  if Going back to park location is not feasible then
33:    Go to line 1
34:  else
35:    $\bar{r} \leftarrow r'$, add the final parking stop to $\bar{r}$
36:    Calculate the cost for $\bar{r}$
37:    for Each route assignment $q$ in $R^j$ do
38:      if $\bar{r}$ dominates $q$ then
39:        $R^j = R^j \setminus \{q\}$
40:      else
41:        if $q$ dominates $\bar{r}$ then
42:          Go to line 48
43:        end if
44:      end if
45:    end for
46:    $R^j \leftarrow R^j \cup \{\bar{r}\}$
47:  end if
48:  Recursively call $dfs(s', F^j, P, R^j, w', r')$
49: end for
```

Algorithm 2 enumerates the set of all feasible and non-dominated route assignments by calling dfs, populates the corresponding IP instance and solves the instance with the help of an IP solver.

Algorithm 2 $enumeration(O, D, L)$

```
1: INPUT: Set of orders, $O$, set of driver groups, $D$, set of locations, $L$
2: $R \leftarrow \emptyset$
3: for $c, d \in L$ do
4:     Initialize $P_{cd}$
5: end for
6: for $j \in D$ do
7:     for $a, b \in O$ do
8:         Initialize $F^j_{ab}$
9:     end for
10:    $S \leftarrow \emptyset$, $R^j \leftarrow \emptyset$
11:    Call $dfs(S, F^j, P, R^j)$
12:    $R \leftarrow R \cup R^j$
13: end for
14: Given $R$, $O$ and $D$, create an instance of $DRAYAGE(O, D, R)$
15: Solve $DRAYAGE(O, D, R)$ with an IP solver
16: OUTPUT: An optimal set of route assignments
```

2.3.2 Generating Route Assignments

In column generation, we seek route assignments that not only are feasible, but also have negative reduced cost. Route assignment generation is implemented based on the enumeration scheme described in Section 2.3.1. The search is carried out separately for each driver group. When the number of new columns for a driver group reaches an upper limit, we terminate the search for that particular driver group and start the search for the next driver group. When there are no more driver groups left to search, all new columns are added to the LP formulation and the next column generation cycle begins.

When the upper limit on the number of new columns is not binding, the column generation algorithm as described above may take as long as enumeration. To avoid long running times, we use a labeling scheme for additional pruning. We keep a list of non-dominated labels, denoted $\mathcal{L}_i$, for each order i. Initially, $\mathcal{L}_i = \emptyset, \forall i \in O$. As we traverse the search tree, we add a new label l_r to $\mathcal{L}_i$ for each feasible route assignment r, where i is the last order covered by r. Each label l_r keeps the earliest completion time, the reduced cost, and the minimum duration for route r. We say l_r dominates l_q if l_r is less than or equal to l_q componentwise. If l_r is dominated by a label in $\mathcal{L}_i$, then we prune the search tree at the node corresponding to r. Otherwise, we add l_r to $\mathcal{L}_i$, remove dominated labels from $\mathcal{L}_i$ and continue traversing the search tree.

The labeling scheme as described above is inexact. An exact labeling scheme needs to take into account the set of orders covered since we are solving a constrained "elementary" shortest path problem (i.e., constrained shortest path problem with no cycles allowed). A survey of constrained shortest path problems and methodology is given in [14]. We used the column generation-based method with labeling to obtain solutions. Since the labeling scheme may miss some route assignments which have negative reduced cost, we calculated LP relaxation values separately in order to get valid lower bounds for our instances. To solve the LP relaxation to optimality, we started by using the column generation scheme with labeling. When no route assignments with negative reduced cost were found, we switched to exact column generation by turning off the labeling scheme.

In Algorithm 3, the changes to dfs are highlighted in bold letters. In addition to the set of parameters passed to Algorithm 1, dual variables (π), and label sets ($\mathcal{L}^j$ for driver group j) are needed in Algorithm 3. The logic for the overall generation-based method is outlined in Algorithm 4. The set of locations and the limit on the number of columns per cycle are denoted by L and M, respectively, in Algorithm 4. The output of Algorithm 4 is an optimal set of route assignments.

Algorithm 3 $dfs(S, F^j, P, R^j, w, \pi, \mathcal{L}^j)$

```
1: for $a \in O$ do
2:   The steps from 3 to 31 of Algorithm 1 are repeated here
3:   if Going back to park location is not feasible then
4:     Go to line 1
5:   else
6:     $\bar{r} \leftarrow r'$, add the final parking stop to $\bar{r}$
7:     Calculate the reduced cost for $\bar{r}$ with $\pi$
8:     Create a new label $l$ for order $a$ using $\bar{r}$
9:     if There is a label in $\mathcal{L}_a^j$ that dominates $l$ then
10:      Go to line 1
11:    else
12:      Remove the labels in $\mathcal{L}_a^j$ that are dominated by $l$
13:    end if
14:    if $\bar{r}$ is not dominated by a route assignment in $R^j$ then
15:      if $\bar{r}$ dominates a route assignment $q$ in $R^j$ then
16:        $R^j = R^j \setminus \{q\}$
17:      end if
18:      if The reduced cost of $\bar{r}$ is negative and $|R^j| \leq K - 1$ then
19:        $R^j \leftarrow R^j \cup \{\bar{r}\}$
20:      end if
21:    end if
22:  end if
23:  Recursively call $dfs(s', F^j, P, R^j, w', r', \pi, \mathcal{L}^j)$
24: end for
```

Algorithm 4 *generation*(O, D, L, M)

```
1: $R \leftarrow \emptyset$
2: for $c, d \in L$ do
3:    Initialize $P_{cd}$
4: end for
5: for $j \in D$ do
6:    for $a, b \in O$ do
7:       Initialize $F^j_{ab}$
8:    end for
9: end for
10: Given $R$, $O$ and $D$, create an instance of $DRAYAGE(O, D, R)$
11: repeat
12:    Solve the LP relaxation of $DRAYAGE(O, D, R)$
13:    Save dual values to $\pi$
14:    $S \leftarrow \emptyset$, $R_{new} \leftarrow \emptyset$
15:    for  $j \in D$ do
16:       $R^j \leftarrow \emptyset$, $\mathcal{L}^j \leftarrow \emptyset$,
17:       Call $dfs(S, F^j, P, R^j, \pi, \mathcal{L}^j, \frac{M}{|D|})$
18:       $R_{new} \leftarrow R_{new} \cup R^j_{new}$
19:    end for
20:    Merge $R_{new}$ into $R$ and let $Q$ be the set of routes assignments in $R$ that become dominated
21:    Add $R_{new}$ to the LP, remove $Q$ from the LP, and resolve the relaxation of $DRAYAGE(O, D, R)$
22: until  $R_{new} = \emptyset$
23: Enforce integrality constraints and solve $DRAYAGE(O, D, R)$
```

2.4 Numerical Results

In this section, we refer to the enumeration-based method as `Enum`, and the column generation-based method with labeling as `Colgen`. We used our methods on data provided by Schneider National Inc., which is the largest truckload carrier in North America. Schneider has defined truck/rail regions, and manages daily truck/rail operations separately for each region. The company dray drivers work exclusively in one region. Each region may have multiple rail ramps.

We tested `Enum` and `Colgen` on a weekly data set from the St. Louis area. We used CPLEX 9.0 callable libraries to solve the LP relaxation and the IP formulation [21]. Since the number of orders was small (ranging from 5 to 28 per day), both `Colgen` and `Enum` were included in the test. `Colgen`, which may produce suboptimal solutions, gave the same solution as `Enum` for all seven days. So, we can conclude that at least for small problems, doing column generation at the root node suffices.

The running times on a Sun-Fire-280R workstation for the two methods are given in Table 5 . IP^* is the optimal solution value found by both methods. The "Cycles" column reports the number of times column generation is called with new dual values in `Colgen`. The "Rows" column reports the number of rows in the formulation of each instance. R_0 denotes the initial set of variables for `Colgen`. R' is the final set of columns for `Colgen`. $\widehat{R}$ is the set of all non-dominated feasible route assignments found by `Enum`. The "Gen" and "Enum" columns respectively report running times in seconds for the `Colgen` and `Enum`.

Table 5: Running times for St. Louis, $|D| = 7$

Day	$\|O\|$	IP^*	Cycles	Rows	$\|R_0\|$	$\|R'\|$	$\|\widehat{R}\|$	Gen	Enum
1	28	4098.90	5	37	161	2474	6823	1.49	9.95
2	24	2538.24	4	31	113	732	1181	0.24	0.37
3	14	454.83	3	21	69	292	452	0.13	0.15
4	11	338.56	2	18	60	908	1614	0.67	3.49
5	11	986.69	2	21	74	76	79	0.03	0.05
6	8	3183.91	1	12	20	20	25	0.05	0.06
7	5	742.98	2	11	22	24	26	0.04	0.04

We also ran `Colgen` for monthly data from the Northern California region. The results are summarized in Table 6. LP^* denotes the optimal LP relaxation value we obtained as described in Section 2.3.2. IP' is the optimal solution value when integrality is enforced. IP' gives an upper bound on the objective value because column generation is inexact and is performed only at the root node of the branch-and-bound tree. "Gap" refers to the gap percentage between LP^* and IP'. The running times for solving IP' were obtained on a Sun-Fire-280R machine and are given in the last column in seconds.

The Northern California region has a much higher volume than St. Louis, with about 60 orders per day on average. The number of rail ramps in the region is 3. About 24% of the orders have live delivery stops (DL_W) at their destinations and about 22% of the orders have live pickup stops (PL_W) at their origins. About 54% of the orders are inbound, and the rest of the orders are outbound. We grouped the 30 company drivers who work in

Table 6: The column generation-based method results for Northern California, $|D| = 17$

Day	$\lvert O\rvert$	LP^*	IP'	Gap	Rows	$\lvert R_0\rvert$	$\lvert R'\rvert$	Time
1	81	3038.40	3041.54	0.10%	100	1046	62926	76.02
2	84	2651.30	2652.03	0.03%	105	1053	68593	34.41
3	10	0.00	0.00	0.00%	29	162	162	0.46
4	9	0.00	0.00	0.00%	27	117	117	0.41
5	96	4851.98	4854.59	0.05%	118	1230	158026	605.75
6	81	3521.72	3521.72	0.00%	100	986	55416	24.39
7	72	4717.73	4732.02	0.30%	92	832	16510	7.02
8	66	874.60	874.60	0.00%	88	782	41183	16.67
9	63	369.75	369.75	0.00%	86	817	21513	8.19
10	11	85.50	85.50	0.00%	29	162	162	0.41
11	12	75.75	75.75	0.00%	30	159	159	0.42
12	105	7113.28	7113.48	0.00%	128	1271	87918	103.63
13	94	5185.72	5237.31	0.99%	117	1085	79976	99.51
14	77	2519.73	2526.38	0.26%	96	939	68414	63.51
15	79	3061.97	3064.90	0.10%	99	887	25022	11.56
16	71	1744.18	1744.18	0.00%	96	904	69237	55.91
17	8	22.50	22.50	0.00%	28	120	120	0.41
18	16	59.25	59.25	0.00%	35	200	200	0.43
19	130	12301.74	12301.74	0.00%	152	1517	115362	125.43
20	71	1556.84	1557.59	0.05%	91	878	37452	14.09
21	70	2670.33	2670.33	0.00%	94	899	27717	11.02
22	92	8325.28	8325.28	0.00%	113	1086	8950	2.52
23	87	5819.89	5819.97	0.00%	106	965	40485	19.90
24	11	135.75	135.75	0.00%	30	112	112	0.36
25	6	24.75	24.75	0.00%	24	66	66	0.38
26	11	45.75	45.75	0.00%	29	171	171	0.44
27	117	9649.70	9677.78	0.00%	139	1385	108498	118.62
28	98	6273.63	6273.63	0.00%	118	1199	65040	41.40
29	98	6183.08	6188.38	0.09%	116	1142	58073	45.41
30	70	2886.02	2890.69	0.16%	92	819	14583	8.74
31	19	10.50	10.50	0.00%	37	253	253	0.40

the region into 16 driver groups, and all third party drivers into a single driver group. We constructed R_0 by adding all feasible route assignments which cover single orders. Since all feasible route assignments pairing an order with a third party driver are feasible, and the number of available third party drivers exceeds the number of orders on any day, the initial set of columns was sufficiently large to guarantee a feasible solution for each instance.

All instances were solved in three minutes or less, with the exception of Day 5, which required about 10 minutes to solve. In Table 7, instances are grouped by the number of column generation cycles in `Colgen`. In a column generation cycle, the number of columns added for each driver group was limited by $\frac{10^5}{|D|}$.

Table 7: Number of column generation cycles for Northern California instances

Cycles	Days
1	3, 4, 10, 11, 17, 18, 24, 25, 26, 31
4	1, 7
5	2, 6, 8, 9, 14, 19, 20, 21, 22, 23, 29
6	12, 13, 15, 16, 27, 28, 30
7	5

For most of the instances, the integer solution was obtained at the root node. Day 5 required the most branch-and-bound nodes (549) followed by Day 1 (147), Day 30 (84), Day 29 (75), Day 14 (74) and Day 23 (20). The gap between the upper and lower bounds is very small, and zero in many cases (i.e., the solution is optimal). Although `Colgen` uses an inexact labeling scheme and only generates columns at the root node, it gives good quality solutions in terms of cost.

2.5 Implementation Challenges

In this section, we briefly discuss some of the challenges in implementing a real-time decision support system for drayage dispatchers. With the advances in communication technologies and geographic information systems, dispatchers can now make operational decisions using computerized order and driver information, which is updated throughout the day. The availability of real-time data creates the potential for a decision support system for dispatchers. A decision support system can evaluate many routing and scheduling options in real-time and suggest feasible and cost-effective options to dispatchers.

Dispatchers work in operating centers and manage dray drivers individually. Drayage drivers and dispatchers can communicate in real time with the help of mobile data terminals in vehicles, which send and receive information from the servers in operating centers. The driver can inform the dispatcher about events such as delivery completions, delays at customer locations or equipment failures. The dispatcher can access the information using a computer terminal at the operating center, and can send information back to the driver regarding the next order.

Dispatchers need to stay ahead of the drivers' progress so that drivers always know their next tasks. In drayage operations, staying ahead of a driver typically means having a plan for what the driver is going to do in the next three or four hours. Dispatchers may modify order-driver assignments and driver schedules in order to handle operational uncertainties. For instance, when a driver is delayed at a customer location, the dispatcher may assign the driver's next planned task to another available driver.

A real-time decision support system for dispatchers should

- collect reliable and up-to-date data,
- generate good solutions in real-time and
- have a powerful user interface.

The accuracy of driving time and loading/unloading time estimates is crucial in determining the feasibility of a route. A driving time estimator behind the decision support system can provide estimates that depend on the day of the week, or the time of the day for better accuracy. The loading/unloading durations are more likely to vary than driving times. In order to improve accuracy, the driver can communicate a new estimate for the stop duration to the dispatcher after arriving at a pickup or a delivery location. Then, the dispatcher can input the new estimate into the decision support system to get an updated solution. The system should also keep track of the history of drivers over a sufficient number of days, so that the planned daily schedules satisfy hours-of-service restrictions. Driving time estimates, driver histories and order attributes are likely to be kept in different ways and formats, so an additional challenge is combining different data sources in a coherent way. The data used by the decision support system and the dispatcher must be synchronized with the data source. Synchronization is especially important for the order data, which can change at any time.

Besides reliable and up-to-date information, the response time and the flexibility of the user interface are major factors in usability. The user interface should be flexible and provide the dispatcher with multiple solutions and allow the dispatcher to change the input or customize the suggested solution. For instance, the dispatcher should be able to override estimated time of availability for drivers. The user interface should allow the dispatcher to specify preferences or restrictions on the routes, so that the decision support system suggests solutions that are more acceptable for the dispatcher. The presentation of suggested solutions should include the right level of detail in order to facilitate the comparison and evaluation of different solutions, and should avoid overwhelming the decision maker with too much information. The user interface should allow the dispatcher to edit the suggested solution, and to search for an alternative solution. The dispatcher may prefer to insert additional waiting time to account for potential delay at a future stop, may change the trailer pools used for an empty trailer relocation, or may desire a solution that does not assign an order to a particular driver. Another possibility is that the dispatcher may input some routes manually into the system and ask for a recommendation for covering the remaining orders with available drivers.

The decision support system should respond to a query within seconds to be used in real-time. Given the need for a flexible interface, the optimization routines used behind the decision support system should be able handle different changes to the data and provide good solutions in a very short time. Parallel computation may be useful in generating multiple solutions concurrently.

2.6 Conclusions

We have shown that a column generation-based approach succeeds in modeling and solving the daily drayage problem, which is a complex real-world problem. We were able to improve the running time of our method through very simple but effective techniques such as the use of precedence feasibility matrices.

We can also use our model to get the next best plan by adding a cut to the formulation of the form $\sum(x_r : r \in R^*) \leq |R^*| - 1$, where R^* is the set of route assignments in the current optimal solution. This infeasible path constraint and stronger versions can be found in [1]. Possible future challenges include limiting capacity for trailer pools which will make the current greedy method for choosing extra stop locations suboptimal and will increase the number of non-dominated route assignments substantially.

CHAPTER III

ONLINE DAILY DRAYAGE PROBLEM

3.1 Introduction

Our goal is to gain insights by studying simpler problems before attempting to solve a more general model of the daily drayage problem with uncertainty. Real-world drayage operations include many sources of uncertainty. Customers may call in same day orders. Orders may be canceled. A driver may arrive at a customer location to find out that the trailer is not ready to be picked up yet. Picking up and delivering loads and driving between locations may take longer than expected. In this thesis, we consider only the dynamic addition of new orders.

We discuss two problems derived from daily drayage operations with orders called in on the same day. The first problem is the Online Daily Drayage Problem, which we study in this chapter. The second problem is the Dray Coverage Problem with one customer location which we study in Chapter 4.

In the Online Daily Drayage Problem, there are multiple drivers, and the objective is to minimize the total deadhead mileage while covering all the orders. We do not assume any prior knowledge about future orders and the input data change dynamically in an online fashion. In order to guarantee feasibility, we assume unlimited capacity and allow the addition of new orders which can be covered by an available driver.

3.2 Motivation

In the Online Daily Drayage Problem, our focus is on covering drayage orders with a set of routes that have high overall utilization, where utilization is the percentage of mileage accumulated while moving loaded trailers. By assuming unlimited capacity, we avoid the question of whether an order is going to be covered or not. Since all orders are to be carried out, the total mileage driven with loaded trailers is fixed for all solutions. Although individual drivers may have low utilization, the overall utilization is maximized.

In any snapshot of the system, drivers can be busy or idle at various locations. Since there are multiple drivers with possibly different attributes, the online problem includes not only sequencing and scheduling aspects, but also an assignment aspect. We think that relocation of idle drivers with respect to potential orders and to locations of other drivers is also crucial in creating robust online solutions. By studying the Online Daily Drayage Problem, we hope to gain insights into creating efficient routes when the driver capacity is not a limiting factor.

In Section 3.3, we describe the Online Daily Drayage Problem in detail. We first simplify the daily drayage problem to focus on the dynamic addition of orders and then we introduce the dynamic aspects of the Online Daily Drayage Problem.

3.3 Problem Description

In Section 3.3.1, we describe the Simplified Daily Drayage Problem. The addition of dynamic orders to the Simplified Daily Drayage Problem is described in Section 3.3.2. The feasibility assumptions in Section 3.3.3 complete the description of the Online Daily Drayage Problem. In Section 3.3.4, we give an example of an online instance.

3.3.1 Simplifying the Daily Drayage Problem

In the Simplified Daily Drayage Problem, the core of the problem is the same as the Static Daily Drayage Problem. We are given a set of drayage orders that need to be satisfied by a set of route assignments.

We assume that there is only one rail ramp. Dray orders are pickup and delivery requests which have exactly one stop at the rail ramp and one stop at a customer location. If the pickup is at the rail ramp, we say the order is *inbound*. Otherwise, the order is *outbound*. Stops do not have time windows. Pickup and delivery durations depend only on the location. The pickup stop is always a PL (pickup preloaded trailer) and the delivery stop is always a DL (drop trailer). Therefore, no empty relocation movements are needed in the simplified problem.

All drivers are available at the beginning of the time horizon at the same park location and need to be back at the park location by the end of the time horizon. A route assignment describes a driver's daily shift which starts and ends at the park location. The cost of a route assignment is the total deadhead mileage. We are given the distances between locations and we assume constant speed for all drivers. The goal is to cover all orders while minimizing the total deadhead cost.

We consider a driver *busy* from the time he starts to deadhead to pick up an order to the time the driver delivers the order. We do not allow preemption, that is, a busy driver can become available only after delivering his current order.

