Order Fulfillment Algorithms for Online Retail

by

Pin-Yi Chen

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mechanical Engineering (Interdisciplinary Degree in Analytics for Supply Chain Management)

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2023

Author .....
Department of Mechanical Engineering
October 14, 2022

Certified by .....
Stephen Graves
Abraham J. Siegel Professor of Management Science
Thesis Supervisor

Accepted by .....
Nicolas G. Hadjiconstantinou
Chairman, Department Committee on Graduate Theses

Order Fulfillment Algorithms for Online Retail

by

Pin-Yi Chen

Submitted to the Department of Mechanical Engineering
on October 14, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Mechanical Engineering (Interdisciplinary Degree in
Analytics for Supply Chain Management)

Abstract

We study an order fulfillment problem in an online-retail setting where the retailer's fulfillment system includes warehouses that hold inventory (fulfillment centers), and a transportation network composed of node facilities and transportation arcs. To minimize the total transportation costs for order fulfillment, the online retailer should plan its transportation capacities properly in advance and execute according to the transportation plan by making smart fulfillment decisions. To fulfill a customer order, the online retailer must decide from where to source the inventory needed for the order, as well as how to route the order to the customer to satisfy a delivery time commitment. In this research we focus on the latter decision, namely choosing the route for each order. The online retailer has full control of its transportation system in both planning and execution; in addition, the online retailer can rely on third party carriers to transport some of its orders.

We design an order fulfillment algorithm that makes immediate routing decisions for incoming orders. To determine the route to assign to an order, we compare the costs of all feasible routes. For routes that use retailer-controlled resources, we need to account for the opportunity costs associated with these resources. We propose to do this with the dual values from a transportation quadratic program, which tries to account for the uncertainty of the network flows to estimate the opportunity costs of depleting resources in the transportation network. The dual values are updated periodically with the most updated system states that include resource capacities and demand forecasts. We numerically test our algorithm on a realistic network with inputs inspired by the actual data from our industry collaborator. We compare our algorithm with several benchmark algorithms, including a LP-based algorithm that mimics the algorithm in the retailer's current operating system. The experiments show a 50% reduction in the mean percentage difference in shipping cost from the hindsight solution as compared to the LP-based algorithm.

In addition to the fulfillment algorithm, we formulate a capacity planning problem that determines the optimal level of transportation resources in the network for a given demand forecast. When the demand deviates from the planned capacity by a

lot, adding or removing planned resources becomes a crucial cost-saving mechanism. Motivated by this, we propose ad hoc truck controllers that make online capacity modification decisions and test it with small examples.

Thesis Supervisor: Stephen Graves
Title: Abraham J. Siegel Professor of Management Science

Acknowledgments

My gratitude towards my advisor, Professor Stephen Graves, is beyond words. He was brave enough to accept me as his Ph.D. student despite my slight knowledge of the field of operations research when we first met. His generosity has changed my life trajectory and given me the chance to learn and grow in this field. His guidance was invaluable in helping me conduct this research.

Further thanks go to Professor Georgia Perakis and Professor Alexandre Jacquillat for serving as my committee members. They have been extremely supportive. Throughout the pandemic, our meetings have always been virtual, but scheduling meetings with them has always been a breeze.

I would like to thank our industry partners: Tolga Cezik, Daniel Chen, and Tamar Cohen-hillel for all the thought-provoking discussions.

I deeply appreciate the unwavering support from my parents. They have kept me grounded through all my ups and downs. I would also like to thank my bosom friends, especially Qi, Jinglong, Lily, Boyi, Longjing, for always being there for me. And lastly, I would like to thank James for being my source of happiness.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Literature Reviews