3.3.2 Dynamic Addition of Orders

When dynamic addition of new orders is allowed, the Simplified Daily Drayage Problem becomes the Online Daily Drayage Problem. In the online problem, we start with an initial set of orders, and new orders may be released at any time until the end of the time horizon. The release times of future orders are unknown. The number of future orders is also unknown. We only learn the details of an order when that particular order is released.

3.3.3 Guaranteeing Feasibility

In order to guarantee feasibility, we assume that the number of available drivers is unlimited, and the release time for any order is early enough such that a fresh driver can start from the park location when the order is released, carry out the order and return to the park location by time T.

3.3.4 A Dynamic Example

A simple example of an online instance is given in Figure 5. The rail ramp is the square, the park location is the triangle and customer locations are the circles. The arrows denote orders. The figure includes two snapshots of the system at times 0 and t.

At time 0, we only know about orders 1 and 2. Drivers are idle at the park location. The tentative route assignment for a driver is to carry out order 1 and then order 2 before

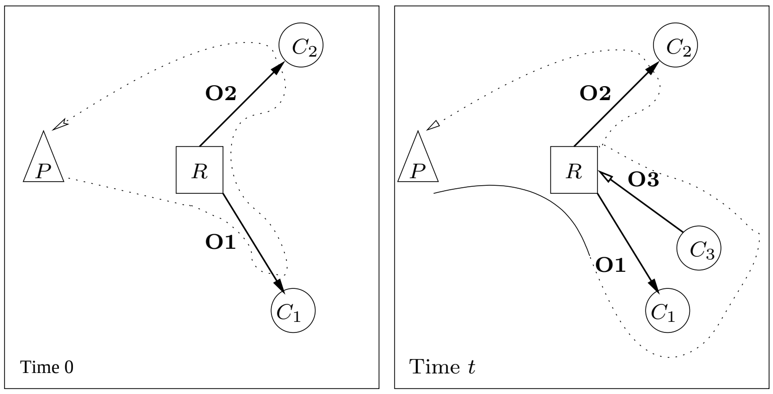


Figure 5: Order 3 is dynamically added to the problem.

returning to the park location. The tentative stop sequence can be written as

$$P \to R(\text{PL } O_1) \to C_1(\text{DL } O_1) \to R(\text{PL } O_2) \to C_2(\text{DL } O_2) \to P.$$

At time t, order 3 is released, and we learn about the customer location C_3 along with the pickup and delivery information for order 3. The driver is busy at time t moving order 1. We modify the remainder of the tentative route assignment to cover order 3 after the delivery of order 1. The new sequence of stops taking place after time t can be written as

$$C_1(\text{DL } O_1) \to C_3(\text{PL } O_3) \to R(\text{DL } O_3) \to R(\text{PL } O_2) \to C_2(\text{DL } O_2) \to P.$$

The modified route assignment is carried out unless a new order is released.

3.4 Reoptimization as a Solution Methodology

Since the cost only depends on the deadhead mileage, the cost of a route assignment is the same for all feasible schedules when the sequence of orders on the route assignment is fixed.

We experimented with an online solution methodology in which the Online Daily Drayage Problem is repeatedly solved to optimality over the set of uncovered orders without regard to the dynamic nature of the problem. Whenever a new order is released, the current status of drivers is updated. Idle drivers make up the currently available capacity and busy drivers are included as future capacity. Since any feasible scheduling of the routing solution has the same cost value, we experimented with different scheduling policies to test their impact on the online solution quality. The steps of a solution methodology with scheduling policy π are summarized in Algorithm 5. Algorithm 5 returns a plan for the remainder of the time horizon and covers all known orders. Every time a new order is released, we call Algorithm 5 to update the solution.

Algorithm 5 Online Solution Generator

```
1: INPUT: Updated set of drivers and orders, scheduling policy $\pi$.
2: Route assignment selection: Find an optimal solution which covers known orders.
3: Scheduling: Schedule each route assignment in the optimal solution according to policy $\pi$.
4: OUTPUT: Return the routing solution and the schedules.
```

In Section 3.4.1, we describe how route assignments are selected in step 2 of Algorithm 5. In Section 3.4.2, we explain the scheduling policies used in step 3 of Algorithm 5.

3.4.1 Optimizing Snapshots

Whenever a new order is released, we take a snapshot of the system and find a new optimal routing solution so that all known orders are covered. Drivers who finished their daily shift and parked at the park location are taken out of the problem. Busy drivers are considered to be available in the future after the completion of their deliveries.

Let t denote the time of the snapshot. We modify the model in Section 2.2.6, since we have to pay special attention to drivers away from their park location at time t. In constructing a plan for the remainder of the time horizon, we must make sure that all drivers return to their park locations by time T, even if they are not assigned new orders. As in the Static Daily Drayage Problem, drivers with the same attributes are grouped together and each route assignment is constructed for a specific driver group. In the model, B denotes the set of driver groups with drivers who are away from their park locations at time t. In other words, the drivers who started their shift before t are in a driver group in B. Let h be the group of drivers who are at the park location and did not start their shifts by t. The set of all driver groups, denoted by D, is given by $D = \{h\} \cup B$. The set of orders not picked up by time t is denoted by O. As in Section 2.2, R_i denotes the set of route assignments which cover order i. The set of feasible route assignments for driver group j which cover at least one order is denoted by R^j, where $j \in D$. The set of all feasible route assignments which cover at least one order is given by $R = \bigcup_{j \in D} R^j$. For each $b \in B$, q_b denotes the route assignment which has only deadheading from the current location of drivers in group b to the park location. The set of route assignments which do not cover any order is denoted by $Q = \bigcup_{b \in B} \{q_b\}$. The cost function $c : R \cup Q \mapsto \Re$ gives the total deadhead cost for a given route assignment. The variable x_r is 1 if $r \in R$ is selected and 0 otherwise. The variable y_b denotes the number of drivers in some group $b \in B$, who deadhead directly to the park location. The number of drivers in group j is given by d_j for $j \in D$.

$$\min \sum_{j \in D} \sum_{r \in R^j} c(r) x_r + \sum_{b \in B} c(r_b) y_b \tag{5}$$

$$\sum_{r \in R_i} x_r = 1, \; \forall i \in O \tag{6}$$

$$\sum_{r \in R^b} x_r + y_b = d_b, \; \forall b \in B \tag{7}$$

$$\sum_{r \in R^h} x_r \leq d_h, \tag{8}$$

$$x_r \in \{0, 1\}, y_b \text{ integer} \tag{9}$$

The constraints (6) guarantee that all orders in O are covered. The constraints (7) ensure that all drivers away from their park locations complete their shifts by the end of the time horizon. The constraint (8) ensures that the number of drivers who start their shift after time t do not exceed the available number.

In our computational experiments, we solved the model by enumerating $R \cup Q$ and solving the resultant IP with CPLEX. In solving each online instance, the sizes of R and Q get smaller and enumeration becomes easier as t increases.

3.4.2 Scheduling Policies

We experimented with four policies, `Earliest`, `Latest`, `Weighted`, and `Deadhead`. Given route assignment r at time t, let n be the number of stops in r. Let t_i^π denote the start time of the activity (loading, unloading, parking) at stop i under scheduling policy π. Let $\mathtt{e}, \ell, \mathtt{w}$, and $\mathtt{d}$ denote respectively `Earliest`, `Latest`, `Weighted`, and `Deadhead` policies. The stop duration at stop i is given by γ_i, $i = 1, \dots, n$ and the transit time from stop i to stop $i+1$ is given by δ_i, $i = 1, \dots, n-1$.

- `Earliest` policy schedules the stops as early as possible.

$$t_1^{\mathtt{e}} = t,\ t_2^{\mathtt{e}} = t_1^{\mathtt{e}} + \delta_1 + \gamma_1, \dots, t_n^{\mathtt{e}} = t_1^{\mathtt{e}} + \delta_{n-1} + \gamma_{n-1}.$$

- `Latest` policy schedules the stops as late as possible.

$$t_n^{\ell} = T - \gamma_n,\ t_{n-1}^{\ell} = t_n^{\ell} - \delta_{n-1} - \gamma_{n-1}, \dots, t_1^{\ell} = t_2^{\ell} - \delta_1 - \gamma_1.$$

- `Weighted` policy schedules the next stop to be between the earliest and the latest feasible times according to parameter $\theta \in (0,1)$ if the next stop is not a delivery. The driver does not wait between the pickup and the delivery of an order. For $i = 1, \dots, n$, let $\theta_i = \theta$ if stop i is a pickup or is parking at end of the day and let $\theta_i = 1$ if stop i is a delivery.

$$\begin{aligned} t_1^{\mathtt{w}} &= \theta_1 t + (1-\theta_1) t_1^{\ell},\ t_2^{\mathtt{w}} = \theta_2 (t_1^{\mathtt{w}} + \delta_1 + \gamma_1) + (1-\theta_2) t_2^{\ell}, \dots, \\ t_n^{\mathtt{w}} &= \theta_{n-1} (t_{n-1}^{\mathtt{w}} + \delta_{n-1} + \gamma_{n-1}) + (1-\theta_{n-1}) t_n^{\ell}. \end{aligned}$$

- `Deadhead` policy uses the earliest schedule for the activities preceding the longest deadhead and uses the latest schedule for the remainder of the route assignment.

$$i^* = \operatorname*{argmin}_{\substack{i=1,\dots,n-1 \\ i+1 \text{ is a pickup}}} \delta_i, \quad t_i^{\mathtt{d}} = t_i^{\mathtt{e}} \text{ for } 1 \le i \le i^* - 1, t_i^{\mathtt{d}} = t_i^{\ell} \text{ for } i^* \le i \le n.$$

In Figure 3.4.2, a sample route is scheduled with the four different policies. Shaded boxes denote the time spent deadheading. Boxes with order names denote the time spent from the pickup until the completion of the order. At time t, the driver is at the rail ramp. The route assignment covers order 1, order 2 and then order 3. In `earliest`, the whole slack in the schedule is at the end. The motivation for the `earliest` policy is to carry out known orders as soon as possible so that more new orders can be covered by the driver before the time horizon constraint becomes active. In `latest`, the whole slack in the schedule is at the beginning. The motivation for the `latest` policy is to delay processing of known orders so that route assignments are made with more information. The policy `weighted`, takes the middle ground and is parameterized by θ. The policy `earliest` is equivalent to `weighted` with $\theta = 1$, and the policy `latest` is equivalent to `weighted` with $\theta = 0$. In the `deadhead` policy, the driver has more information and more flexibility before committing to the most expensive part of his assignment since all the slack in the schedule is placed before the longest deadhead.

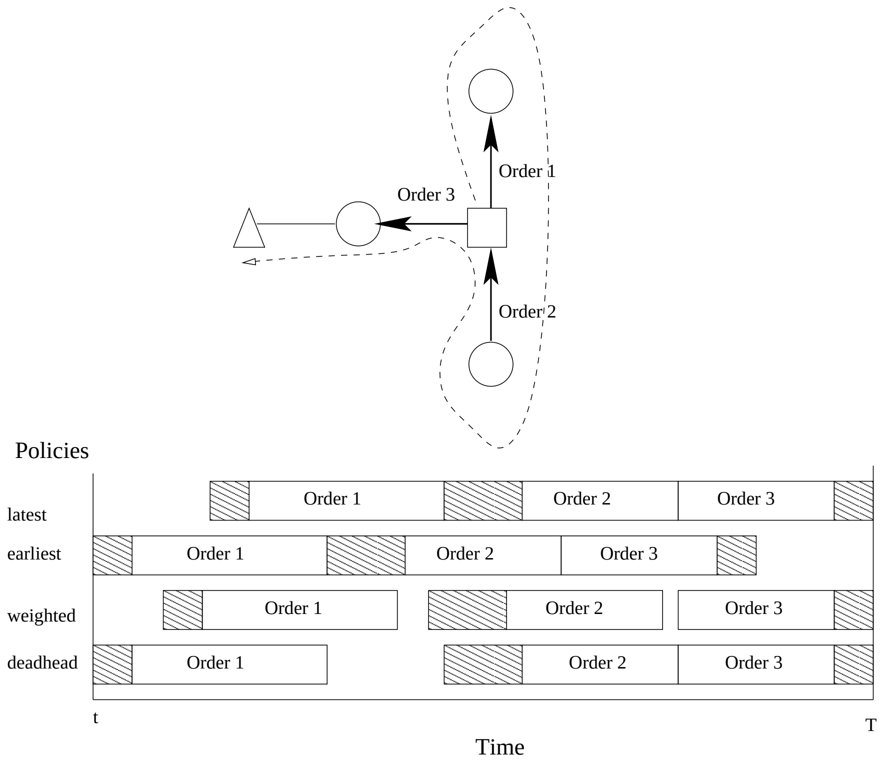


Figure 6: Different scheduling policies for a sample route assignment. Shaded boxes denote deadheading. Boxes with order names denote the time spent starting from the pickup until the completion of the order.

In Section 3.5, we define the perfect information solution and the fair solution. We use the fair solution to evaluate the performance of our solution methodology under different scheduling policies. In Section 3.6, we prove a bound on the performance guarantee of any deterministic online algorithm for the online daily drayage problem. We present numerical results in Section 3.7.

3.5 Competitive Ratio

For each instance of the Online Daily Drayage Problem, one can find a perfect information solution at the end of the time horizon, as if all orders were known in advance. In a perfect information solution, there is no uncertainty, but still no order can be picked up before its release time.

The competitive ratio of an online algorithm is a measure of the worst-case performance of the algorithm compared to the perfect information solution.

Definition 3 (Competitive Algorithm [20])**.** *Let I denote the set of instances for online problem P, and A denote an online deterministic algorithm for P. Let $A(i)$ be the total cost for instance i when solved by algorithm A, and $OPT(i)$ be the total cost for an optimal perfect information solution for instance i. Algorithm A is called* c-competitive *if*

$$A(i) \leq c \cdot OPT(i), \quad \forall i \in I.$$

The competitive ratio *of A is the infimum over all c for which A is c-competitive.*

The perfect information solution can dispatch a driver to pick up an order before the order is released. In our solution methodology, we do not consider tactical relocation of idle drivers to potential pickup locations. Therefore, we introduce the concept of a fair solution. In the fair solution, the problem is again solved as if all the orders were known in advance. However, the start of a driver's deadhead to pick up an order cannot be earlier than the order's release time.

3.6 A General Lower Bound for the Competitive Ratio

In this section, we show that all deterministic competitive online algorithms for the Online Daily Drayage Problem must have a competitive ratio greater than 2.0641. However, we do not know whether a constant competitive ratio exists independent of the data.

Theorem 1. *Any deterministic, competitive and online algorithm for the Online Daily Drayage Problem must have a competitive ratio greater than 2.0641.*

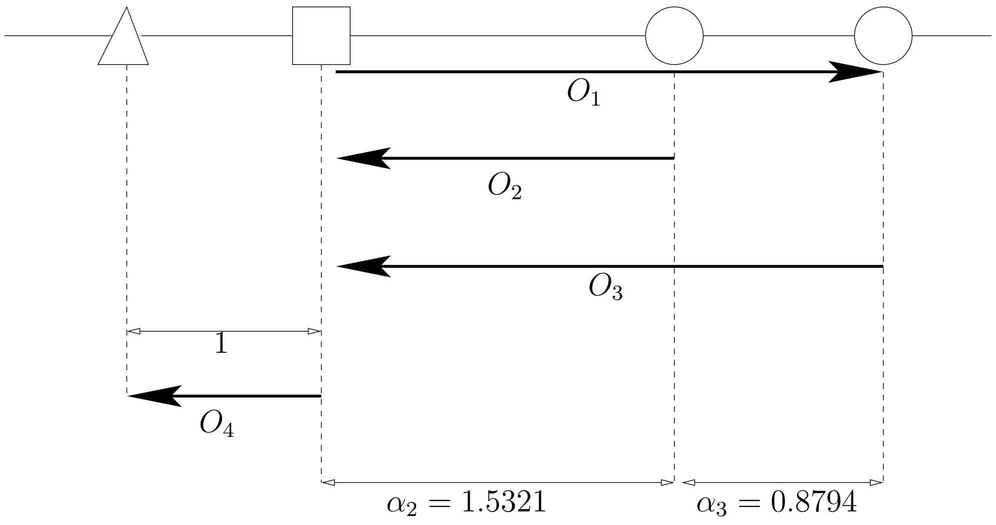
We provide an overview of the proof in Section 3.6.1 and the complete proof in Section 3.6.2.

3.6.1 Overview of the Proof

The proof is inspired by an analogous result in [2]. We create two online problem instances, `Ins1` and `Ins2`, such that both instances have the same pair of orders, O_1 and O_2, at time 0. We choose O_1 and O_2 such that in any solution given by a r-competitive deterministic algorithm, a single driver moves O_1 and then O_2, where $r = 2.0641$. If the algorithm chooses to move O_2 "early", then we show that the online algorithm has made an early commitment in the case of `Ins1`. Otherwise, we show that the online algorithm is too "late" and is forced to use two drivers to cover all of the orders in `Ins2`. In either case, there is an instance for which the algorithm is not r-competitive.

3.6.2 Proof

The orders of the online instances `Ins1` and `Ins2` are in Figure 7. In Figure 7, the rail ramp is the square, the park location is the triangle and customer locations are the circles. The arrows denote orders. `Ins1` has orders $\{O_1, O_2, O_3\}$ where O_3 is released at T_1. `Ins2` has orders $\{O_1, O_2, O_4\}$ where O_4 is released at T_2. Only orders O_1 and O_2 are known in advance. Total stop duration for order i is denoted by d_i. The time horizon is T.



$$d_i = 1,\ \forall i = 1, 2, 3,\ d_4 = 2, \quad T = 13,\ T_1 = 8.4679,\ T_2 = 10$$

Figure 7: Orders and locations for `Ins1` and `Ins2`.

Suppose that we have an r-competitive online algorithm `ALG`. We first prove that `ALG` initially chooses to dispatch a single driver to move O_1 and then O_2. Suppose that no new orders are released. Then, there are three solutions which are listed in Table 8. Since the speed is constant, we use deadheading time instead of deadheading distance in cost calculations. The route $[1, 2]$ is the optimal offline solution. Any r-competitive online

Table 8: Solutions covering orders O_1 and O_2

Solution	Cost	Completion Times
$[1, 2]$	$2 + \alpha_3 = 2.8794$	$2 + 2\alpha_2 + 2\alpha_3 + d_1 + d_2$ $= 8.823 \leq T$
$[2, 1]$	$2 + 2\alpha_2 + \alpha_3 = 5.9436$	$2 + 4\alpha_2 + 2\alpha_3 + d_1 + d_2$ $= 11.8872 \leq T$
$[1], [2]$	$(2 + \alpha_2 + \alpha_3) + (2 + \alpha_2)$	

algorithm must choose route $[1, 2]$ because

$$\frac{c(1) + c(2)}{c(1, 2)} \geq \frac{c(2, 1)}{c(1, 2)} \geq 2.06418 > r,$$

where $c(s_1, \ldots, s_n)$ is the cost of route $[s_1, \ldots, s_n]$.

Let t denote the scheduled pickup time for O_2 in the initial solution given by `ALG`. The earliest and the latest times for picking up O_2 are 5.2909 and 10, respectively, since

$$5.2909 = 1 + \alpha_2 + 2\alpha_3 + d_1 \leq t \leq T - 1 - \alpha_2 - d_2 = 10.$$

First, we show that the competitive ratio of `ALG` for `Ins1` is greater than r if $t < T_1$. For the case $t \geq T_1$, we show that the competitive ratio of `ALG` for `Ins2` is greater than r. Therefore, we conclude that the competitive ratio of any deterministic online algorithm must be greater than r.

Case 1: $5.2909 \leq t < T_1$ and `Ins1`

Route $[1, 2]$ is in progress, and a new order, O_3, is released at time T_1. Since $T_1 > t$, the driver must have already started to carry out order 2. Possible online solutions are given in Table 9.

The best online solution value is 7.2909 since route $[1, 2, 3]$ is infeasible. The best offline solution is $[2, 1, 3]$ with cost $2 + \alpha_2 = 3.5321$ and completion time $\max\{1 + 3\alpha_2 + \alpha_3 + d_1 + d_2, T_1\} + d_3 + \alpha_2 + \alpha_3 + 1 = 12.8872 \leq T$. We have

$$\frac{c(1, 2) + c(3)}{c(2, 1, 3)} \geq 2.064182 > r,$$

Table 9: Online solutions for case 1

Solution	Cost	Completion Times
$[1,2,3]$	$2+\alpha_2+2\alpha_3$	$2+4\alpha_2+4\alpha_3+d_1+d_2+d_3$ $= 14.646 > T$
$[1,2],[3]$	$(2+\alpha_3)+(2+\alpha_2+\alpha_3)$ $= 7.2909$	$\max\{1+\alpha_2+\alpha_3, T_1\}+$ $\alpha_2+\alpha_3+1+d_3 = 12.8794 \leq T$

which contradicts that ALG is r-competitive for case 1.

Case 2: $10 \geq t \geq T_1$ and Ins2

Route $[1,2]$ is in progress and a new order, O_4, is released at time T_2. The latest pickup time for O_1 is $T-1-2\alpha_2-2\alpha_3-d_1-d_2 = 5.177$. Therefore, by time T_2, the driver must have already picked up order 1, otherwise route $[1,2]$ becomes infeasible. Possible online solutions are given in Table 10. The best online solution value is $c(1,2)+c(4)$. The best

Table 10: Online solutions for case 2

Solution	Cost	Completion Times
$[1,2,4]$	$1+\alpha_3$	$\max\{T_1+\alpha_2+d_2, T_2\}+d_4+1$ $= 14 > T$
$[1,2],[4]$	$(2+\alpha_3)+(1) = 3.8794$	$[1,2]$: feasible since $T_1 \leq t_l$ $[4]: \max\{2, T_2+1\}+d_4 = 13 \leq T$
$[1,4,2]$	$2+2\alpha_2+\alpha_3+1$	
$[1,4],[2]$	$(1+\alpha_2+\alpha_3)+(2+\alpha_2)$	
$[1],[2,4]$	$(2+\alpha_2+\alpha_3)+(1+\alpha_2)$	
$[1],[4,2]$	$(2+\alpha_2+\alpha_3)+$ $(1+1+\alpha_2+1)$	

offline solution is $[1,2,4]$ with cost $1+\alpha_3 = 1.8794$, and completion time $\max\{1+2\alpha_2+2\alpha_3+d_1+d_2, T_2\}+d_4+1 = 13 \leq T$. We do not have to consider the completion times for the last four solutions since each has a cost higher than $c(1,2)+c(4)$. We have

$$\frac{c(1,2)+c(4)}{c(1,2,4)} \geq 2.064169 > r,$$

which contradicts that ALG is r-competitive for case 2.

Combining cases 1 and 2, we can conclude that ALG cannot be r-competitive, which completes the proof. The proof is valid even if preemption is permitted while a driver is deadheading to pick up his next order.

We have assigned α_2 and α_3 values that approximately maximize the lower bound. Let

$r_1(\alpha_2, \alpha_3), r_2(\alpha_2, \alpha_3)$, and $r_3(\alpha_2, \alpha_3)$ be defined as

$$\begin{aligned}
r_1(\alpha_2, \alpha_3) =& \frac{c(2,1)}{c(1,2)} = 1 + \frac{2\alpha_2}{2+\alpha_3}, \\
r_2(\alpha_2, \alpha_3) =& \frac{c(1,2)+c(3)}{c(2,1,3)} = 1 + \frac{2+2\alpha_3}{2+\alpha_2}, \\
r_3(\alpha_2, \alpha_3) =& \frac{c(1,2)+c(4)}{c(1,2,4)} = 1 + \frac{2}{1+\alpha_3}.
\end{aligned}$$

These functions of α_2 and α_3 correspond to the ratios we calculate in the proof. The best lower bound we can prove while using the same line of argument and only adjusting the parameters is given by $r^* = \max_{\alpha_2, \alpha_3 \geq 0} \min\{r_1(\alpha_2, \alpha_3), r_2(\alpha_2, \alpha_3), r_3(\alpha_2, \alpha_3)\}$. The maximum can only be achieved if all three ratios are equal. Then, the only positive real root of the polynomial $a^3 + 3a^2 - 3 = 0$ should be chosen as the value of α_3, which is approximately 0.8794. The value for α_2 is chosen to be $(1+\alpha_3)^2 - 2$, which is approximately 1.5321.

3.7 Computational Experiments on Scheduling Policies

We generated 20 instances with 20 orders known in advance. Each instance has a different set of initial orders. The speed of the trucks is assumed to be 60 miles per hour so that miles and minutes can be used interchangeably. For each instance, there is a single rail ramp at the center of a 150×150 mile square region. In each instance, 10 additional orders are added dynamically. The release time of a dynamic order is uniformly distributed over the first 10 hours. Orders are equally likely to be inbound or outbound. The customer stops of orders are generated uniformly over the square region. A randomly generated order may be infeasible, i.e., it cannot be covered by a driver who leaves the park location at the release time of the order, since the attributes of an order are drawn randomly from distributions. An infeasible order is discarded and a replacement is generated until the target number of dynamic orders is achieved. The time horizon is 12 hours. The park location is generated uniformly over the square region with a maximum distance of 30 miles from the rail ramp. The stop duration at the ramp is 30 minutes and the stop duration at a customer location is uniformly distributed between 30 and 210 minutes.

We compared the online solutions given by `Earliest`, `Latest`, `Weighted`, and `Deadhead` with the fair solution. We used Algorithm 6 to carry out the simulation. Let T denote the end of the time horizon, and t denote the simulation clock in Algorithm 6.

Algorithm 6 Simulation for the Online Daily Drayage Problem

```
1: Initialize the status of drivers and orders, let $t \leftarrow 0$.
2: Find a solution for covering all known orders by calling Algorithm 5.
3: New Event: Randomly generate a new order release event; let $t_e \geq t$ denote the release
   time.
4: If $t_e \geq T$, then discard the new order event, let $t_e \leftarrow T$.
5: Execution: Carry out the routing decisions according to the chosen schedule until $t_e$.
6: System Update: Update the status of drivers and orders, $t \leftarrow t_e$.
7: If $t_e = T$, end simulation. Else, go back to step 2.
```

We calculated the fair and perfect information solutions for each instance, and compared them with the online algorithm's solutions. The results are given in Table 11.

Table 11: Percentage deviation of online algorithms compared to fair solution

	`Earliest`	`Latest`	`Weighted` with $\theta = 0.5$	`Deadhead`
Average	17.03%	20.24%	18.96 %	14.36%
Std. Dev.	8.70%	9.62%	10.22%	8.51%
Best	4.72%	4.48%	4.02%	1.70%
Worst	34.30%	39.29%	39.29%	29.74%

There does not seem to be substantial differences among the scheduling policies, although `Deadhead` seems to perform better on average with a smaller variance.

3.8 Adding Anticipation to Reoptimization

In the original methodology, route assignments are selected such that the total cost to cover all known orders is minimized. In the new methodology, we change the cost function used in the route assignment selection to favor route assignments that can potentially cover additional orders. The modified cost of a route assignment is the total cost minus a reward for the slack in the schedule.

The gap length is a parameter which is an estimate for the time needed to carry out an order. For instance, given the schedule in Figure 8, if we take the gap length to be 100 minutes, the number of gaps is given by

$$\left\lfloor \frac{50}{100} \right\rfloor + \left\lfloor \frac{120}{100} \right\rfloor + \left\lfloor \frac{60}{100} \right\rfloor = 0 + 1 + 0 = 1.$$

The reward is linear in the number of gaps.

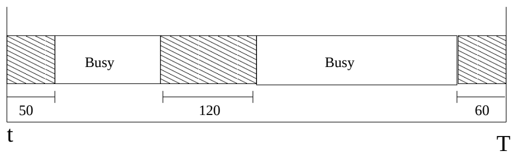


Figure 8: An example schedule for gap calculation.

More formally, let g denote the gap length, and let μ denote the reward per gap. Let $p(r)$ be the scheduling policy used for route assignment r and let n_r denote the number of stops in route assignment r. The start time for the activity in stop i is denoted by $t_i^{p(r)}$ for $i = 1, \ldots, n_r$. Let $t_0^{p(r)} = t$ and $t_{n_r+1}^{p(r)} = T$ and $\delta_0 = \gamma_0 = \gamma_{n_r+1} = 0$ for all r. The number of gaps in the schedule of r, denoted η_r, is given by

$$\eta_r = \sum_{i=0}^{n_r} \left\lfloor \frac{t_{i+1}^{p(r)} - t_i^{p(r)} - \delta_i - \gamma_i}{g} \right\rfloor .$$

We replace $c()$ in (5) with $c'()$ such that $c' : R \cup Q \mapsto \Re$ and $c'(r) = c(r) - \eta_r \mu$.

We experimented with a new scheduling policy denoted `coverage`. In `coverage`, a route assignment which covers at least one gap can be scheduled according to `earliest`, `latest`, or `deadhead` policy. Given a set of route assignments which has at least one gap, `coverage` policy chooses one of the three schedules for each route assignment. The goal is to maximize the duration for which there is at least one idle driver who can potentially cover new orders. For route assignments with no gaps, the default schedule is determined by `deadhead` policy.

We run Algorithm 5 with the modified cost to find a routing solution denoted by R^*. We use one of `earliest`, `latest` and `deadhead` policies as the default scheduling policy in calculating η_r for the modified cost $c'()$. Let R_1^* be the set of route assignments in R^* which has at least one gap. We choose κ and mark maximal number time values denoted by $\tau_1, \tau_2, \ldots, \tau_K$, such that $\tau_1 = t$, $\tau_2 = \tau_1 + \kappa$, $\tau_3 = \tau_2 + \kappa$, $\ldots$, $\tau_K = \tau_{K-1} + \kappa \leq T$, and $\tau_K + \kappa > T$. The coefficient α_{rk}^{π} is 1 if there is a gap in the schedule of route assignment

r under policy π which includes time τ_k, i.e., there exists $i' \in \{0, 1, \ldots, n_r\}$ such that $t^{\pi}_{i'+1} - t^{\pi}_{i'} - \delta_{i'} - \gamma_{i'} \geq g$ and $t_{i'+1} \geq \tau_k \geq t_{i'} + \delta_{i'} + \gamma_{i'}$. The decision variable x^{π}_r is 1 if the route assignment r is scheduled with respect to policy $\pi \in \{\mathtt{e}, \ell, \mathtt{d}\}$. When the model is solved to optimality, the decision variable y_k is 1 only if time t_k is not in a gap of any selected schedule, since the objective is to minimize the sum of all y_k's. The parameter Γ denotes the minimum number of drivers idle at one time to assume that a time mark is covered.

$$\min \sum_{k=1}^{K} y_k \tag{10}$$

$$x^{\mathtt{d}}_r + x^{\mathtt{e}}_r + x^{\ell}_r = 1, \quad \forall r \in R^* \tag{11}$$

$$\sum_{r \in R^*} \left(\alpha^{\mathtt{d}}_{rk} x^{\mathtt{d}}_r + \alpha^{\mathtt{e}}_{rk} x^{\mathtt{e}}_r + \alpha^{\ell}_{rk} x^{\ell}_r \right) + y_k \geq \Gamma, \quad k = 1, \ldots, K \tag{12}$$

$$x^{\mathtt{d}}_r, x^{\mathtt{e}}_r, x^{\ell}_r \in \{0, 1\}, \quad r \in R^* \tag{13}$$

For route assignments with at least one gap, the schedules are selected according to the solution of (10)-(13). The route assignments in $R^* \setminus R^*_1$ are scheduled according to the default policy.

In Section 3.9, we present computational results with the modified cost function and the additional policy.

3.9 Additional Computational Experiments

We created three data sets as in Section 3.7. In the first data set, `Data10`, 20 orders are known in advance and 10 new orders are dynamically released. In the second data set, `Data20`, the same number of orders is known in advance, but the number of dynamic orders is increased to 20. In the last data set, `Data40`, the number of dynamic orders is 40. Each data set has 20 instances. For each instance, we measure the performance of an algorithm with the average percentage deviation of the algorithm's solution value from the value of a fair solution. In all of the figures in Section 3.9, we present the average performance for a dataset instead of reporting individual performances on each instance.

The effect of μ on the performance of various scheduling policies is tested for `Data10`, `Data20` and `Data40` and the results are plotted in Figures 9, 10 and 11. We take g to be 150 minutes in all cases. When $\mu = 0$, the solution methodology is equivalent to the method described in Section 3.4. In the figures, D, E and L stand for `deadhead`, `earliest` and `latest` respectively.

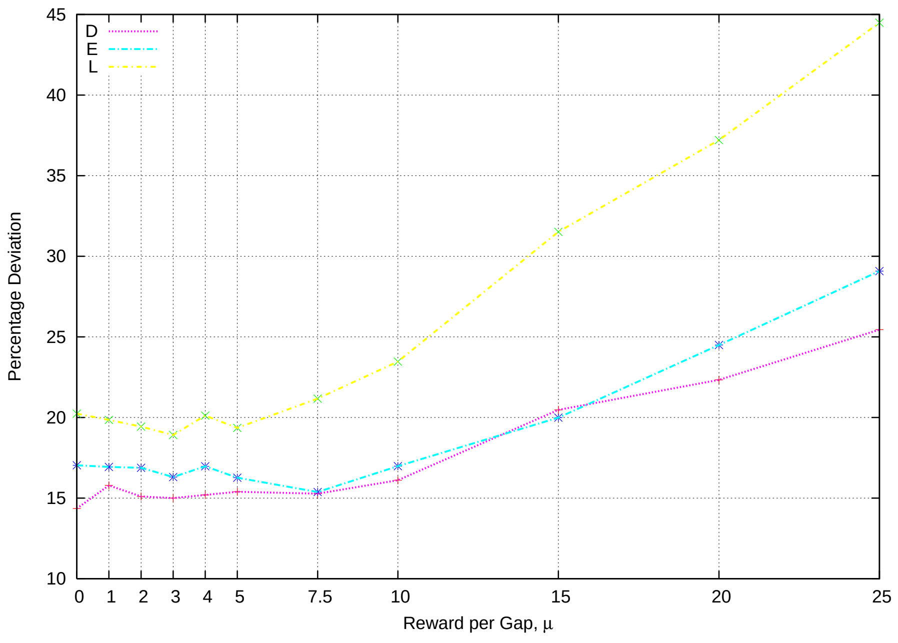


Figure 9: Performance of policies on `Data10`.

`Deadhead` still seems to be the best scheduling policy whereas `latest` seems to perform the worst on average. `Earliest` and `deadhead` policies perform close to each other for `Data10` and `Data20`, and `deadhead` stands out as the best policy for `Data40`. As the degree of dynamism in the data increases, the best μ value tends to increase as well. For `Data10`, increasing μ does not improve performance in general. For `Data20`, the values of μ between 3 and 10 seem to give better results than the original solution methodology with $\mu = 0$ under `earliest` and `deadhead` policies. The best performance for `Data40` seems to be obtained by `deadhead` policy with $10 \leq \mu \leq 15$.

In Figures 12, 13 and 14, we present numerical results for the effect of g on the performance of `deadhead`. For `Data10`, g values between 150 and 300 do not change the performance for values of μ that produce good results. This result is expected since the

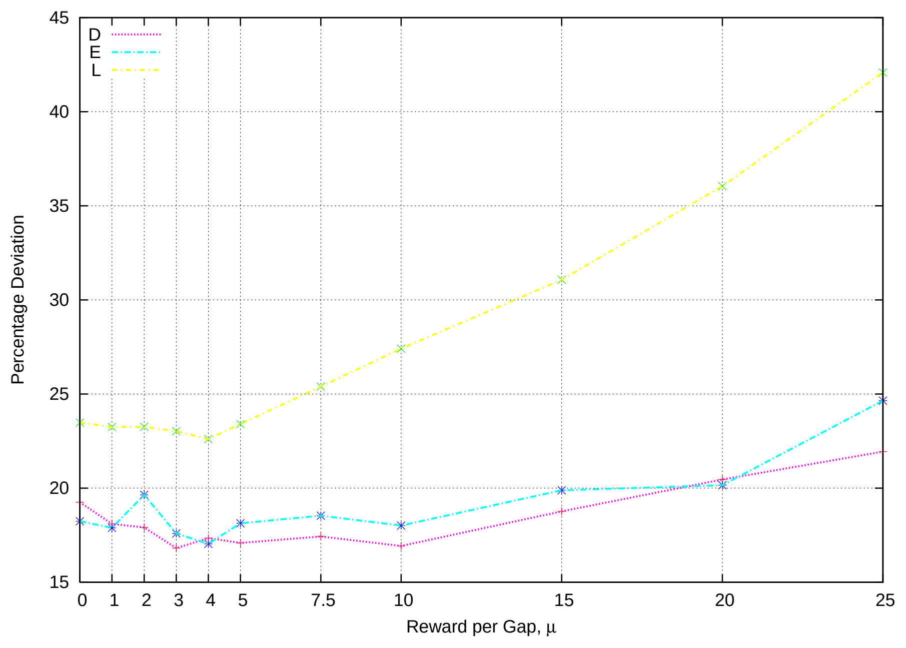


Figure 10: Performance of policies on `Data20`.

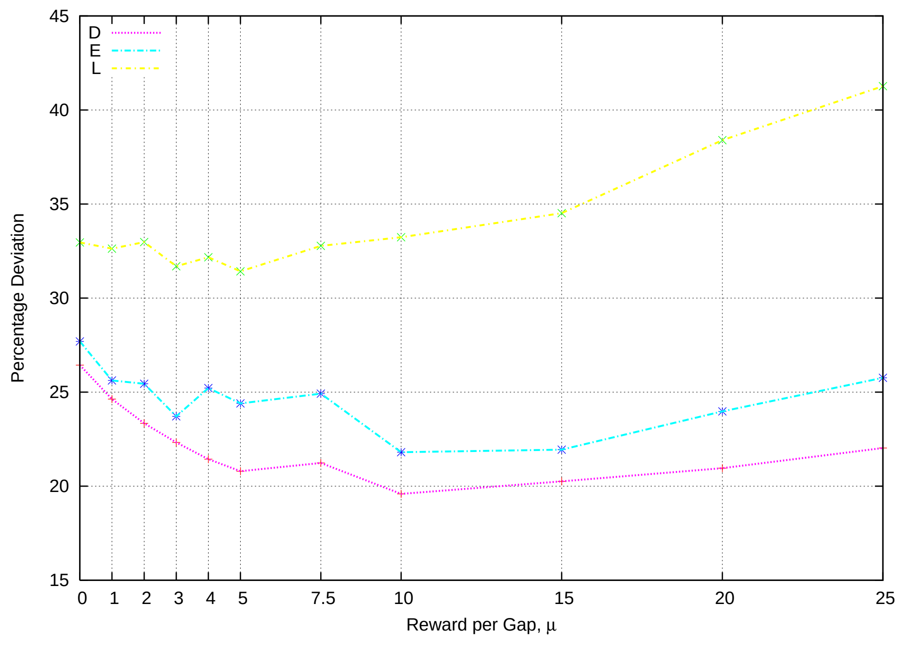


Figure 11: Performance of policies on `Data40`.

performace gain by rewarding gaps is very limited for the case with 10 dynamic orders. For `Data20`, as μ increases, $g = 300$ seems to perform better than our initial selection of $g = 150$. The behavior for `Data20` suggests that rewarding bigger gaps generously rather than giving smaller rewards to more routes that have smaller gaps should be preferred for this data set. The results for `Data40` suggest that there is a trade-off between choosing big gaps with big rewards and small gaps with small rewards for more dynamic data sets. For smaller values of μ, algorithms with smaller values of g perform better and for larger values of μ, algorithms with larger values of g perform better.

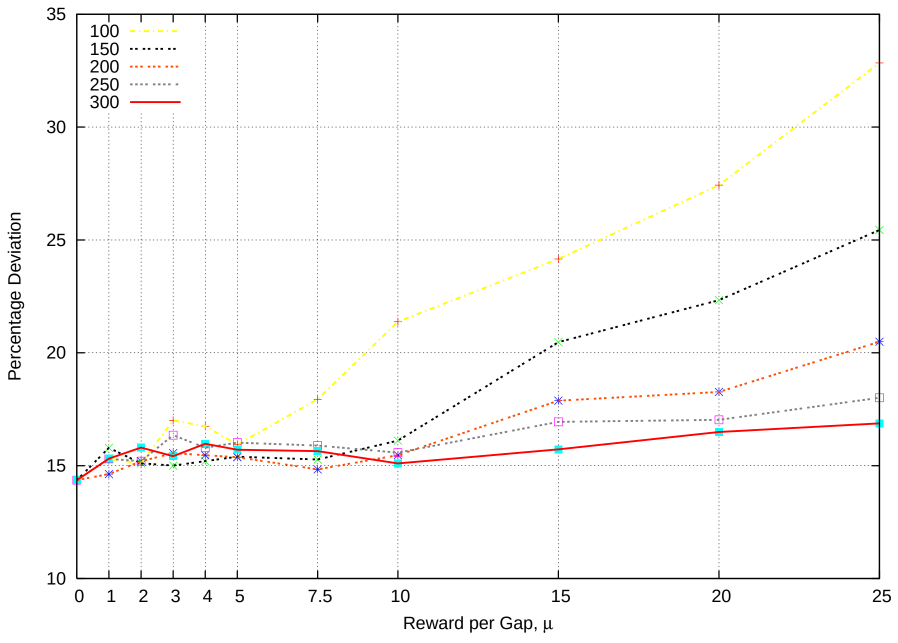


Figure 12: Performance of `Deadhead` on `Data10` for various values of g.