Online retail (Ecommerce) sales take up an increasingly high percentage of total retail sales over the past 25 years. The U.S. Ecommerce sales as a percent of total Retail Sales has reached 16% (U.S. census bureau [5]) in 2020 Q3 during the pandemic, which is almost triple the percentage in 2013 Q2. Riding this wave of Ecommerce growth, Amazon has experienced exponential growth, and has become the largest online retailer in the U.S.; its total sales account for 50 percent of the entire U.S. Ecommerce retail market [7]. Interestingly, Amazon is not alone, as monopolistic online retailers are ubiquitous around the world. For example, Shoppe and Lazada are dominating in Southeast Asia; JD.com and Alibaba Group (who controls multiple online retail channels such as Taobao and Tianmao) are leading the Chinese Ecommerce market. Each of these online retail giants process a large volume of packages, and their order delivery costs have increased dramatically. Amazon, for example, has increased its spending on shipping from 3.99 billion dollars in 2011 to 76.70 billion dollars in 2021 [9].

Over the last two decades, the way that these online retail giants fulfill orders has continued to evolve. The online retailers receive and store product inventory in their network of fulfillment centers. When a customer places an order, the retailer decides from which fulfillment center to supply the order and then picks the order and prepares it for shipping. Initially, the retailer would then transfer the order to a third-party carrier (like FedEx or UPS) to deliver the order from the fulfillment

center to the customer. In this setting, the order fulfillment problem for the online retailer is to make a real-time decision of from which warehouse (fulfillment center) to fulfill each order (which we refer to as the "inventory-assignment" decision of the order fulfillment problem). This decision is primarily based on the inventory availability across the fulfillment centers and on the third-party costs for shipping.

An early paper that addresses this problem is Xu and Graves [22]. They identify shortcomings of the "myopic" policy, which picks the fulfillment center solely based on a given third-party per package costs without consideration of current inventory states and future demand arrivals. To improve upon these fulfillment decisions, they propose an algorithm that periodically reevaluates the decisions and reassigns orders to reduce the number of split shipments for multi-item orders. (Under similar context, the value of delaying fulfillment decisions is further investigated by Wei et al. [21], in particular, the balance between the cost-saving from consolidating delayed shipment and the cost of shipping orders last-minute.) In the following decade, more researchers have investigated the order fulfillment problem. Due to the curse of dimensionality, it is impossible to solve real-world order fulfillment problems to optimality. Therefore, researchers have proposed different heuristics (e.g., [1] and [16]), and some provide asymptotic or non-asymptotic performance guarantees.

For a finite-horizon order fulfillment problem, if we have perfect knowledge of demand arrivals and inventory positions over the finite time horizon, then we can determine the optimal assignment decisions by solving a deterministic linear program (LP); this LP matches the demand to the supply with the objective of minimizing the shipping cost. Motivated by the above, researchers have designed heuristics that leverage either the primal or dual solutions of this type of LPs and that incorporate demand forecasts, inventory positions and costs. Acimovic and Graves [1] propose a heuristic that assigns each order to the fulfillment center with the least "total" cost, consisting of the shipping cost plus the inventory opportunity cost. Acimovic and Graves [1] obtain the inventory opportunity costs from the dual values of item (SKU)-dependent LPs, i.e., the authors solve a stand-alone LP for each item, which reduces the size of the LPs. In addition, they periodically resolve the LPs to update the

opportunity costs. On the other hand, Jasin and Sinha [16] proposed a probabilistic heuristic where orders are assigned to fulfillment centers randomly according to the fractions derived from the primal solution of the LP that solves the demand-to-supply matching problem with all SKUs jointly. In high level, the more orders for a SKU that were assigned to an FC in primal solution, the higher is the probability of fulfilling an order for the SKU from the FC. For multi-item orders, consolidating orders (shipping multiple items from a one FC) could reduce costs. However, the heuristic decouples the fulfillment decision across different items in the same order by assigning each item independently according to the probabilities derived from the LP, which overlooks any opportunity for consolidation. To fix this shortcoming, the authors propose an improved heuristic that still uses the primal solutions from the LP but assigns items to the same FC if they are "positively correlated" according to a correlated rounding scheme. Building upon Jasin et al. [16]'s work, Ma [20] provides an improved correlated rounding scheme with a better guarantee to the optimal solution. Also based upon Jasin et al.'s work [16], Amil et al. [2] propose another heuristic that first selects the fulfillment center(s) for multi-item orders according to, again, probabilities derived from the LP. It then creates inventory thresholds to decide whether to actually fulfill the order, or not, with the selected fulfillment center(s). If the order is not fulfilled, then the order is being discarded (not fulfilled by the selected fulfillment center). Note that this second step in the heuristic is to bound the probability of running out inventories by the end of the time horizon. Their algorithm is shown to be asymptotically optimal and has guarantees in the non-asymptotic setting.

The above-mentioned heuristics are designed for networks where all FCs have the same function in the fulfillment process. In fact, other researchers have studied fulfillment policies in fulfillment networks, where there are different types of categories of FCs. Here, we describe two common network structures– the two-layer network and omnichannel.

The two-layer network is adopted by several monopoly retailers in China (e.g, JD.com and Alibaba). The network has two types of FCs (often referred to as distribution centers) in the two-layer network. One type is the downstream FCs that locate

closer to customers with limited capacity and that store high-velocity items. The other type is upstream FCs with larger capacities which replenish the downstream FCs and also fulfill customers' orders, particularly those orders for low velocity items. Zhao et al. [23] study a myopic fulfillment policy in the two-layer network and show that the policy performs well in this type of network through theoretical and numerical results.

The omnichannel network holds inventory in stores and fulfillment centers and serves two types of demands (online and in-store). The retail stores have a capacity to fulfill online orders, and a key challenge in this network is deciding what and how many online orders to fulfill from what retail stores. Andrew et al. [3] propose a primal-dual algorithm in the omnichannel setting. Their approach is particularly valuable when demand is very uncertain, and they have shown upper and lower bounds on the competitive ratio assuming adversarial demand.

So far, we have cited research that examines the inventory assignment decision for order fulfillment. However, in reality, order fulfillment decisions are intertwined with other problems such as inventory allocations and pricing problem. For example, the decisions of inventory allocation are important inputs to the order fulfillment problem, and the decision of order fulfillment and inventory allocation jointly affect the pricing decision. When trying to optimize the overall profit of a company, the consideration of these adjacent problems could provide additional values. Here, we list several papers that study the joint problem of order fulfillment and other aspects of the supply chain. Devalve et al. [10] study the value of fulfillment flexibility (the ability of fulfilling an order from different multiple FCs) in two-layer networks. They test multiple heuristics under different networks with different flexibility numerically and show that with the right choice of fulfillment policy, fulfillment flexibility may lead to non-negligible cost-savings. Lim et al. [19] and Govindarajan et al. [13] study joint inventory allocation and the order fulfillment problem. Lei et al. [17] and Harsha et al. [14] study joint pricing and the order fulfillment problem. Lei et al. [18] study the joint product display, ranking and pricing with the order fulfillment problem.

As the online retailing industry has grown, some retailers have explored new strategies for order fulfillment. One new strategy is for the retailer to invest in its own logistics network and resources. With this investment, the retailer can then handle some fraction of its order deliveries, and hence, rely less on third party carriers. This investment can entail setting up new types of facilities between fulfillment centers and customers for consolidation and sorting of shipments, as well as building transportation fleets to connect the facilities and to reach the customer. Currently, Amazon appears to deliver a significant portion of their packages by their own logistics system. The number of packages delivered by the Amazon logistics system has grown more than 5-fold between 2018 and 2022. [8]. Moreover, in 2020, Amazon surpassed FedEx, one of the largest delivery companies in U.S., in terms of number of packages delivered (Amazon's 4.2 billion compared to FedEx's 3.3 billion [15]). This change in strategy fundamentally changes the order fulfillment problem. For each order, the order fulfillment decision includes not only the choice of fulfillment center(s) from which to obtain the inventory for the order, but also the shipping route and carrier to transport the inventory units from the fulfillment center(s) to the customer.