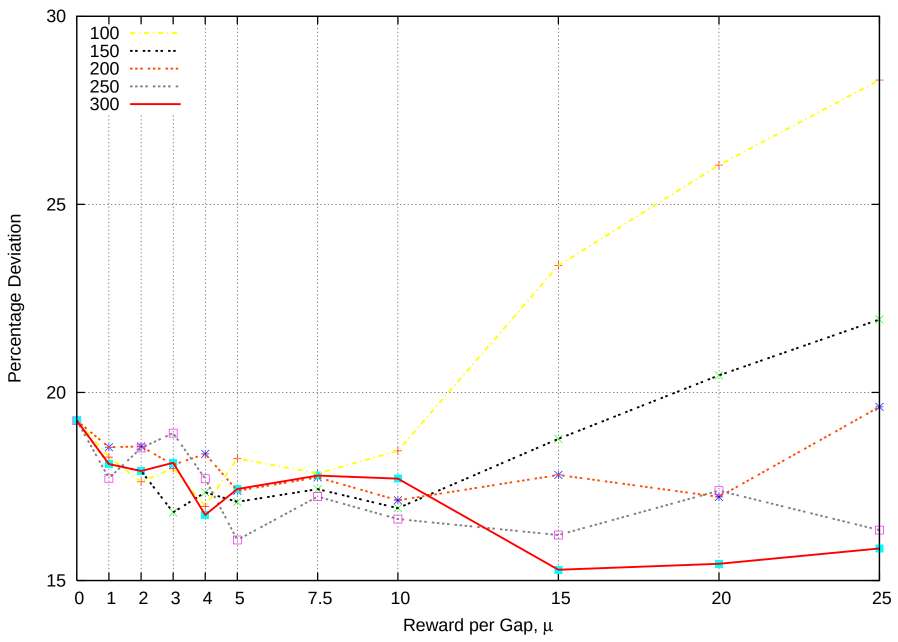


Figure 13: Performance of `Deadhead` on `Data20` for various values of g.

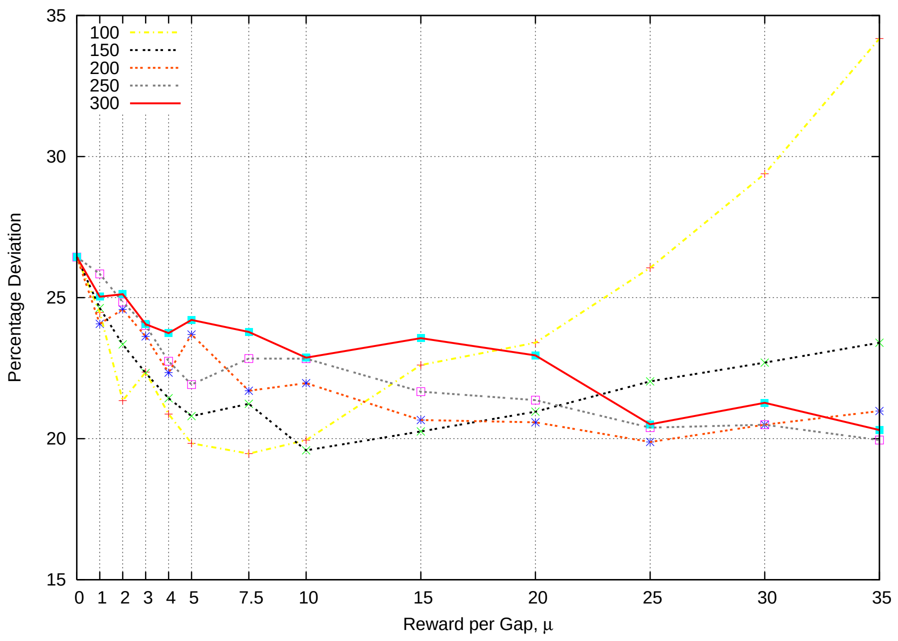


Figure 14: Performance of `Deadhead` on `Data40` for various values of g.

Another way to calculate η_r is to count the idle-time blocks that are large enough. The original method of calculating η_r is finding the total number of gaps that can fit inside idle-time blocks. More precisely,

$$\eta_r = \sum_{i=0}^{n_r} \omega_t^r, \text{ where } \omega_t^r = \begin{cases} 1 & \text{if } t_{i+1}^{p(r)} - t_i^{p(r)} - \delta_i - \gamma_i \geq g, \\ 0 & \text{otherwise.} \end{cases}$$

In Figures 15,16 and 17, we evaluate the performance of the new η_r calculation method by plotting the reduction in the average percentage deviation. The positive values indicate improvement gained by switching to the new η_r formula. The scheduling policy is `deadhead`.

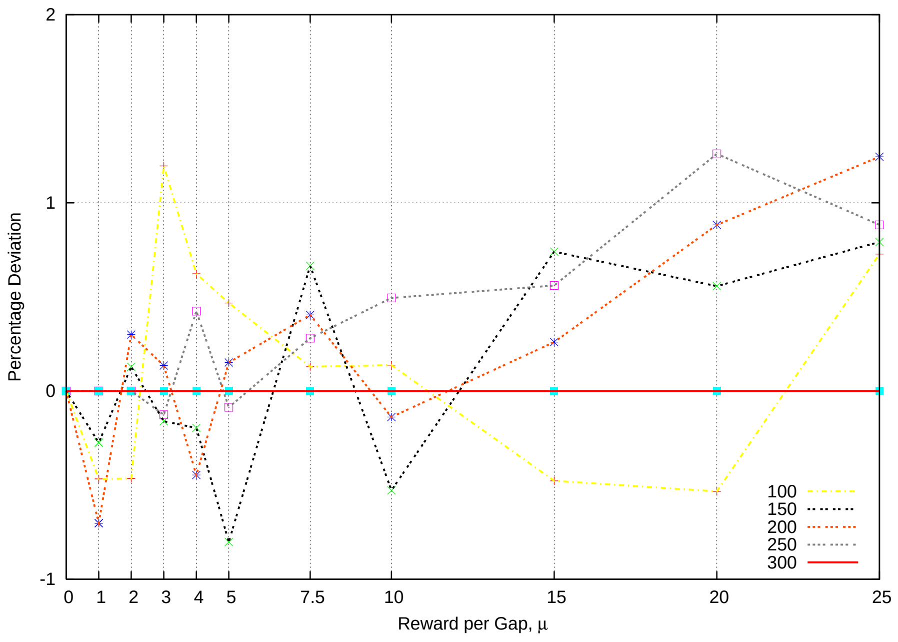


Figure 15: The reduction in average percent deviation for `Data10`.

In most cases, the performance gain is below 1%, and the new formula for η_r does not always improve performace. Among different values of g, the most consistent improvement is obtained for $g = 200$.

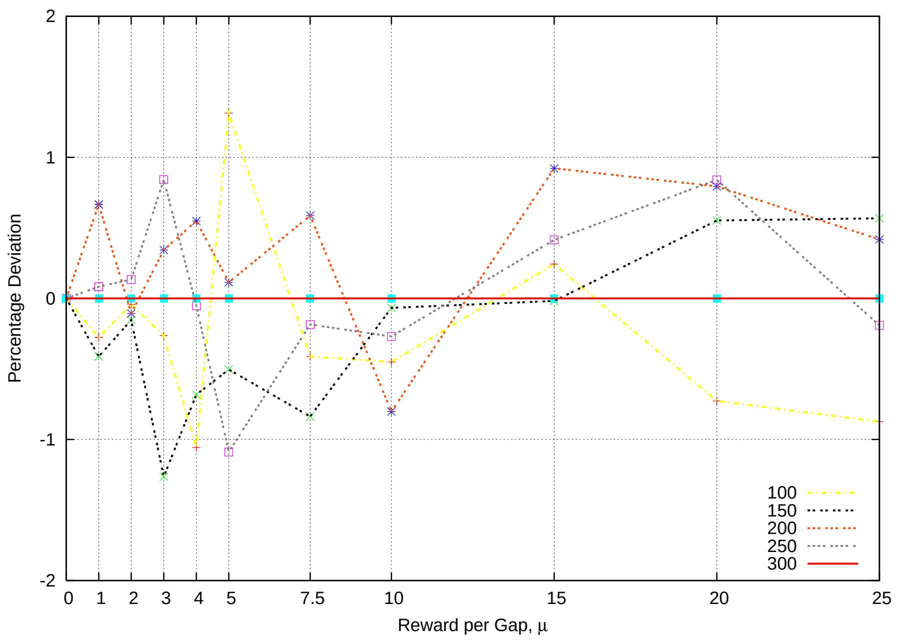


Figure 16: The reduction in average percent deviation for `Data20`.

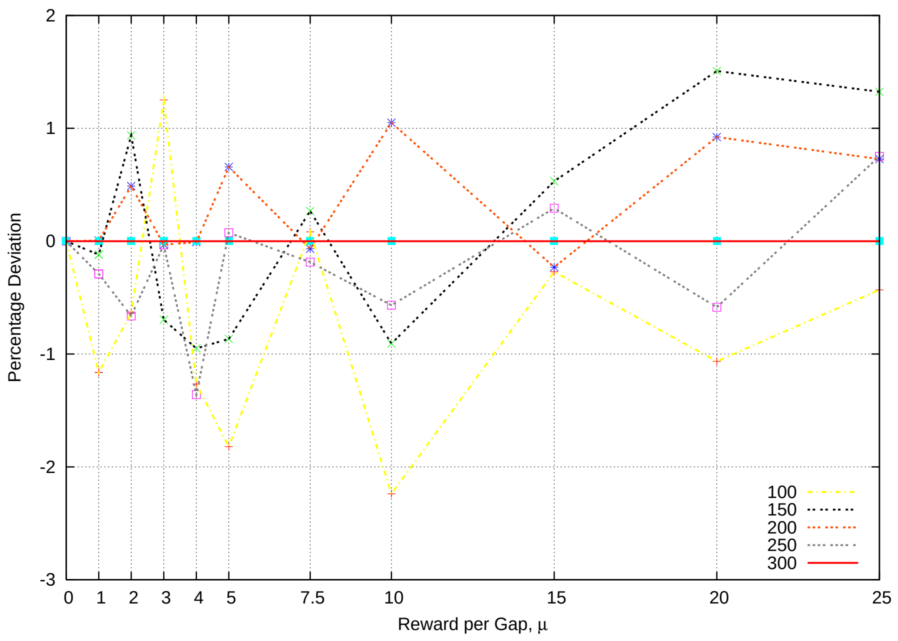


Figure 17: The reduction in average percent deviation for `Data40`.

In Figures 18, 19 and 20, we compare `earliest`, `latest`, `deadhead` to the new policy `coverage`. In `coverage`, the default scheduling policy is in effect before the rescheduling of the selected routes. In the figures, the performance of `coverage` with `earliest`, `latest`, `deadhead` as the default scheduling policy is labeled by CE, CL, and CD respectively. The new η_r formula is used , κ equals 30, Γ is 1 and g is taken to be 200 minutes.

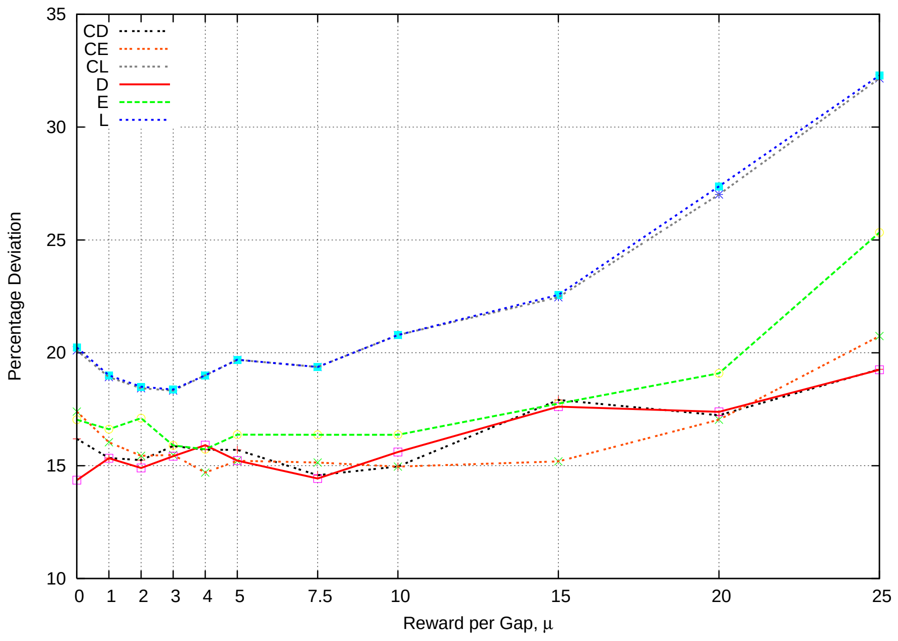


Figure 18: The performance of various policies for `Data10`.

The rescheduling of selected routs seems to improve the solution by a small margin in some cases. The performance of `coverage` is mostly determined by the default scheduling policy.

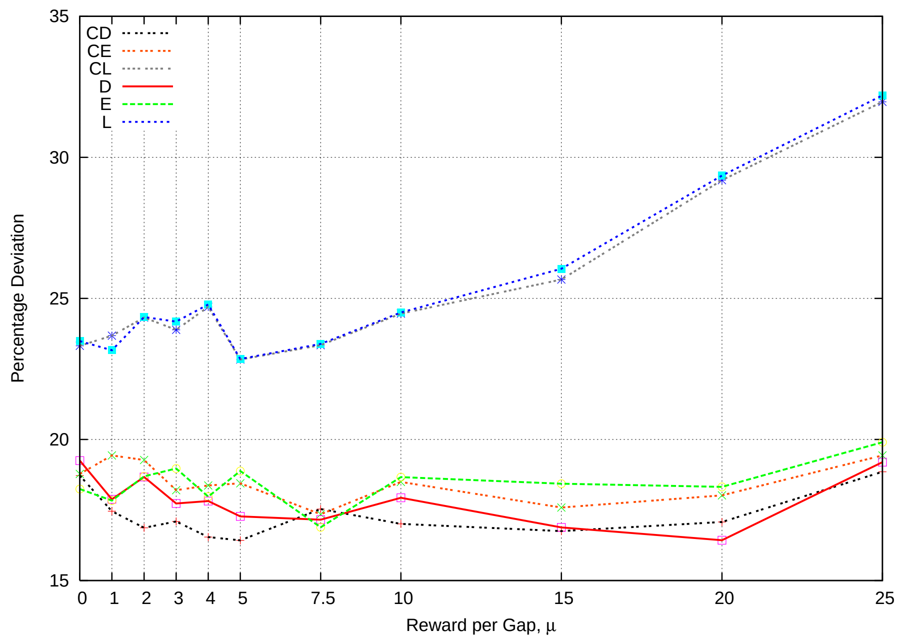


Figure 19: The performance of various policies for `Data20`.

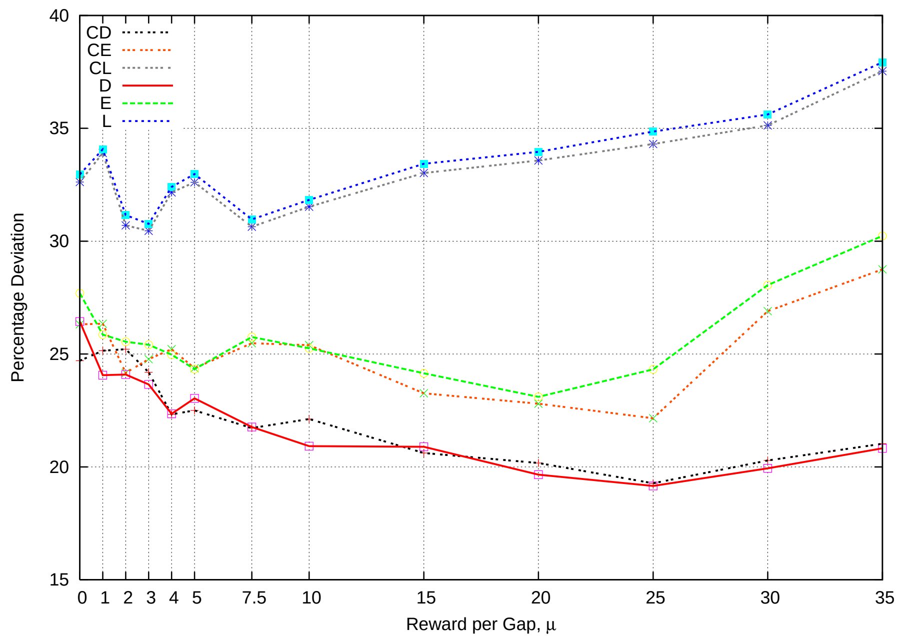


Figure 20: The performance of various policies for `Data40`.

We present the effect of Γ on the performance of `coverage` in Figures 21, 22 and 23. The new η_r formula is used with $g = 200$ and $\kappa = 30$. The default policy is `deadheading`.

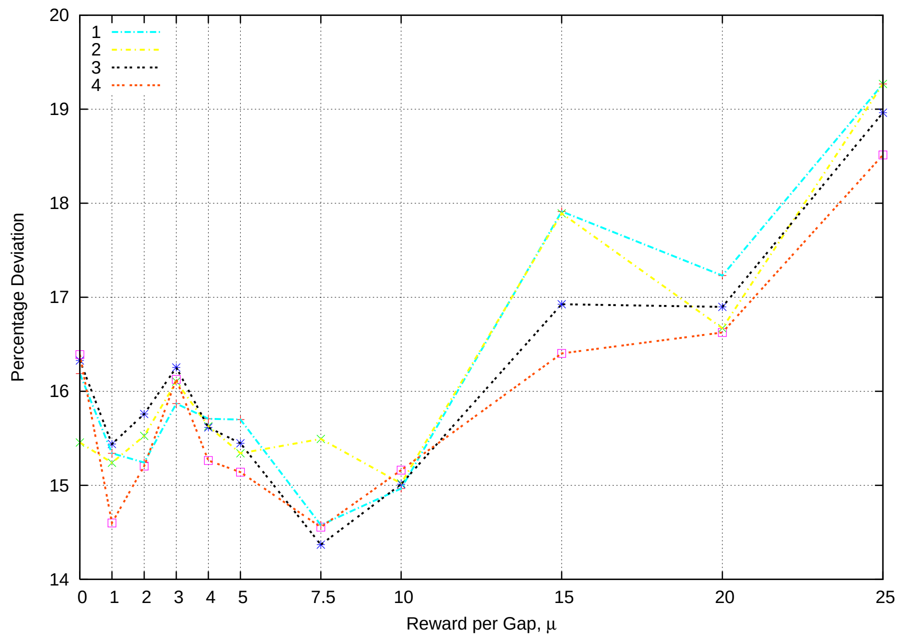


Figure 21: Performance of `Coverage` on `Data10` for various values of Γ.

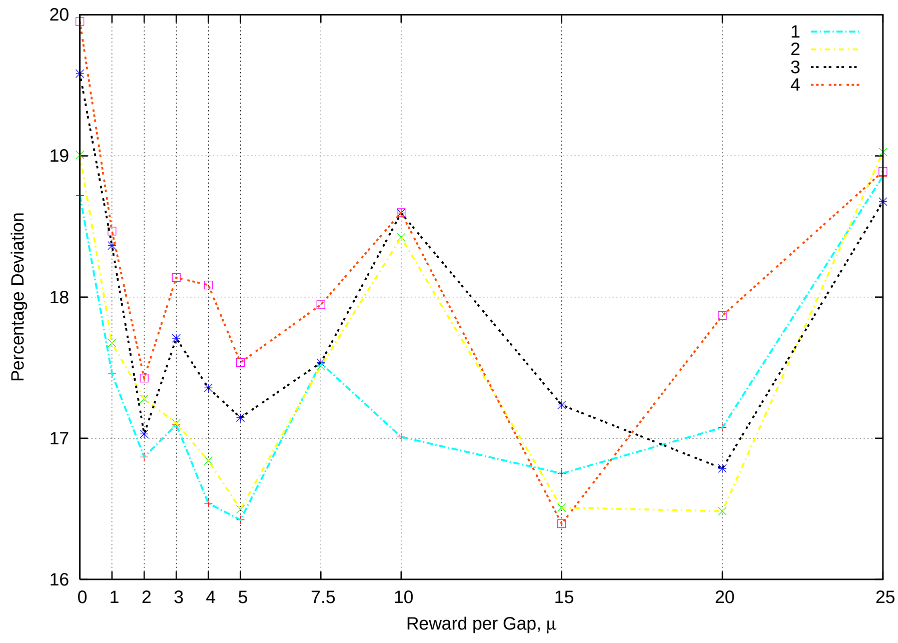


Figure 22: Performance of `Coverage` on `Data20` for various values of Γ.

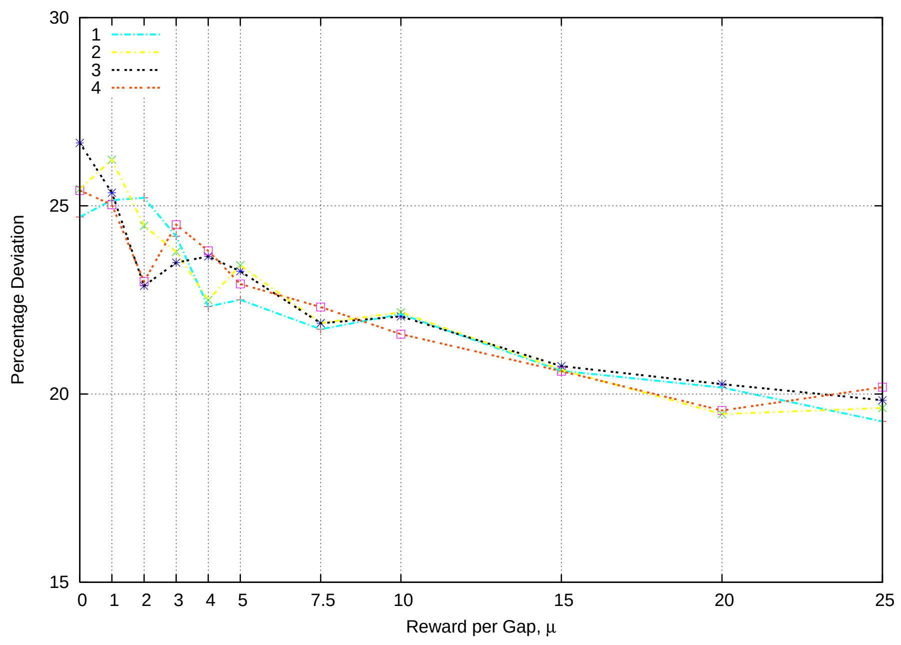


Figure 23: Performance of `Coverage` on `Data40` for various values of Γ.

The performance does seem to vary in some cases (e.g., $\mu = 15$ in `Data10`, $0 \leq \mu \leq 5$ in `Data20`), but it is hard to draw general conclusions about the best value for Γ.

3.10 A Priori Information on Potential Customer Locations

In order to assess the validity of the assumption that we do not know any future potential customer locations until orders are released at those locations, we collected statistics on the Northern California data. The statistics are summarized in Table 12. The locations are grouped into buckets by the number of days they appear in orders. The first column reports the range of the buckets. For instance, the first row holds the statistics for the bucket of locations that appear in orders on 21 or more days. The second column reports how many locations fall into that particular bucket. The percentage of the locations in a bucket is in the third column. The volume of a location (i.e., the number of orders originating at or destined to a location in the bucket) is in the fourth column. The last column reports the percentage of the volume.

The numbers show that about 15.6% of the volume is related to a small set of locations.

Table 12: Breakdown of customer locations with respect to recurrence

Buckets	Locations		Volume	
28-21	4	1.6%	309	15.6%
20-16	10	4.0%	402	20.2%
15-11	17	6.9%	355	17.8%
10-6	42	16.9%	484	24.3%
5-1	175	70.6%	439	22.21%

On the other hand, a significant portion of the volume is related to locations that appear at most five days. For future models, starting with some known potential customer locations is a possible generalization that seems to be the case for actual drayage operations. The additional information can be used to develop techniques that relocate idle drivers more effectively.

3.11 Conclusions and Future Directions

We modeled the dynamic addition of new orders into the Daily Drayage Problem after removing some of the complexities of the Static Daily Drayage Problem. In order to solve the resultant Online Daily Drayage Problem, we first used myopic optimization-based routing with various scheduling policies. Among the scheduling policies, `Deadhead` performed best on average and seemed to be robust. We modified the optimization-based method by rewarding the route assignments with more slack in their schedules. By increasing the reward to select route assignments which are more likely to allow order insertions, the online cost was reduced in most cases. As the reward increased beyond a certain value, the cost of lower utilization outweighed any potential gain.

In the Online Daily Drayage Problem, the objective is to minimize total deadhead cost. Alternative objective functions, such as maximizing the number of orders covered, can be used instead. We assumed that all drivers can be routed dynamically, but in reality, the route assignments for third party drivers are static. In a problem with static and dynamic routes, an additional question is which orders should be covered by static routes.

CHAPTER IV

DRAY COVERAGE PROBLEM WITH ONE CUSTOMER

4.1 Introduction

An important question for drayage operations is when (and for how long) the processing of an order should be delayed so that more information is available before a driver is committed to an order. Another issue is the relocation of idle drivers. We expect the time and the destination of a driver's relocation to have a significant impact on the solution when the percentage of unknown orders is high. We hope to gain some insights into how to relocate drivers in general and when to delay processing of orders by isolating a driver.

In this chapter, we define and study the Dray Coverage Problem with one customer. This problem includes a single driver. New orders are released randomly into the system. The goal is to maximize the expected number of orders covered by the single driver within a given time horizon.

By restricting our problem to a single driver, we focus on sequencing and scheduling orders on one route. The assignment of drivers to orders is no longer an issue. Instead, we need to select which orders are to be covered. Since the objective is to cover as many orders as possible, the solution still favors routes with high utilization.

In Section 4.2, we describe the Dray Coverage Problem with one customer. A Markov Decision Process model for the problem is given in Section 4.3. In Section 4.4, we describe the method for calculating the expected value of an optimal policy and show that there is an optimal policy in which an order at the driver's current location is never delayed. We study relocate-or-wait decisions in Section 4.5. We present computational results and observations about optimal policies for different parameters in Section 4.6. We show that a heuristic policy, called the seesaw policy, covers one order less than an optimal policy in the worst case in Section 4.7. In Section 4.8, we present a special case in which the seesaw policy is optimal. We extend the Markov Decision Process model to the case with three locations and provide some computational results in Section 4.9. We generalize an observation for the Dray Coverage Problem with one customer in Section 4.10. Our conclusions are in Section 4.11.

4.2 Description

The Dray Coverage Problem with one customer has the same simplifying assumptions as given in Section 3.3.1. The goal here is to maximize the number of orders covered by a single driver. In this special case with one customer, one location is the rail ramp and the other is a customer location. All future orders either originate from or are sent to the only customer location.

Let N be the number of periods in the time horizon. We assume that moving an order or deadheading takes one period. In each period, at most one additional order is released to the system. Let H denote the rail ramp, and L denote the customer location. The new order originates at H with probability p_H, or at L with probability p_L. The probability that no new order is added to the system at a particular time period is $1 - p_H - p_L$. We assume $p_H \geq p_L$, but if an order is more likely to originate at a customer location, we can pick H to be the customer location and the analysis remains the same. The driver has three choices–moving an order, deadheading to the other location or waiting at the current location.

In each period, there is a random event which potentially adds a new order to the system. A *scenario* is an ordered list of N event outcomes which describe all external future changes to the system. Given a fixed scenario, the problem becomes deterministic, and the movements of the driver can be plotted on a location-time graph as in Figure 24. A horizontal dashed line represents waiting, whereas a diagonal dashed line represents deadheading. A diagonal solid line corresponds to moving an order. For instance, the driver first waits at H, then moves an order to L, deadheads back to H in the following period and finally moves another order to L.

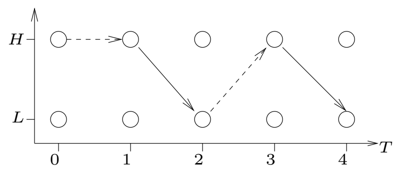


Figure 24: Driver's movements on a location-time graph

4.3 A Finite-Horizon Markov Decision Process Formulation

The problem can now be formulated as a finite-horizon Markov decision process [34]. In the formulation, the time index t denotes the number of periods left, hence the time index is initially N and decreases by one after each period until $t = 0$.

- **Decision Epochs:** At the beginning of each period, a decision has to be made which determines the driver's task for the period. The periods are indexed in descending order starting from N for the immmediate period to 1 for the final period. The time horizon is finite.

- **States:** The location of the driver and the number of orders ready to be picked up at locations H and L determine the state of the system. If the driver is at location α and there are i orders at location H and j orders at location L, the state of the system, s, is given by $s = (\alpha, i, j)$. The set of all states, S, is given by

$$S = \{(\alpha, i, j) : \alpha \in \{L, H\}, \text{ and } i, j \in Z_{+}^{0}\}.$$

  Since there is no limit on the number of orders at a location, $|S|$ is infinite. In Section 4.4, we show that we can treat S as a finite set.

- **Actions:** In general, there are three possible actions at each decision epoch–moving an order, waiting at the current location and deadheading to the other location, which are respectively denoted by $\{M, W, D\}$. The set of actions is defined for each state s and is denoted by A_s. Let $S_0 = \{(H, 0, j) : j \in Z_{+}^{0}\} \cup \{(L, i, 0) : i \in Z_{+}^{0}\}$ and $S_1 = S \setminus S_0$. In words, S_0 is the set of states in which there are no orders at the driver's current location. Therefore, moving an order is not a possible action for a state in S_0. Thus,

$$A_s = \begin{cases} \{W, D\} & \forall s \in S_0, \\ \{M, W, D\} & \forall s \in S_1. \end{cases}$$

- **Rewards:** The number of orders covered is the total reward. The reward depends on both the given state and the action chosen. A reward of 1 is earned for moving an order to its destination (M), while zero reward is earned for waiting (W) or deadheading (D). For $t = N, \ldots, 1$,

  $$r_t(s, a) = \begin{cases} 1 & \text{if } s \in S_1, a = M \\ 0 & \text{otherwise} \end{cases}$$

  The reward at the end of the time horizon is zero, therefore $r_0(s, .) = 0$, $\forall s \in S$.

- **Transition Probabilities:** Transition probabilities do not change over time. For $t = N, \ldots, 1$ and $(H, i, j) \in S_1$,

  $$p_t(s'|(H, i, j), M) = \begin{cases} p_H & \text{if } s' = (L, i, j), \\ p_L & \text{if } s' = (L, i-1, j+1), \\ 1 - p_H - p_L & \text{if } s' = (L, i-1, j), \\ 0 & \text{otherwise.} \end{cases}$$

  For $t = N, \ldots, 1$ and $(L, i, j) \in S_1$,

  $$p_t(s'|(L, i, j), M) = \begin{cases} p_H & \text{if } s' = (H, i+1, j-1), \\ p_L & \text{if } s' = (H, i, j), \\ 1 - p_H - p_L & \text{if } s' = (H, i, j-1), \\ 0 & \text{otherwise.} \end{cases}$$

For $t = N, \ldots, 1$ and $(\alpha, i, j) \in S$,

$$p_t(s'|(\alpha,i,j),D) = \begin{cases} p_H & \text{if } s' = (\alpha', i+1, j), \alpha' \neq \alpha, \\ p_L & \text{if } s' = (\alpha', i, j+1), \alpha' \neq \alpha, \\ 1 - p_H - p_L & \text{if } s' = (\alpha', i, j), \alpha' \neq \alpha, \\ 0 & \text{otherwise,} \end{cases}$$

$$p_t(s'|(\alpha,i,j),W) = \begin{cases} p_H & \text{if } s' = (\alpha, i+1, j), \\ p_L & \text{if } s' = (\alpha, i, j+1), \\ 1 - p_H - p_L & \text{if } s' = (\alpha, i, j), \\ 0 & \text{otherwise.} \end{cases}$$

The objective is to maximize the expected total reward, i.e., maximize the expected number of orders covered. A *decision rule* outputs an action given the current state. In an instance with N periods, a *policy* is a sequence of N decision rules. A policy can use different decision rules at different time periods. Our goal is to find an optimal or a good policy for the problem. An optimal policy may depend on a number of parameters.

4.4 Calculating Expected Values

Given a state s, we calculate $v_N(s)$, the optimal expected reward for N periods starting in state s, with the optimality equations

$$v_n(s) = \max_{a \in A_s} \left\{ r_n(s,a) + \sum_{s' \in S} p_n(s'|s,a) v_{n-1}(s') \right\}, \ 1 \leq n \leq N, \tag{14}$$

and the boundary condition $v_0(s) = r_0(s,.) = 0$, $\forall s \in S$. We solve the equations recursively by substituting $v_{n-1}(s')$ values when evaluating $v_n(s)$. A formal algorithm for solving these equations is given in [34], which is called the backward induction algorithm. For our problem, the backward induction algorithm can be written as in Algorithm 7.

We use several observations to speed up the calculations. Before stating the observations, we introduce some notation. Consider $s = (\alpha, x, y)$ and let $v_t(s)$ be the maximum expected number of orders covered when the system is in state s with t periods left. For $t \geq 1$, we define $v_t^a(s)$ as

$$v_t^a(s) = r_t(s,a) + \sum_{s' \in S} p_t(s'|s,a) v_{t-1}(s'), \ a \in A_s.$$

Algorithm 7 Backward Induction Algorithm

```
1: Set $n \leftarrow 0$ and $v_0(s) = r_0(s, .) = 0, \ \forall s \in S$.
2: Set $n \leftarrow n + 1$ and compute $v_n(s)$ for each $s \in S$ by
       $$v_n(s) = \max_{a \in A_s} \left\{ r_n(s,a) + \sum_{s' \in S} p_n(s'|s,a) v_{n-1}(s') \right\}.$$
3: If $n = N$, stop. Otherwise, return to 2.
```

Finally, let $p' = 1 - p_H - p_L$.

Since moving an order and deadheading take one period each, at most $\lfloor \frac{N}{2} \rfloor$ orders that do no originate at the driver's current location can be covered. For the orders originating at the driver's current location, the limit increases by one since the driver does not have to deadhead to pick up the first order. Observation 1 shows that we may treat S as a finite set.

Observation 1. *For $N \geq 1$ and $i, j \geq 0$,*

$$\begin{aligned} v_N(H,i,j) &= v_N(H, \min\{i, \left\lfloor \frac{N}{2} \right\rfloor + 1\}, \min\{j, \left\lfloor \frac{N}{2} \right\rfloor\}), \\ v_N(L,i,j) &= v_N(L, \min\{i, \left\lfloor \frac{N}{2} \right\rfloor\}, \min\{j, \left\lfloor \frac{N}{2} \right\rfloor + 1\}). \end{aligned}$$

Observation 2 states that an additional order at H or L cannot increase the expected value more than one.

Observation 2. *For $N \geq n \geq 1$, $\alpha \in \{H, L\}$ and $i, j \geq 0$,*

$$1 \geq v_n(\alpha, i+1, j) - v_n(\alpha, i, j), \ \textit{and} \ 1 \geq v_n(\alpha, i, j+1) - v_n(\alpha, i, j).$$

Observation 3 shows that if there are no orders at either location, the expected value of having the driver at H is equal to the the expected value of having the driver at L. When there are no orders in the system, the only question is whether the driver should be at L or at H in the beginning of the next period. If the optimal location for the next period is the same as the driver's current location, he can just wait. Otherwise, he can deadhead to the optimal location. Therefore, the current location of the driver does not change the optimal expected value when there are no orders in the system.

Observation 3. $v_N(H,0,0) = v_N(L,0,0), \forall N \geq 1$.

Proposition 3 states that when there is an order at the driver's current location, i.e., $s \in S_1$, there is an optimal solution in which the driver carries out such an order in the next period.

Proposition 3. *For* $1 \leq n \leq N$ *and* $s_1 \in S_1$, $v_n^M(s_1) \geq \max\{v_n^D(s_1), v_n^W(s_1)\}$.

Proof. Suppose $N = 1$. Then $v_1^M(s_1) = r_1(s_1, M) + \sum_{s' \in S} p_1(s'|s_1, M) \cdot 0 = 1$, $v_1^D(s_1) = r_1(s_1, D) + \sum_{s' \in S} p_1(s'|s_1, D) \cdot 0 = 0$, and $v_1^W(s_1) = r_1(s_1, W) + \sum_{s' \in S} p_1(s'|s_1, W) \cdot 0 = 0$. So, the proposition holds for $N = 1$. If $N = 2$, $v_2^M(s_1) = 1 + \sum_{s' \in S} p_1(s'|s_1, M) \cdot v_1(s') \geq 1$. For other actions, we have $v_2^D(s_1) = 0 + \sum_{s' \in S} p_1(s'|s_1, D) v_1(s') \leq 1$ and $v_2^W(s_1) = 0 + \sum_{s' \in S} p_1(s'|s_1, W) v_1(s') \leq 1$ since $v_1(s') \leq 1$, $\forall s' \in S$. The proposition holds for $N = 2$ as well. Suppose the proposition is true for $N \geq 2$ and $s_1 = (H, i, j)$ where $i \geq 1$. For $N + 1$, we have

$$\begin{aligned}
v_{N+1}^W(H,i,j) &= p_H v_N(H,i+1,j) + p_L v_N(H,i,j+1) + p' v_N(H,i,j) \\
&= p_H v_N^M(H,i+1,j) + p_L v_N^M(H,i,j+1) + p' v_N^M(H,i,j) \quad \text{(by induction)} \\
&= 1 + p_H^2 v_{N-1}(L,i+1,j) + 2p_H p_L v_{N-1}(L,i,j+1) + p_L^2 v_{N-1}(L,i-1,j+2) \\
&\quad + 2p_H p' v_{N-1}(L,i,j) + 2p_L p' v_{N-1}(L,i-1,j+1) + p'^2 v_{N-1}(L,i-1,j).
\end{aligned}$$

When we calculate a lower bound for $v_{N+1}^M(H,i,j)$, we get

$$\begin{aligned}
v_{N+1}^M(H,i,j) &= 1 + p_H v_N(L,i,j) + p_L v_N(L,i-1,j+1) + p' v_N(L,i-1,j) \\
&\geq 1 + p_H v_N^W(L,i,j) + p_L v_N^W(L,i-1,j+1) + p' v_N^W(L,i-1,j) \\
&= 1 + p_H^2 v_{N-1}(L,i+1,j) + 2p_H p_L v_{N-1}(L,i,j+1) + p_L^2 v_{N-1}(L,i-1,j+2) \\
&\quad + 2p_H p' v_{N-1}(L,i,j) + 2p_L p' v_{N-1}(L,i-1,j+1) + p'^2 v_{N-1}(L,i-1,j) \\
&= v_{N+1}^W(H,i,j).
\end{aligned}$$

If we compare $v^M_{N+1}(H,i,j)$ and $v^D_{N+1}(H,i,j)$, we have

$$\begin{aligned} v^M_{N+1}(H,i,j) - v^D_{N+1}(H,i,j) &= 1 + p_H(v_N(L,i,j) - v_N(L,i+1,j)) \\ &\quad + p_L(v_N(L,i,j+1) - v_N(L,i+1,j+1)) \\ &\quad + p'(v_N(L,i,j) - v_N(L,i+1,j)) \\ &\geq 1 - p_H - p_L - p' = 0 \text{ (by Observation 2)}. \end{aligned}$$

Therefore, we have $v^M_{N+1}(H,i,j) \geq \max\{v^D_{N+1}(H,i,j), v^W_{N+1}(H,i,j)\}$. The case for $s_1 = (L,i,j)$ can be proven analogously. □

For $1 \leq n \leq N$, $i \geq 1$ and $j \geq 0$, the optimality equations (14) can be rewritten as

$$\begin{aligned} v_n(H,i,j) &= 1 + p_H v_{n-1}(L,i,j) + p_L v_{n-1}(L,i-1,j+1) + p' v_{n-1}(L,i-1,j), \\ v_n(L,j,i) &= 1 + p_H v_{n-1}(H,j+1,i-1) + p_L v_{n-1}(H,j,i) + p' v_{n-1}(H,j,i-1), \\ v_n(H,0,j) &= \max\left\{ \begin{array}{l} p_H v_{n-1}(H,1,j) + p_L v_{n-1}(H,0,j+1) + p' v_{n-1}(H,0,j), \\ p_H v_{n-1}(L,1,j) + p_L v_{n-1}(L,0,j+1) + p' v_{n-1}(L,0,j) \end{array} \right\}, \\ v_n(L,j,0) &= \max\left\{ \begin{array}{l} p_H v_{n-1}(L,j+1,0) + p_L v_{n-1}(L,j,1) + p' v_{n-1}(L,j,0), \\ p_H v_{n-1}(H,j+1,0) + p_L v_{n-1}(H,j,1) + p' v_{n-1}(H,j,0) \end{array} \right\}, \end{aligned}$$

by using Proposition 3. We now know that M is the best action if $s \in S_1$. We are interested in finding which decisions maximize $v_N(s)$ if $s \in S_0$. We are particularly interested in $v_N(H,0,0)$, which is the maximum expected number of orders covered when there are no orders in the system. In Section 4.5, we analytically study the optimal action when there are no orders at the driver's current location from $N = 1$ to $N = 5$. For larger values of N, we numerically compute optimal actions for some values of p_H and p_L in Section 4.6.