In this research, we examine this new element of the order fulfillment problem. In particular, we decouple the order fulfillment problem into two decisions: the choice of fulfillment center and the choice of shipping route (or carrier). We assume that the inventory-assignment problem is solved, i.e., the choice of fulfillment center(s) is given for each order. We then focus our research effort on the transportation-assignment problem, i.e., the routing decision for the delivery between fulfillment centers and customers.

To show the challenges of the route selection, as well as the similarity and differences between the inventory-assignment and the transportation-assignment of the fulfillment problem, we provide an illustrative example. Consider the following network with 1 fulfillment center and 2 customer destinations – A and B (Figure 1-1). The fulfillment center can use a third-party carrier to ship orders to each customer destination. In addition, the retailer can use its own logistics resources to transport an order from the fulfillment center to a delivery station (DS), from which a last-mile

carrier can then transport orders to each customer destination.

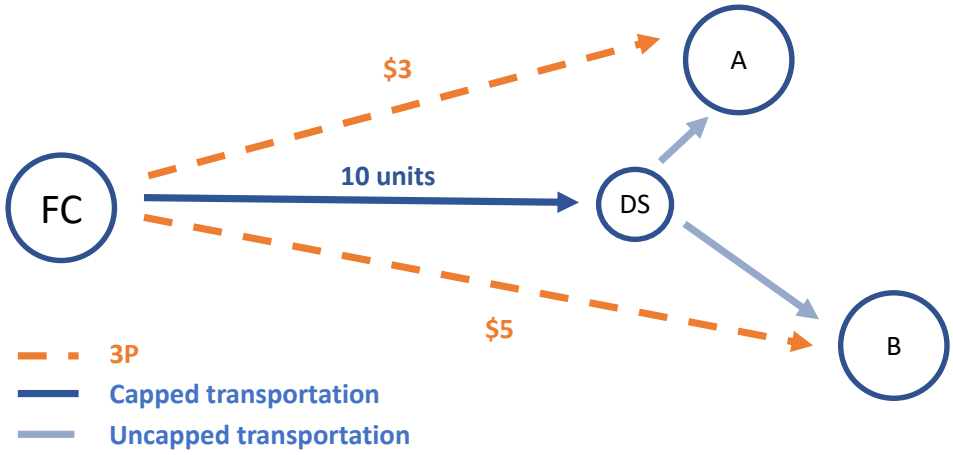


Figure 1-1: Network structure of the illustrative example.

Suppose that the single fulfillment center holds sufficient inventory for all items in the assortment, and the cost of last-mile delivery (from DS to customers) is negligible. When an order arrives, the online retailer needs to decide in real-time how to ship the order. The order can either be delivered by the retailer's transportation (with limited capacity of 10 units for a single time period) with zero variable cost, or delivered by third-party with a per-package cost of \$3 for destination A and \$5 for destination B. We observe that assigning the "expensive demand" (demand from destination B) to the retailer's transportation is always an optimal decision. The challenge lies in the decision for "cheap demand" (demand from destination A). If we assign too much cheap demand to the retailer's transportation, we might consume all the limited capacity before the end of the time period. We might then have a subsequent demand from destination B that then needs to be assigned to the third party. If we assign too little cheap demand to the retailer's transportation, we might risk wasting the limited capacity; that is, at the end of the time period we have unused capacity.

From this example, we observe similarities between the inventory-assignment and

the transportation- assignment problem. Both problems have limited resources, and when the order arrives, one needs to decide how to utilize these limited resources. For the inventory-assignment problem, the limited resources are inventories, while for the transportation- assignment problem, the limited resources are typically the truck capacity on a route or on a leg of a route. However, the two problems are also different in a couple of ways. First, the transportation resources are planned in advance and are perishable. For instance, the retailer may have planned for a truck to depart from a fulfillment center by a given time (10 AM) and travel to a delivery station. According to the plan, the truck departs at this time, whether it is full or not; hence, any unused capacity perishes. Second, at the time of the real-time decision, much of the cost of the retailer's transportation is sunk; for instance, in the above example, once a truck is scheduled for 10 AM, the retailer incurs a fixed cost for the truck to go from the fulfillment center to the delivery station, regardless of its load. Finally, the limited resource is shared across all items, and hence, only the volume and destination of orders is relevant.