4.5 Deadheading or Waiting

In this section, we study $v_N(s)$ for $N \leq 5$ and $s_0 \in S_0$ as a function of p_H and p_L. We consider three general cases within S_0:

1. The system is empty.

2. The driver is at H with no orders at H and at least one order at L.

3. The driver is at L with no orders at L and at least one order at H.

We are interested in optimal actions for the three cases. Given such a small problem with one customer and a single driver, we expected to find simple descriptions of optimal policies. This is indeed the case for $N = 1, 2, 3$. However, starting with states $(H, 0, 1)$ when $N = 4$ and $(H, 0, 0)$ when $N = 5$, optimal policies show dependence on probabilities for higher values of N. We study the optimal actions analytically up to $N = 5$ in this section. Numerical study of optimal actions for higher values of N are given in Section 4.6.

We use $A^*_N(s_0)$ to denote the set of optimal actions for state s_0 with N periods left, so $A^*_N(s_0) = \{a \in A_{s_0} : v_N(s_0) = v^a_N(s_0)\}$. By solving the optimality equations (14) up to $N = 3$, we get

- For $N = 1$, $v_1(s_0) = 0$ and $A^*_1(s_0) = \{D, W\}$.

- For $N = 2$ and $i \geq 1$,

$$\begin{aligned} &v_2(H,0,0) = p_H, \quad &&A^*_2(H,0,0) = \{W\}, \quad &&v_2(L,0,0) = p_H, \quad &&A^*_2(L,0,0) = \{D\}, \\ &v_2(H,0,i) = 1, &&A^*_2(H,0,i) = \{D\}, &&v_2(L,i,0) = 1, &&A^*_2(L,i,0) = \{D\}. \end{aligned}$$

- For $N = 3$ and $i \geq 1$,

$$\begin{aligned} v_3(H,0,0) &= 2p_H + p_L - p_H^2, \qquad & A^*_3(H,0,0) &= \{D, W\}, \\ v_3(H,0,i) &= 1 + 2p_H - p_H^2, & A^*_3(H,0,i) &= \{D\}, \\ v_3(L,0,0) &= 2p_H + p_L - p_H^2, & A^*_3(L,0,0) &= \{D, W\}, \\ v_3(L,i,0) &= 1 + 2p_L - p_L^2, & A^*_3(L,i,0) &= \{D\}. \end{aligned}$$

We have $v_N(H, 0, i) = v_N(H, 0, 1)$ and $v_N(L, i, 0) = v_N(L, 1, 0)$for $i \geq 1$ due to Observation 1. Let d^0 be a decision rule such that $d^0(s_0) = \displaystyle\bigcap_{N=1}^{3} A^*_N(s_0)$ for $s_0 \in S_0$ and $d^0(s_1) = M$

for $s_1 \in S_1$. Then,

$$d^0(\alpha, x, y) = \begin{cases} M & (\alpha, x, y) \in S_1, \\ D & \alpha = L, x \geq 0, y = 0, \\ W & \alpha = H, x = 0, y = 0, \\ D & \alpha = H, x = 0, y \geq 1. \end{cases}$$

For $N = 1, 2, 3$, the policy $\pi^0 = \{d^0, d^0, d^0\}$ is optimal by the construction of d^0. The policy π^0 uses the same decision rule at each decision epoch and does not depend on p_H or p_L. In words, the optimal policy is to move an order if there is any at the current location, relocate to H if the system is empty and deadhead to the other location if there is an order at the other location but not at the current location. Although π^0 seems somewhat intuitive, it does not necessarily yield the optimal value for larger values of N. The optimal action for $v_4(H, 0, 1)$ depends on p_H and p_L.

$$v_4(H,0,1) = \begin{cases} 1 + 3p_H + 2p_L - p_L^2 p_H - 3p_H^2 + p_H^3 + p_H^2 p_L - p_L^2 & \text{if } g(p_L, p_H) \geq 0, \\ 1 + 3p_H - p_H^2 & \text{if } g(p_L, p_H) < 0, \end{cases}$$

$$A_4^*(H,0,1) = \begin{cases} D & \text{if } g(p_L, p_H) \geq 0, \\ W & \text{if } g(p_L, p_H) < 0, \end{cases}$$

where $g(a, b) = -2b^2 - a^2 - ba^2 + b^3 + 2a + b^2 a$. The optimal action for $v_5(H, 0, 0)$ depends on p_H and p_L as well.

$$v_5^D(H,0,0) = -p_H^2 + 4p_H - p_L^2 - 2p_H^3 + 3p_L + p_H^4,$$

$$v_5^W(H,0,0) = \begin{cases} -p_H p_L^3 - p_H^2 + 4p_H - p_L^2 + p_H p_L^2 - 2p_H^3 & \text{if } g(p_L, p_H) \geq 0, \\ \quad +3p_L + p_H^4 + p_H^3 p_L - p_H^2 p_L & \\ p_L^3 - p_H^2 + 4p_H - 3p_L^2 + p_H p_L^2 - 2p_H^3 & \text{if } g(p_L, p_H) < 0. \\ \quad +3p_L + p_H^4 - p_H^2 p_L^2 + p_H^2 p_L & \end{cases}$$

Let $f(a, b) = -a^2 + 2a - ba + b^2 a - b^2$ and y' be the only real root of the equation $b^3 + b - 1 = 0$. Let R denote the set of feasible values for p_H and p_L, i.e., $R = \{(p_L, p_H) : p_L + p_H \leq 1, p_H \geq p_L \geq 0\}$. We partition R into R^W, R^D and R^E such that

- $v_5^W(H, 0, 0) > v_5^D(H, 0, 0)$ if $(p_L, p_H) \in R^W$,

- $v_5^W(H,0,0) = v_5^D(H,0,0)$ if $(p_L, p_H) \in R^E$,
- $v_5^W(H,0,0) < v_5^D(H,0,0)$ if $(p_L, p_H) \in R^D$.

For the probability values in R^W and R^D, the optimal actions are waiting and deadheading respectively. For values in R^E, the optimal action can be either waiting or deadheading. By studying $v_5^W(H,0,0)$ and $v_5^D(H,0,0)$, which are polynomials of p_H and p_L, we can say that

$$\begin{aligned}
R^W &= \{(p_L, p_H) \in R : p_L \neq 0, f(p_L, p_H) < 0\}, \\
R^E &= \{(p_L, p_H) \in R : p_L = p_H\} \bigcup \{(p_L, p_H) \in R : f(p_L, p_H) = 0\} \\
&\quad \bigcup \{(p_L, p_H) \in R : p_L = 0, p_H > 0\} \\
&\quad \bigcup \{(p_L, p_H) \in R : 0.5 \leq p_H \leq y' \approx 0.6823, p_L = 1 - p_H\}, \\
R^D &= R \setminus (R^W \cup R^E).
\end{aligned}$$

For $v_5(L,0,0)$, by Observation 3, we conclude that

$$v_5(L,0,0) = v_5(H,0,0), A_5^*(L,0,0) = \begin{cases} \{D\}, & (p_L, p_H) \in R^W, \\ \{D, W\}, & (p_L, p_H) \in R^E, \\ \{W\}, & (p_L, p_H) \in R^D. \end{cases}$$

Let $\Delta(p_H, p_L, N)$ denote the difference between expected values of deadheading and waiting in the first time period given p_H, p_L, N, and the initial state $(H,0,0)$, i.e, $\Delta(p_H, p_L, N) = v_N^D(H,0,0) - v_N^W(H,0,0)$. Then,

$$\Delta(p_H, p_L, 5) = \begin{cases} p_H p_L^3 - p_H p_L^2 - p_H^3 p_L + p_H^2 p_L & \text{if } g(p_L, p_H) \geq 0, \\ -p_L^3 + 2p_L^2 - p_H p_L^2 + p_H^2 p_L^2 - p_H^2 p_L & \text{if } g(p_L, p_H) < 0. \end{cases}$$

The contours of $\Delta(p_H, p_L, 5)$ are given in Figure 25. The maximum of $\Delta(p_H, p_L, 5)$ is attained around $(p_L, p_H) = (0.2067, 0.5238)$ and the minimum of $\Delta(p_H, p_L, 5)$ is attained around $(p_L, p_H) = (0.1480, 0.8520)$. The maximum and the minimum values are respectively around 0.009 and -0.07. On the contour that partially lies on the $p_H = p_L$ line, $\Delta(p_H, p_L, 5)$ is zero.

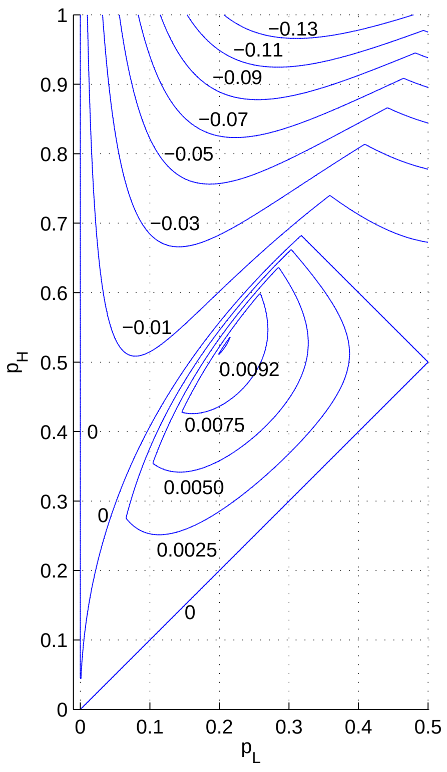


Figure 25: Contours of $\Delta(p_H, p_L, 5)$.

In Figure 26, all possible values of p_H and p_L are partitioned into three regions. The optimal actions for $(H, 0, 0)$ when $N = 5$ and for $(H, 0, 1)$ when $N = 4$ in each region is given in Table 13. Region A is R^W, region B is $\{(p_L, p_H) \in R^D : g(p_L, p_H) < 0\}$ and region C is $\{(p_L, p_H) \in R^D : g(p_L, p_H) > 0\}$.

Table 13: Optimal actions in each region

Region	$(H, 0, 0)$, $N = 5$	$(H, 0, 1)$, $N = 4$
A	W	W
B	D	W
C	D	D

The optimal actions for the remaining states in S_0 when $N = 4$ or 5 can be summarized as

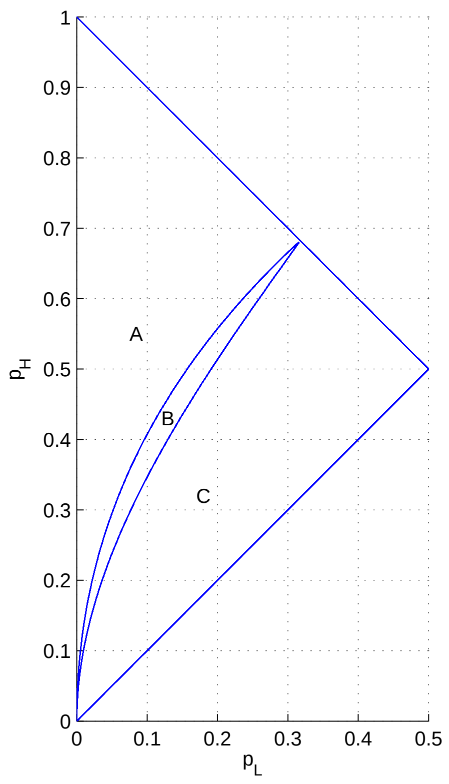


Figure 26: Decision regions for $(H, 0, 0)$, $N = 5$ and $(H, 0, 1)$, $N = 4$.

- For $N = 4$ and $i \geq 2$,

$$v_4(H,0,0) = -p_L^2 + 2p_L - p_H^2 + 3p_H, \qquad A_4^*(H,0,0) = \{W\},$$

$$v_4(H,0,i) = 2 - p_H^2 + 2p_H, \qquad A_4^*(H,0,i) = \{D\},$$

$$v_4(L,0,0) = v_4(H,0,0), \qquad A_4^*(L,0,0) = \{D\},$$

$$v_4(L,1,0) = 1 + p_H^3 - p_L^2 + 2p_L - 3p_H^2 + 3p_H, \qquad A_4^*(L,1,0) = \{D\},$$

$$v_4(L,i,0) = 2 - p_L^2 + 2p_L, \qquad A_4^*(L,i,0) = \{D\}.$$

- For $N = 5$ and $i \geq 2$,

$$\begin{aligned}
v_5(H,0,1) &= 1 - 2p_H^3 - 3p_L^2 + 3p_L - p_H^2 + p_L^3 & A_5^*(H,0,1) &= \{D\}, \\
&\quad + 4p_H + p_H^4, \\
v_5(H,0,i) &= 2 - 2p_H^3 - p_H^2 + 4p_H + p_H^4, & A_5^*(H,0,i) &= \{D\}, \\
v_5(L,1,0) &= 1 + 2p_L p_H + 4p_H^3 + 3p_L - 6p_H^2 - p_L p_H^2 & A_5^*(L,1,0) &= \{D\}, \\
&\quad - p_L^3 + 4p_H + 2p_L^2 p_H^2 - p_H^4 - 3p_L^2 p_H, \\
v_5(L,i,0) &= 2 - p_L^2 + 4p_L - 2p_L^3 + p_L^4, & A_5^*(L,i,0) &= \{D\}.
\end{aligned}$$

The actions chosen by policy π^0 is optimal for these remaining states in S_0.

We showed that there are states for which the optimal action depends on p_H and p_L when $N = 4, 5$. When the system is empty with five periods left, the intuitive decision of relocating the driver to location H is not optimal for some values of p_H and p_L although $p_H \geq p_L$. Unexpected rules for optimal actions are not limited to the case of the empty system. When the driver is at location H and the only order in the system is at location L, the optimal action could waiting or deadheading depending on p_H, p_L and N. The rules determining the optimal actions become harder to find analytically as N increases. We study the optimal actions for higher values of N numerically in Section 4.6.

4.6 Computational Experiments

We know that for $s \in S_1$, moving an order is an optimal action for the first decision epoch regardless of N, p_H and p_L. Hence, we evaluated $v_N(H,0,0)$, $v_N(L,1,0)$, $v_N(L,2,0)$, $v_N(H,0,1)$, and $v_N(H,0,2)$ for various values of p_H, p_L and N.

1. In general, $v_N(H,0,0)$ was optimized with policies which start with waiting at H for the first period. For some (p_H, p_L) pairs, there was an odd number $m(p_H, p_L)$ such that for all odd N between 5 and $m(p_H, p_L)$, the optimal policies started with deadheading in the first period. We showed that N must be at least 5 in order to have optimal policies starting with a deadhead and presented the conditions on (p_H, p_L) which determine the initial optimal action for $(H,0,0), N = 5$ in Section 4.5. In

Table 14, $m(p_H, p_L)$ values for various p_H and p_L combinations are listed. Blank entries correspond to infeasible (p_H, p_L) pairs (e.g. $p_L > p_H$), whereas X means waiting was the optimal action for all $(H, 0, 0), N = 1, \ldots, 100$ for the corresponding (p_H, p_L) pair.

Table 14: $m(p_H, p_L)$ values for various p_H and p_L combinations.

$p_L \backslash p_H$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.05	11	5	X	X	X	X	X	X	X
0.10	X	9	5	5	X	X	X	X	X
0.15		23	9	5	X	X	X	X	X
0.20		X	15	7	5	X	X	X	
0.25			37	11	7	5	X		
0.30			X	21	9	5	X		
0.35				59	13	7			
0.40				X	25	X			
0.45					85				
0.50					X				

We observed that having $\Delta(p_H, p_L, N) > 0$ for some N, implied $\Delta(p_H, p_L, 5) > 0$. We also noted that for all (p_H, p_L) pairs such that $\Delta(p_H, p_L, N) > 0$,

$$\Delta(p_H, p_L, 5) \geq \Delta(p_H, p_L, 7) \geq \cdots \geq \Delta(p_H, p_L, m(p_H, p_L)).$$

2. Waiting and deadheading are the only two possible actions for $v_N(H, 0, i)$, $i = 1, 2$. There were two threshold values n_o and n_e such that for every odd $N < n_o$ and for every even $N < n_e$, deadheading was the optimal action for the first period. For all odd $N \geq n_o$ and for all even $N \geq n_e$, waiting was the optimal action for the first period. The threshold values n_o and n_e depended on p_H, p_L and i. Both n_e and n_o increased with i while n_o was always greater than n_e. In Table 15, (n_e, n_o) pairs are listed for a number of p_H and p_L combinations when $i = 1$. In Table 16, (n_e, n_o) pairs are listed for a number of p_H and p_L combinations when $i = 2$. In both tables, X means there was no $N \leq 100$ for which the optimal action switched from D to W. Blank entries correspond to infeasible (p_H, p_L) values.

3. Deadheading was the optimal action for the first period for $v_N(L, i, 0)$, $N = 1, \ldots, 100$, and $i = 1, 2$.

Table 15: (n_e, n_o) pairs for $(H, 0, 1)$

$p_L \backslash p_H$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.05	10,X	6,X	4,11	4,9	4,9	4,7	4,7	4,7	4,7
0.10	X,X	8,X	6,17	4,11	4,11	4,9	4,9	4,7	4,7
0.15		16,X	8,23	6,15	4,11	4,11	4,9	4,7	
0.20		X,X	10,39	6,19	6,15	4,11	4,9	4,9	
0.25			20,X	8,29	6,17	4,13	4,11		
0.30			X,X	14,47	8,23	6,17	4,13		
0.35				26,X	10,35	8,21			
0.40				X,X	18,65	10,31			
0.45					54,X				
0.50					X,X				

Table 16: (n_e, n_o) pairs for $(H, 0, 2)$

$p_L \backslash p_H$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.05	18,X	10,X	8,19	6,13	6,11	6,11	6,11	6,9	6,9
0.10	X,X	14,X	10,27	8,17	6,13	6,13	6,11	6,11	6,9
0.15		28,X	12,45	10,21	8,17	6,15	6,13	6,11	
0.20		X,X	18,X	12,29	8,21	8,17	6,15	6,13	
0.25			34,X	16,41	10,25	8,19	8,17		
0.30			X,X	22,67	14,35	10,25	8,19		
0.35				44,X	20,51	14,33			
0.40				X,X	36,93	22,47			
0.45					X,X				
0.50					X,X				

We observed that optimal actions continue to depend on p_H, p_L and N for $N > 5$. An unforeseen result was the effect of the parity of N on optimal actions. For instance, the optimal action in an empty system was to relocate to H whenever N was even and depended on p_H and p_L only for some odd N values. The optimal actions for wait-or-deadhead decisions at H show even more interesting behavior. We summarize our numerical conclusions in Table 17. Although, we do not have proof whether any of the conclusions in Table 17 are true in general, we have not encountered any case which contradicts the conclusions.

For the wait-or-deadhead situation with no orders at the driver's current location and at least one order at the other location, the optimal action seems to be deadheading to H if the driver is at L. For the other case where the driver is at H, i.e., $(H, 0, i)$ with $i \geq 1$, the optimal action is deadheading for smaller values of N and waiting for larger values of

Table 17: Conclusions suggested by numerical results

State		Optimal Action
$(H, 0, 0)$	:	Wait at H if N is even
$(H, 0, 0)$	:	Wait at H or deadhead to L depending on (p_H, p_L) if N is odd
$(H, 0, 0)$	:	Wait at H for large enough odd values of N regardless of (p_H, p_L)
$(L, i, 0)$, $i \geq 1$	:	Deadhead to H
$(H, 0, i)$,$i \geq 1$	:	Deadhead to L if N is small, and otherwise wait at H

N. The threshold value of N, for which the optimal action changes from deadheading to waiting, not only depends on values of p_H and p_L but also on i and the parity of N. For fixed p_H, p_L and i, the threshold value for odd values of N is higher than the threshold value for even values of N. When p_H, p_L and the parity of N are fixed, the threshold value increases as the number of orders at the other location increases.

4.7 A Near-Optimal Policy

Since the optimal policies depend heavily on N, p_H, p_L and the initial state, we searched for a simpler policy which would give good solutions. In this section, we study the properties of a near-optimal policy, which we call the seesaw policy.

At each time period, the seesaw policy follows the same simple decision rule–never wait. If an order is available for delivery, then move it; otherwise deadhead. What the seesaw policy might look like on the location-time graph is in Figure 27. The policy is simple and does not depend on the probability distribution, the initial state or the number of time periods remaining.

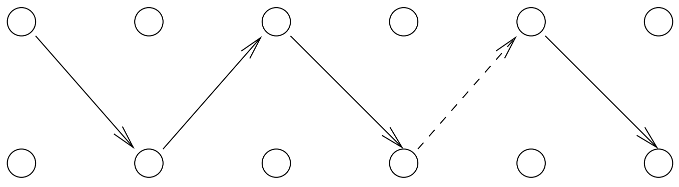


Figure 27: A solution given by the seesaw policy

More formally, let d^1 denote the decision rule used by the seesaw policy, denoted by π^1, in each decision epoch. Then,

$$d^1(s) = \begin{cases} M & s \in S_1, \\ D & s \in S_0. \end{cases} \quad \text{and } \pi^1 = \{d^1, d^1, \dots, d^1\}.$$

Theorem 2. *The difference between the expected number of orders covered under the seesaw policy and the optimal policy is at most one.*

We show that Theorem 2 is true by first studying the deterministic case. We compare the seesaw policy with another policy that we call the wait-and-seesaw policy, denoted π^2. In the wait-and-seesaw policy, the driver waits in the first time period, and then follows the seesaw policy, i.e., $\pi^2 = \{W, d^1, d^1, \dots, d^1\}$.

Proposition 4. *For a fixed scenario of events, the number of orders covered by the seesaw policy is not less than the number of orders covered by the wait-and-seesaw policy minus one.*

Proof. Given a solution under the wait-and-seesaw policy, one can construct a new solution as in Figure 28. The selected portion of the original solution is repeated with one period delay. Since, in the new solution, no orders are carried out until period two, the remainder is still feasible.

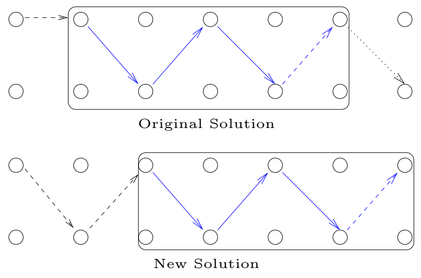


Figure 28: Constructing a new solution from the wait-and-seesaw policy's solution

The new solution covers one less order if an order is carried out in the last period in the original solution, which is represented by a dotted line in Figure 28. Otherwise, the solution covers the same number of orders. The new solution is dominated by the seesaw policy's solution which follows the same path in the location-time graph and carries out orders as soon as possible. $\square$

Proposition 5. *For a fixed scenario of events, either the seesaw policy or the wait-and-seesaw policy covers the optimal number of orders.*

Proof. For a fixed scenario, an optimal solution can be described by a sequence of wait, deadhead and move actions. Suppose that we picked an optimal solution. In our proof, we transform the optimal solution into a form comparable to the solutions given by the seesaw and the wait-and-seesaw policies. The transformations are summarized in Figure 29.

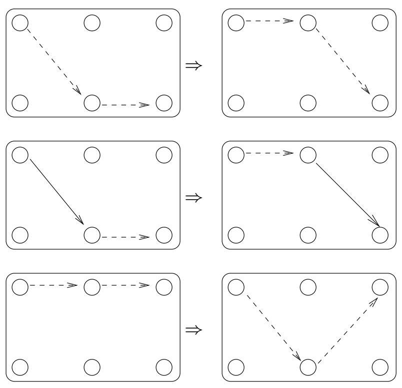


Figure 29: The transformations preserve feasibility of the solution and the number of orders covered.

In an optimal solution, if a wait decision follows deadheading, we can get a new solution by swapping the waiting and deadheading actions. The new solution is still feasible and covers the same number of orders as the original. The same is true if moving an order is followed by waiting. If we keep replacing D, W with W, D and M, W with W, M until none remain in the solution, the resultant solution has a number of W's followed by D's and M's. Moreover, we can replace all consecutive wait decisions with consecutive deadhead decisions. Hence, if the number of W's is even, we end up with a solution that follows the same path on the location-time graph as the solution of the seesaw policy. If the number of W's is odd, the final solution has only one W in the beginning, followed by M's and D's. Either way, the final solution covers the optimal number of orders and is dominated by either the seesaw policy's solution or the wait-and-seesaw policy's solution. □

By combining Propositions 4 and 5 that are related to the deterministic case, we can argue that Theorem 2 is true because one plus the number of orders covered by the seesaw policy is greater than or equal to the maximum number of orders covered in each scenario.

4.8 A Special Case With a Never Empty Location H

Since there is always an order at H, we can denote the states as $(\alpha, ., j)$ where α is the driver's location and j is the number of order at location L. Let p_L be the probability that a new order is released at L at any time period.

Theorem 3. *The seesaw policy is optimal if there is always an order at H.*

Proof. When there is at least one order at the current location, under the seesaw policy the order is carried out, which we know is optimal. The state $(L, ., 0)$ is the only state in which there is no order at the current location. Under the seesaw policy, the driver should deadhead to H. So, if we can show that deadheading is optimal, then the proof is complete. For $N = 1$, $v_1^D(L, ., 0)_d = v_1^W(L, ., 0) = 0$ and for $N = 2$, $v_2^D(L, ., 0)_d - v_2^W(L, ., 0) = 1 - p_L$, so we may assume $N \geq 3$. We have

$$\begin{aligned} v_N^D(L, ., 0) - v_N^W(L, ., 0) = 1 &+ (1 - p_L)^2 v_{N-2}(L, ., 0) + 2(1 - p_L)p_L v_{N-2}(L, ., 1) \\ &+ p_L^2 v_{N-2}(L, ., 2) - (1 - p_L)v_{N-1}(L, ., 0) - p_L v_{N-1}(L, ., 1). \end{aligned}$$

Since $2v_{N-2}(L, ., 1) \geq v_{N-2}(L, ., 1) + v_{N-2}(L, ., 0)$, we can rewrite the right hand side and change the equality to inequality as

$$\begin{aligned} v_N^D(L, ., 0) - v_N^W(L, ., 0) \geq 1 &+ (1 - p_L)^2 v_{N-2}(L, ., 0) + (1 - p_L)p_L v_{N-2}(L, ., 0) \\ &+ (1 - p_L)p_L v_{N-2}(L, ., 1) + p_L^2 v_{N-2}(L, ., 1) \\ &- (1 - p_L)v_{N-1}(L, ., 0) - p_L v_{N-1}(L, ., 1). \end{aligned}$$

After collecting the terms, we get

$$\begin{aligned} v_N^D(L, ., 0) - v_N^W(L, ., 0) \geq 1 &+ (1 - p_L)v_{N-2}(L, .0) + p_L v_{N-2}(L, ., 1) \\ &- (1 - p_L)v_{N-1}(L, ., 0) - p_L v_{N-1}(L, ., 1). \end{aligned}$$

Since $v_{N-2}(s) - v_{N-1}(s) \geq -1$,

$$\begin{aligned} v_N^D(L,.,0) - v_N^W(L,.,0) &\geq 1 + (1-p_L)\Big(v_{N-2}(L,.0) - v_{N-1}(L,.0)\Big) \\ &\quad + p_L\Big(v_{N-1}(L,.0) - v_{N-1}(L,.1)\Big), \\ &\geq 1 - (1-p_L) - p_L = 0. \end{aligned}$$

Therefore, deadheading to H is optimal for $(L,.,0)$, and hence the seesaw policy is optimal.

Alternatively, pick a scenario and suppose waiting is strictly better than deadheading. Pick an optimal solution starting with waiting and remove the first and last time periods to obtain route r that starts at L and spans $N-2$ periods. In the worst case, route r covers one less order than the optimal value. We can construct a new solution in which the driver initially deadheads to H, moves an order from H to L and then follows route r. Since the driver does not move any order originating from L until the third period, the new solution is feasible and covers at least the optimal number of orders. This contradicts the fact that waiting is strictly better than deadheading for the given scenario. $\square$

Under the seesaw policy, the driver is back in the same location every two periods. Suppose the driver is initially at H and the time horizon is infinite. By looking at the system every two periods, it is possible to define a Markov chain. The definition and properties of Markov chains can be found in [35]. The states and the transition probabilities are given in Figure 30.

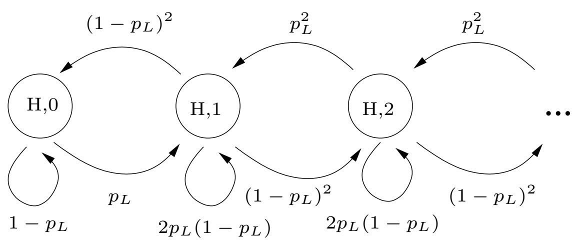


Figure 30: Markov states and transition probabilities

The probability transition matrix P is given as

$$P = \begin{pmatrix} 1-p_L & p_L & & \\ (1-p_L)^2 & 2p_L(1-p_L) & p_L^2 & \\ & \ddots & \ddots & \ddots \end{pmatrix}.$$

We need to solve $\Pi = \Pi \cdot P$ for Π in order to find the stationary distribution. We get

$$\Pi_0 = \frac{1-2p_L}{1-p_L}, \quad \Pi_k = \frac{p_L^{2k-1}}{(1-p_L)^{2k}} \cdot \Pi_0, \forall k \geq 1.$$

The stationary distribution exists if

$$\left(\frac{p_L}{1-p_L}\right)^2 < 1 \Rightarrow p_L < 0.5.$$

The average coverage for two periods is

$$1 + (1-\Pi_0) \cdot 1 + \Pi_0 \cdot p_L = 1 + \frac{p_L}{1-p_L} + \frac{p_L - 2p_L^2}{1-p_L} = 1 + 2p_L.$$

4.9 Extension to Two Customers

In this problem, we add a second customer location, denoted by K, to the problem. Orders that originate at H must be delivered to either K or L, and vice versa. Recall that H represents the rail ramp. Since all drayage orders either originate or end at the rail ramp, we require that all orders must be either picked up at H or delivered to H in the case with two customers. Without loss of generality, we may assume that new orders are more likely to originate at K than L. To be consistent with the two location case, we assume that a new order is more likely to appear at H than all other locations combined. The change in the location layout and the order types are in Figure 31. Bold arrows in Figure 31 represent possible origin-destination pairs for orders. moving an order or deadheading from one location to another takes one period.

4.9.1 An Extended Finite-Horizon Markov Decision Process Formulation

The finite-horizon Markov decision process formulation can be extended to the case with two customer locations.

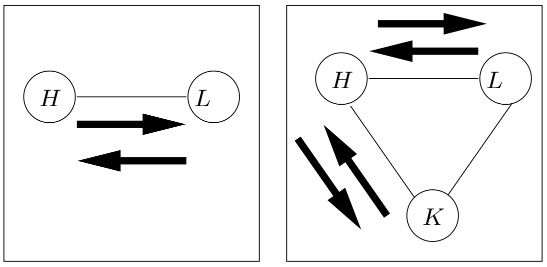


Figure 31: New problem setup with two customer locations compared to the original problem setup.

- **Decision Epochs:** At the beginning of each period, a decision has to be made which determines the driver's task for the period. The periods are indexed in descending order starting from N for the immmediate period to 1 for the final period. The time horizon is finite.

- **States:** Let n_ℓ and n_k denote the number of orders whose destinations are location L and location K, respectively. The total number of orders ready to be picked up at H is equal to the sum of n_ℓ and n_k. Let n_L and n_K denote the number of orders whose origins are location L and location K, respectively. Recall that the destination of all orders originating at locations L and K is location H. If the driver is at location α, the state of the system, s, is given by $s = (\alpha, n_\ell, n_k, n_L, n_K)$. The set of all states, S, is given by

$$S = \{(\alpha, n_\ell, n_k, n_L, n_K) : \alpha \in \{L, H, K\}, \text{ and } n_\ell, n_k, n_L, n_K \in Z_+^0\}.$$

- **Actions:** Overall, there are seven possible actions which are listed in Table 18. For

Table 18: All possible actions

Action	Description
W	Wait at current location
D_h	Deadhead to H
D_k	Deadhead to K
D_ℓ	Deadhead to L
M_h	Move order to H
M_k	Move order to K
M_ℓ	Move order to L

$\alpha \in \{K, H, L\}$, let S_0^α denote the set of states in which the driver is at location α, and there are no orders originating from location α. Let S_1^δ denote the set of states in which there is at least one order originating from the driver's current location to location H if $\delta = h$, to location L if $\delta = \ell$, and to location K if $\delta = k$. The set of actions is defined for each state s and is denoted by A_s.

$$A_s = \begin{cases} \{W, D_\ell, D_k\} & \forall s \in S_0^H, \\ \{W, D_h, D_k\} & \forall s \in S_0^L, \\ \{W, D_\ell, D_h\} & \forall s \in S_0^K, \\ \{M_h, W, D_h, D_k\} & \forall s \in S_1^h \text{ with } s = (L, n_\ell, n_k, n_L, n_K), \\ \{M_h, W, D_h, D_\ell\} & \forall s \in S_1^h \text{ with } s = (K, n_\ell, n_k, n_L, n_K), \\ \{M_\ell, W, D_h, D_\ell\} & \forall s \in S_1^\ell \setminus S_1^k, \\ \{M_k, W, D_h, D_k\} & \forall s \in S_1^k \setminus S_1^\ell, \\ \{M_\ell, M_k, W, D_\ell, D_k\} & \forall s \in S_1^k \cap S_1^\ell. \end{cases}$$

- **Rewards:** The number of orders covered is the reward. The reward at the end of the time horizon is zero, i.e., $r_0(s, .) = 0$, $\forall s \in S$. The reward at the decision epochs depends on the given state and the action chosen. A reward of 1 is gained for moving an order to its destination (M_ℓ, M_k, M_h) and zero reward is given for waiting (W) or deadheading (D_ℓ, D_k, D_h). For $t = N, \ldots, 1$,

$$r_t(s, a) = \begin{cases} 1 & \text{if } s \in S_1^h, a = M_h \\ 1 & \text{if } s \in S_1^\ell \setminus S_1^k, a = M_\ell \\ 1 & \text{if } s \in S_1^k \setminus S_1^\ell, a = M_k \\ 1 & \text{if } s \in S_1^k \cap S_1^\ell, a \in \{M_k, M_\ell\} \\ 0 & \text{otherwise.} \end{cases}$$

- **Transition Probabilities:** Transition probabilities do not change over time. The probability distribution of a new order is given in Table 19.

Table 19: All possible events

Probability	Origin	Destination
p_ℓ	H	L
p_k	H	K
p_K	K	H
p_L	L	H
$p' = 1 - p_\ell - p_k - p_K - p_L$	No new order	

For $t = N, \ldots, 1$ and $s = (H, n_\ell, n_k, n_L, n_K) \in S_1^k$,

$$p_t(s'|s, M_k) = \begin{cases} p_\ell & \text{if } s' = (K,\, n_\ell + 1,\, n_k - 1,\, n_L,\, n_K), \\ p_k & \text{if } s' = (K,\, n_\ell,\, n_k,\, n_L,\, n_K), \\ p_L & \text{if } s' = (K,\, n_\ell,\, n_k - 1,\, n_L + 1,\, n_K), \\ p_K & \text{if } s' = (K,\, n_\ell,\, n_k - 1,\, n_L,\, n_K + 1), \\ p' & \text{if } s' = (K,\, n_\ell,\, n_k - 1,\, n_L,\, n_K), \\ 0 & \text{otherwise.} \end{cases}$$

For $t = N, \ldots, 1$ and $s = (H, n_\ell, n_k, n_L, n_K) \in S_1^\ell$,

$$p_t(s'|s, M_\ell) = \begin{cases} p_\ell & \text{if } s' = (L,\, n_\ell,\, n_k,\, n_L,\, n_K), \\ p_k & \text{if } s' = (L,\, n_\ell - 1,\, n_k + 1,\, n_L,\, n_K), \\ p_L & \text{if } s' = (L,\, n_\ell - 1,\, n_k,\, n_L + 1,\, n_K), \\ p_K & \text{if } s' = (L,\, n_\ell - 1,\, n_k,\, n_L,\, n_K + 1), \\ p' & \text{if } s' = (L,\, n_\ell - 1,\, n_k,\, n_L,\, n_K), \\ 0 & \text{otherwise.} \end{cases}$$

For $t = N, \ldots, 1$ and $s = (L, n_\ell, n_k, n_L, n_K) \in S_1^h \cap S^L$,

$$p_t(s'|s, M_h) = \begin{cases} p_\ell & \text{if } s' = (H,\, n_\ell + 1,\, n_k,\, n_L - 1,\, n_K), \\ p_k & \text{if } s' = (H,\, n_\ell,\, n_k + 1,\, n_L - 1,\, n_K), \\ p_L & \text{if } s' = (H,\, n_\ell,\, n_k,\, n_L,\, n_K), \\ p_K & \text{if } s' = (H,\, n_\ell,\, n_k,\, n_L - 1,\, n_K + 1), \\ p' & \text{if } s' = (H,\, n_\ell,\, n_k,\, n_L - 1,\, n_K), \\ 0 & \text{otherwise.} \end{cases}$$

For $t = N, \ldots, 1$ and $s = (K, n_\ell, n_k, n_L, n_K) \in S_1^h \cap S^K$,

$$p_t(s'|s, M_h) = \begin{cases} p_\ell & \text{if } s' = (H, n_\ell + 1, n_k, n_L, n_K - 1), \\ p_k & \text{if } s' = (H, n_\ell, n_k + 1, n_L, n_K - 1), \\ p_L & \text{if } s' = (H, n_\ell, n_k, n_L + 1, n_K - 1), \\ p_K & \text{if } s' = (H, n_\ell, n_k, n_L, n_K), \\ p' & \text{if } s' = (H, n_\ell, n_k, n_L, n_K - 1), \\ 0 & \text{otherwise.} \end{cases}$$

For $t = N, \ldots, 1$ and $s = (\alpha, n_\ell, n_k, n_L, n_K) \in S_0$,

$$p_t(s'|s, W) = \begin{cases} p_\ell & \text{if } s' = (\alpha, n_\ell + 1, n_k, n_L, n_K), \\ p_k & \text{if } s' = (\alpha, n_\ell, n_k + 1, n_L, n_K), \\ p_L & \text{if } s' = (\alpha, n_\ell, n_k, n_L + 1, n_K), \\ p_K & \text{if } s' = (\alpha, n_\ell, n_k, n_L, n_K + 1), \\ p' & \text{if } s' = (\alpha, n_\ell, n_k, n_L, n_K), \\ 0 & \text{otherwise.} \end{cases}$$

For $t = N, \ldots, 1$ and $s = (\alpha, n_\ell, n_k, n_L, n_K) \in S_0 \setminus S_0^L$,

$$p_t(s'|s, D_\ell) = \begin{cases} p_\ell & \text{if } s' = (L, n_\ell + 1, n_k, n_L, n_K), \\ p_k & \text{if } s' = (L, n_\ell, n_k + 1, n_L, n_K), \\ p_L & \text{if } s' = (L, n_\ell, n_k, n_L + 1, n_K), \\ p_K & \text{if } s' = (L, n_\ell, n_k, n_L, n_K + 1), \\ p' & \text{if } s' = (L, n_\ell, n_k, n_L, n_K), \\ 0 & \text{otherwise.} \end{cases}$$

For $t = N, \ldots, 1$ and $s = (\alpha, n_\ell, n_k, n_L, n_K) \in S_0 \setminus S_0^K$,

$$p_t(s'|s, D_k) = \begin{cases} p_\ell & \text{if } s' = (K, n_\ell + 1, n_k, n_L, n_K), \\ p_k & \text{if } s' = (K, n_\ell, n_k + 1, n_L, n_K), \\ p_L & \text{if } s' = (K, n_\ell, n_k, n_L + 1, n_K), \\ p_K & \text{if } s' = (K, n_\ell, n_k, n_L, n_K + 1), \\ p' & \text{if } s' = (K, n_\ell, n_k, n_L, n_K), \\ 0 & \text{otherwise.} \end{cases}$$

For $t = N, \ldots, 1$ and $s = (\alpha, n_\ell, n_k, n_L, n_K) \in S_0 \setminus S_0^H$,

$$p_t(s'|s, D_h) \quad = \quad \begin{cases} p_\ell & \text{if } s' = (H, n_\ell + 1, n_k, n_L, n_K), \\ p_k & \text{if } s' = (H, n_\ell, n_k + 1, n_L, n_K), \\ p_L & \text{if } s' = (H, n_\ell, n_k, n_L + 1, n_K), \\ p_K & \text{if } s' = (H, n_\ell, n_k, n_L, n_K + 1), \\ p' & \text{if } s' = (H, n_\ell, n_k, n_L, n_K), \\ 0 & \text{otherwise.} \end{cases}$$

- **Assumptions:** We assume that $p_\ell + p_k \geq \max\{p_L, p_K\}$ and $p_K \geq p_L$.