In contrast, for the prior research on the inventory-assignment problem, the researchers assume that there is a known shipping cost (which might reflect third-party delivery costs) associated with fulfillment center decisions (ex: [1], [16], [20], [2], [23]). In chapter 2, we introduce the fulfillment problem in more detail with illustrative examples and study these examples from both theoretical and numerical standpoint.

In chapter 3, we provide a modeling framework for the transportation network. There are several components of the network design and operation that add complexity to the model. There are different types of transportation arcs and facility nodes run by the retailer on independent schedules (ex: truck departures, labor time shifts). In addition, part of the network relies on third-party carriers. For example, retailers could have a fixed or flexible contract with third-party companies that dictates from a given origin (retailer-owned facility node) to a destination (designated area of customers), a certain quantity (in terms of either package count, weight, or size) of shipments would be handled by a third-party company. In our work, for each order, one challenge is to determine all the feasible routes that could deliver the order from

an assigned FC to the customer destination by the order promise time. Each route is composed of a set of resources, regardless of mode and type as long as it matches a feasible pre-determined pattern, and the upstream and downstream resources are time compatible. Therefore, in real-time, making a route decision for an order entail choosing from a set of feasible routes. Note that we assume the mapping between route and resources are exogenous (pre-determined) which could be formulated based on historical data or domain knowledge in practice. For our numerical simulation, the mapping is created based on transit and dwell times, which are described in more detail in the chapter.

In chapter 4, we develop our quadratic program (QP) algorithm that makes real-time routing decisions. The QP algorithm is similar to the algorithm described in Acimovic and Graves [1], where opportunity costs are periodically updated in the background for each fulfillment options (FCs in the context of inventory fulfillment problem), and fulfillment decisions are based on these derived costs. However, Acimovic and Graves (2015) determine shadow prices for the inventory for a SKU at each FC, which can then be used to guide the inventory assignment decision. In contrast, we determine the shadow prices for the fulfillment resource, e.g, the transportation resources, which we will use to guide the route choice for each order.

We will compare our QP algorithm to an alternative which uses an LP to find the resource shadow prices. We discuss here some of our motivation for the QP algorithm relative to the LP algorithm. One advantage of the LP algorithm is that it is computationally efficient to solve, and therefore can be easily adopted in practice. However, the LP formulation itself has several weaknesses. First, it considers only the expected demand and ignores the uncertainties in the forecasts. Second, the shadow prices will be zero on any resource that has a primal solution less than the capacity (by complementary slackness), which leads to an on-off-switch-like cost structure. This motivates us to propose a different formulation that takes demand uncertainty into account implicitly in the parameters of the formulation, and for which the opportunity costs will increase gradually as the resource utilization comes closer to capacity. (More details can be found in Appendix A).

In chapter 5, we test the algorithm on a realistic example against different benchmarks, which include greedy (myopic) algorithm, LP algorithm (which mimics the policy in our collaborators' production system), and the hindsight solution. First, we run these algorithms on a self-contained subnetwork extracted from the actual transportation network of our collaborator. We aggregate historical demand arrivals to generate realistic test cases for the experiments. The experimental results suggest that the QP algorithm is able to reserve valuable and limited transportation resources for demands with expensive alternatives, which leads to lower fulfillment costs compared to the benchmarks. Next, we expand the network structure to a more complex network that captures important features in the actual network. We perform sensitivity tests on the network, where we observe the robustness of the QP algorithm under different scenarios.