The objective is to maximize the expected total reward, i.e., to maximize the expected number of orders covered.

4.9.2 Optimal Decisions for $N = 1, 2, 3$

For $N = 1, 2$, the value function calculations are straightforward. The optimal decisions for $N = 2$ can be summarized as

- if there is one order at the driver's current location, move the order,
- else if there is more than one order at the driver's current location, move the order whose destination is more likely to have an order in the next period,
- else if there is an order at another location, deadhead to that location,
- otherwise relocate to H.

For $N = 3$, location K is the optimal relocation point when there are no orders in the system. When the driver is at K or L with only one order originating at H and destined to L, the choice between relocating to K or H depends on the probabilities. If $(p_K - p_L)(1 - p_k) - (p_L - p_L^2) \geq 0$, the optimal action is to relocate to K, otherwise to H. As expected, for large enough p_L values, the optimal relocation point is H, since the driver is more likely to have an order ready at L after two periods spent deadheading to H and moving the already released order from H to L. If the probability of having a new order at K is more than a certain threshold, the optimal relocation point is K. As the probability of having a new order with origin H and destination K increases, the expression's value changes in favor of relocating to H. The decisions for $N = 3$ are summarized in Table 20. The problem with

Table 20: Optimal decisions at various states when $N = 3$ with $i, j \geq 1$.

$(H, 0, 0, 0, 0)$	D_k
$(H, 0, 0, 0, 1)$	D_k
$(H, 0, 0, 1, 0)$	D_l
$(H, 0, 0, 1, 1)$	D_k/D_l
$(H, 0, i, \ , \ )$	M_k
$(H, i, 0, \ , \ )$	M_l
$(H, i, j, 0, 0)$	M_k
$(H, i, j, 0, 1)$	M_k
$(H, i, j, 1, 0)$	M_l
$(H, i, j, 1, 1)$	M_k/M_l
$(K, 0, 0, 0, 0)$	W
$(K, 1, 0, 0, 0)$	D_h or W
$(K, \ , \ , \ , i)$	M_h
$(K, 0, \ , 1, 0)$	D_l
$(K, \ , 1, 0, 0)$	D_h
$(K, 1, \ , 1, 0)$	D_h/D_l
$(L, 0, 0, 0, 0)$	D_k
$(L, 1, 0, 0, 0)$	D_h or D_k
$(L, \ , \ , i, \ )$	M_h
$(L, \ , 0, 0, 1)$	D_k
$(L, \ , 1, 0, 0)$	D_h
$(L, \ , 1, 0, 1)$	D_h/D_k

one customer location is a special case of the problem with two customers with p_L set to 0. Hence, all of the results of the case with one customer carry over. The addition of a new customer begins to have non-trivial affects on the optimal actions as early as $N = 3$. For the Dray Coverage Problem with One Customer, both H and L are equally good choices for relocation in an empty system when $N = 3$. When there is an additional customer location K with a higher probability of having a new order than L, the optimal relocation point changes to K. The dependence of optimal actions on parameter values starts to appear at $N = 3$ for states $(L, 1, 0, 0, 0)$ and $(K, 1, 0, 0, 0)$.

4.10 Generalizations

Consider the Dray Coverage Problem with one customer and with continuous time horizon and the additional requirement that the driver must start and end at a park location. If we assume that total stop duration of any order is constant, we can generalize Proposition 3, which states moving an order immediately is optimal if the order is available at the driver's current location.

At time t, given when and where the driver will be available next, an order is *feasible* if the driver can move the order and deadhead to the park location by time T.

Proposition 6. *At time t, if the driver is idle at the origin of a feasible order, say order i, which is ready for pickup (i.e., released before time t) and the total stop duration of any order is the same, then there is an optimal solution in which the driver starts to move order i at time t.*

Proof. Suppose that there is an optimal schedule s', where the driver does not pick up order i at time t. Such a schedule must cover at least one feasible order since order i is feasible. Let o be the first order covered in s' and t_o be the completion time of order o.

If order o has the same origin and destination pair as order i, then one can get a new schedule by replacing the first order with order i. The new schedule can start at time t. The activities following the first order can start at their original times in s'. The new schedule covers the same number of orders.

If order o goes in the opposite way, there has to be a deadhead to the destination of order i in the beginning of s'. We construct a new schedule, which starts with order i and covers the optimal number of orders, in two steps. First, we replace the deadhead in s' with order i. The new schedule can again start at time t. The activities following order i until the pickup of order o are scheduled as early as possible. Therefore, the pickup of order o is at most delayed by the total stop duration of order i. As the second step of the construction, we replace order o with a deadhead to the origin of order i. Since the total stop duration of any order is the same, the deadhead to the origin of order i can be completed before t_o. The activities following order o can start at their original times in the new schedule. $\square$

Proposition 6 is not true for the general case with multiple customer locations. Consider the example in Figure 32. Let the stop duration at any location be half an hour. Suppose that the driver is idle at the rail ramp at time $T - 6.5$. By moving order 2 and then order

1, the driver can be back at the park location by $T - 0.5$. However, if the driver starts by delivering order 1 which is ready for pickup at the rail ramp, there is not enough time to move order 2. In this example, it is strictly better not to move order 1 right away although order 1 is ready for pickup at the driver's current location.

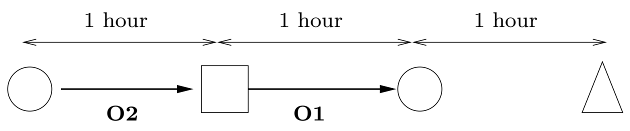


Figure 32: The time horizon does not allow the driver to start with order 2 and then to cover order 1.

It is possible to modify the finite-horizon Markov Decision Process model in Section 4.3 to handle more general cases. If the driver is required to be back at a park location, we can introduce a new state representing the parked driver and extend transition probabilities accordingly. If the stop durations of orders are different, the state space can be extended to differentiate orders with different durations.

4.11 Conclusions

Although the Dray Coverage Problem with one customer is a very simplified problem, the structure of the optimal policies depends on various parameters and is sometimes counter-intuitive. We believe the-end-of-day effect is the reason for the complex behavior of optimal policies. Although it is not possible to give a simple description of an optimal policy for all parameter values, the difference between the optimal expected value and the expected value of the seesaw policy is at most one. The seesaw policy is easy to state and does not depend on parameters. In addition, the seesaw policy is optimal if there is always an order at H. In the case with two customer locations, we observe that the optimal policy's dependency on parameters starts at $N = 3$. For the case with two locations, it is optimal to pick up an order available at the driver's current location, but the result does not generalize to the cases with three or more locations.

CHAPTER V

CONCLUSIONS AND FUTURE DIRECTIONS

In this thesis, we have addressed issues in modeling and optimizing daily drayage operations. In Chapter 2, our emphasis was on capturing the complicated aspects of real-world drayage operations in our model and finding a solution method that would yield optimal or near-optimal results. In Chapters 3 and 4, we defined and studied two abstract daily drayage problems with uncertainty. Our goal was to gain insights into managing drayage operations when dynamic addition of orders was allowed.

In Chapter 2, we were able to model the Static Daily Drayage Problem with complicated cost functions and constraints using a column generation-based approach. The solution methods used simple preprocessing techniques such as dominance among route assignments and the precedence feasibility matrix to improve running times. The generation-based method was able to solve the instances that were created from historical data within one percent of the optimal value in a reasonable time.

We believe the reason for the small integrality gap and fast convergence of the solution method is the fact that the number of orders on a feasible drayage route is typically limited by 3 or 4 due to the duration of drays relative to duration of driver shifts. Solving instances with larger set of orders may require additional tuning of the solution method. The model can be extended to include staging movements and movements with containers by defining new types of drayage orders. The biggest challenge is limiting capacity at equipment pools. Modeling trailer capacity is inherently difficult because trailers used in intermodal transportation are shared with other over-the-road and regional networks.

Another future direction for the Static Daily Drayage Problem is to extend the time horizon to weeks or months. Tactical questions such as "What should be the driver capacity?" and "What is the best mix of company and third party drivers?" can be addressed. Additional complexities can be modeled over a longer time horizon. For instance, when solving a daily problem, the maximum daily work hours for a driver is determined by his or her driving history and hours-of-service restrictions. With a longer time horizon, the model can use company drivers more effectively by including hours-of-service restrictions that span multiple days. Another complexity that can be modeled is storage cost at rail ramps. Keeping equipment at the rail ramp for more than a certain number of days results in holding cost.

In the remainder of the thesis, we incorporated some uncertainty, namely the dynamic addition of drayage orders, into the problem. In Chapter 3, we defined the Online Daily Drayage Problem in which there is no prior knowledge about future orders and the objective is minimizing total deadhead mileage. We made some assumptions to simplify the problem and to guarantee that all orders were feasible.

We have shown that the competitive ratio for any deterministic online algorithm must be greater than 2.0641. However we do not know whether this is a tight bound, i.e., whether a deterministic online algorithm with competitive ratio of 2.0641 exists. The initial solution methodology we developed repeatedly solves a myopic problem whenever a new order is released. The methodology has two major steps–selection of routes that cover known orders and scheduling of the selected routes. We later modified the solution methodology to reward routes with gaps in their schedules. We evaluated the performance of online algorithms by comparing their solution values with the value of the fair solution. We have seen that the performance of various scheduling policies can be improved by rewarding routes that can potentially cover additional future orders. Among the scheduling policies that schedule each route individually, `deadhead` has performed the best on average. With the `coverage` policy, we were able to improve the performance of `deadhead` by rescheduling the selected routes, so that gaps in the schedules covered more of the time horizon.

A potentially promising technique for improving online algorithms is to anticipate the origin of dynamic orders and to relocate idle drivers accordingly. With strategic relocation of idle drivers in an online algorithm, the online solution can outperform the fair solution. Under the assumption that we do not have any prior knowledge of future orders, it is very challenging to come up with a good strategy for relocating idle drivers. Since each relocation is a deadhead and increases the cost of the route, a poorly chosen relocation decision can easily ruin the objective value of a solution. Our initial attempts in using strategic relocation performed poorly. In our methodology, we have solved integer programs to find the best selection of routes. The impact of selecting the best routing solution instead of a good heuristic routing solution on the overall performance of an online algorithm is another issue worth investigating.

In Chapter 4, we studied Dray Coverage Problem with One Customer, where a single driver moves orders, which either originate at the rail ramp or at the customer, and the objective is to maximize the expected number of orders covered. Our goal was to gain insights about wait-or-deadhead decisions and when (and for how long) an order should be delayed. We have shown that moving an order originating at the driver's current location is optimal when there is one customer location. This is not necessarily true for the case with two customers. When there are no orders at the driver's current location, the optimal action has turned out to depend on many parameters although the problem setup is very simple. We have shown that a heuristic policy guarantees a solution value of the maximum expected number of orders covered less one for any starting state and any set of parameters. We have observed that the complicated dependence of the optimal action on parameters and the initial state continued for the case with two customers.

The Markov Decision Process model can be extended to account for the duration difference between moving an order and deadheading. We do not know whether a heuristic with a performance guarantee exists for the case with two customers. Studying the optimal policies for multiple drivers can provide additional insights.

REFERENCES

[1] Ascheuer, N., Fischetti, M., and Grötschel, M., "A polyhedral study of the asymmetric traveling salesman problem with time windows," *Networks*, vol. 36, pp. 69–79, 2000.

[2] Ascheuer, N., Krumke, S. O., and Rambau, J., "The online transportation problem: competitive scheduling of elevators." Technical Report 98-34, Konrad-Zuse-Zentrum fur Informationstechnik Berlin, 1998.

[3] Association of American Railroads, "Rail intermodal transportation," July 2006. http://www.aar.org/GetFile.asp?File_ID=143.

[4] Bard, J. F., Kontoravdis, G., and Yu, G., "A branch-and-cut procedure for the vehicle routing problem with time windows," *Transportation Science*, vol. 36, pp. 250–269, 2002.

[5] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H., "Branch-and-price: column generation for solving huge integer programs," *Operations Research*, vol. 46, pp. 316–329, 1998.

[6] Bent, R. and Hentenryck, P., "Scenario based planning for partially dynamic vehicle routing problems with stochastic customers," *Operations Research*, vol. 53, pp. 977–987, 2004.

[7] Bertsimas, D. J. and Simch-Levi, D., "A new generation of vehicle routing research: robust algorithms, addressing uncertainty," *Operations Research*, vol. 44, pp. 286–304, 1996.

[8] Bräysy, O. and Gendreau, M., "Vehicle routing problem with time windows, part i: Route construction and local search algorithms," *Transportation Science*, vol. 39, pp. 104–118, 2005.

[9] Bräysy, O. and Gendreau, M., "Vehicle routing problem with time windows, part ii: Metaheuristics," *Transportation Science*, vol. 39, pp. 119–139, 2005.

[10] Campbell, A. and Savelsbergh, M., "Decision support for consumer direct grocery initiatives," *Transportation Science*, vol. 39, pp. 313–327, 2005.

[11] Cook, W. and Rich, J. L., "A parallel cutting plane algorithm for the vehicle routing problem with time windows." Technical Report TR99-04, Computational and Applied Mathematics, Rice University, 1999.

[12] Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M., and Soumis, F., *The Vehicle Routing Problem*, ch. The VRP with Time Windows, pp. 157–193. Monographs on Discrete Mathematics and its Applications, SIAM, 2001.

[13] Desrochers, M., Desrosiers, J., and Solomon, M., "A new optimization algorithm for the vehicle routing problem with time windows," *Operations Research*, vol. 40, pp. 342–354, 1992.

[14] Desrosiers, J., Dumas, Y., Solomon, M. M., and Soumis, F., *Network Routing*, ch. Time Constrained Routing and Scheduling, pp. 35–139. Elsevier Science, 1995.

[15] Dictionary.com, "dray," September 2006. *Dictionary.com Unabridged (v 1.0.1).* Based on the Random House Unabridged Dictionary, © Random House, Inc. 2006. http://dictionary.reference.com/browse/dray.

[16] Dumas, Y., Desrosiers, J., and Soumis, F., "The pickup and delivery problem with time windows," *European Journal of Operational Research*, vol. 54, pp. 7–22, 1991.

[17] Fleischmann, B., Gnutzmann, S., and Sandvoss, E., "Dynamic vehicle routing based on online traffic information," *Transportation Science*, vol. 38, pp. 420–433, 2004.

[18] Gendreau, M., Laporte, G., and Seguin, R., "An exact algorithm for the vehicle routing problem with stochastic demand and customers," *Transportation Science*, vol. 29, pp. 143–155, 1995.

[19] Gendreau, M. and Potvin, J., *Fleet Management and Logistics*, ch. Dynamic vehicle routing and dispatching, pp. 115–126. Kluwer, 1998.

[20] Grötschel, M., Krumke, S. O., and Rambau, J., *Online Optimization of Large Scale Systems: State of the Art.* Springer, 2001.

[21] ILOG, Inc., *ILOG CPLEX 9.0 User's Manual*, 2003.

[22] Jula, H., Dessouky, M., Ioannou, P., and Chassiakos, A., "Container movement by trucks in metropolitan networks: modeling and optimization," *Transportation Research Part E*, pp. 235–259, 2005.

[23] Justice, E., *Optimization of chassis reallocation in doublestack container transportation systems.* PhD dissertation, University of Arkansas, 1996.

[24] Kohl, N., *Exact methods for time constrained routing and related scheduling problems.* PhD dissertation, Technical University of Denmark, 1995.

[25] Kohl, N., Desrosiers, J., Madsen, O., Solomon, M., and Soumis, F., "2-path cuts for the vehicle routing problem with time windows," *Transportation Science*, vol. 33, pp. 101–116, 1999.

[26] Kohl, N. and Madsen, O., "An optimization algorithm for the vehice routing problem with time windows based on lagrangian relaxation," *Operations Research*, vol. 45, pp. 395–406, 1997.

[27] Krumke, S., Rambau, J., and Torres, L., "Real-time dispatching of guided and unguided automobile service units with soft wime windows," in *Algorithms - ESA 2002*, vol. 2461 of *Lecture notes in computer science*, pp. 637–648, Springer Berlin/Heidelberg, 2002.

[28] Laporte, G., Louveaux, F., and Mercure, H., "The vehicle routing problem with stochastic travel times," *Transportation Science*, vol. 26, pp. 161–170, 1992.

[29] Macharis, C. and Bontekoning, Y. M., "Opportunities for OR in intermodal freight transport research: A review," *European Journal of Operational Research*, vol. 153, pp. 400–416, 2004.

[30] Morlok, E. K. and Spasovic, L. N., "Redesigning rail-truck intermodal drayage operations for enhanced service and cost performance," *Transportation Research Forum*, vol. 34, pp. 16–31, 1994.

[31] Nemhauser, G. L. and Wolsey, L. A., *Integer and Combinatorial Optimization.* John Wiley & Sons, Inc., 1999.

[32] Powell, W. B., Towns, M. T., and Marar, A., "On the value of optimal myopic solutions for dynamic routing and scheduling problems in the presence of user noncompliance," *Transportation Science*, vol. 34, pp. 67–85, 2000.

[33] Psaraftis, H. N., "Dynamic vehicle routing: Status and prospects," *Annals of Operations Research*, vol. 61, pp. 143–164, 1995.

[34] Puterman, M., *Markov Decision Process: Discrete Stochastic Dynamic Programming.* John Wiley and Sons, Inc., 1994.

[35] Ross, S., *Stochastic Processes.* John Wiley and Sons, Inc., second ed., 1996.

[36] Savelsbergh, M., "Local searchin routing problems with time windows," *Annals of Operations Research*, vol. 4, pp. 285–305, 1985.

[37] Savelsbergh, M. and Sol, M., "The general pickup and delivery problem," *Transportation Science*, vol. 29, pp. 17–29, 1995.

[38] Savelsbergh, M. and Sol, M., "Drive: Dynamic routing of independent vehicles," *Operations Research*, vol. 46, pp. 474–490, 1998.

[39] Smilowitz, K., "Multi-resource routing with flexible tasks: an application in drayage operations," *IIE Transactions*, vol. 38, pp. 555–568, July 2006.

[40] Spasovic, L. N., *Planning intermodal drayage network operations.* PhD dissertation, University of Pennsylvania, 1990.

[41] Spivey, M. Z. and Powell, W. B., "The dynamic assignment problem," *Transportation Science*, vol. 38, pp. 399–419, 2004.

[42] Swihart, M. R. and Papastavrou, J., "A stochastic and dynamic model for the single-vehicle pick-up and delivery problem," *European Journal of Operational Research*, vol. 114, pp. 447–464, 1999.

[43] Taylor, G., Broadstreet, F., Meinert, T., and Usher, J., "An analysis of intermodal ramp selection methods," *Transportation Research Part E*, vol. 38, pp. 117–134, 2002.

[44] Thomas, B. and White III, C., "Anticipatory route selection," *Transportation Research Part E*, vol. 38, pp. 473–487, 2004.

[45] Walker, W., "Network economies of scale in short haul truckload operations," *Journal of Transportation Economics and Policy*, vol. 26, pp. 3–17, 1992.

[46] Wang, X. and Reagan, A., "Local truckload pickup and delivery with hard time window constraints," *Transportation Research Part B*, vol. 36, pp. 97–112, 2002.

[47] Wikipedia.com, "Dray," August 2006. http://en.wikipedia.org/wiki/Dray.

[48] Wikipedia.com, "Intermodal freight transportation," August 2006. http://en.wikipedia.org/wiki/Intermodal_freight_transport.

[49] Xu, H., Chen, Z.-L., Rajagopal, S., and Arunapuram, S., "Solving a practical pickup and delivery problem," *Transportation Science*, vol. 37, pp. 347–364, 2003.

[50] Yang, J., Jaillet, P., and Mahmassani, H., "Real-time multivehicle truckload pickup and delivery problems," *Transportation Science*, vol. 38, pp. 135–148, 2004.