Chapter 2

Introduction to the Order Fulfillment Problem through Small Examples

This research grew out of a partnership with a large American-based online retailer that sells a large catalog of physical items and operates its own network of fulfillment centers and transportation systems around the United States. Millions of orders are placed online each day and are promised to be delivered within a short period of time – often within a day or two. The online retailer needs to decide in real-time how to ship each order, which includes answering a series of questions. These include: from which warehouse should the order be shipped? What delivery route should the order take? And what transportation resources should the order consume along the way? Given the ever-increasing volume of incoming orders, minimizing fulfillment costs by making smart order fulfillment decisions at every order arrival instance becomes crucial to the business. In this thesis, we investigate these questions and develop models and algorithms that support this decision-making in real-time.

Over the last decade, the way the online retailer delivers orders has greatly changed, which expands the scope of the order fulfillment problem. The online retailer previously relied exclusively on third-party companies to transport and deliver its orders. These companies deliver orders from the online retailer's fulfillment centers (which store the inventory) to customers' doorsteps. In this setting, the order fulfillment problem focused mainly on determining from which fulfillment center should

the order be shipped? For a given fulfillment center, the order would then be assigned to the third-party carrier that could meet the service-time commitment at the least cost. Acimovic and Graves [1] addressed this order fulfillment problem with a heuristic algorithm, which was implemented and made millions of order fulfillment decisions daily. However, as the order volume continued to scale, the online retailer started to develop its own logistic capability for order fulfillment. In particular, the retailer invested in its own transportation fleet and facilities so as to handle orders from the fulfillment centers to its customers.

At this time, the online retailer operates a complex transportation network with hundreds of facilities (nodes) that are connected with transportation lanes (arcs). The retailer currently ships a large fraction of its orders with its own fulfillment and transportation systems and relies on third-party shippers for the remaining orders. This change in operational strategy changes the online order fulfillment problem. For each order, the order fulfillment decision includes both the choice of fulfillment center(s) from which to obtain the inventory for the order, plus the shipping routes and carriers to transport the inventory units from the fulfillment center(s) to the customer. In this research, we focus on the latter, and will assume that the choice of fulfillment center(s) is given for each order. We will then examine how to decide the route and the transportation resources to deliver each order from the fulfillment center to the customer's doorstep. Note that in reality, many orders may contain multiple items, and some of these multi-item orders will be split (or not) into multiple shipments from different fulfillment centers. As suggested above, we assume the decision to split an order is given for each order, along with the assignment of shipments to fulfillment centers. In addition, with this simplification, we use "order" and "shipment" interchangeably throughout the thesis.

2.1 Brief Introduction to the Transportation Network

For our research, we assume that the retailer's transportation network is composed of three types of facilities: Fulfillment Centers (FC), Sortation Centers (SCs) and Delivery Stations (DS). Each FC stores inventory and is often located in rural areas. Each order that is transported by the retailer's logistics system originates at an FC; most of the orders from the FC will go next to a SC, and the remainder will go directly to a DS. Each SC operates like a cross-dock facility and does not hold any inventory. The SC receives inbound orders from multiple FCs, and then sorts these orders based on destination; the sorted shipments are then "cross-docked" and loaded onto outbound trucks that are destined for DSs. Each DS is the terminal facility at which shipments are sorted based on the final destination; these DSs are typically located in urban areas that are close to customers. A last-mile carrier delivers the shipments from the DS to the customers. Every facility has its own work shifts and shipments can only be processed during those times. In addition, facilities are connected with transportation arcs or lanes, on which there is typically a transportation plan that determines the frequency of truck departures.

From FC to customers, shipments can be shipped solely by transportation resources that are controlled by the retailer, or solely by third-party companies, or by a mixture of both. On a typical retailer-controlled route, shipments travel from an FC to a SC, then from the SC to a DS, and finally from the DS to the customer. For some routes, there can be a direct shipment from the FC to a DS, skipping any SCs; occasionally a shipment may go through multiple SC's, for instance if it is a long route from one coast to the other. Third-party deliveries can originate from either an FC or SC and will always terminate at the customer's doorstep. Figure 2-1 illustrates the relationship between facilities.

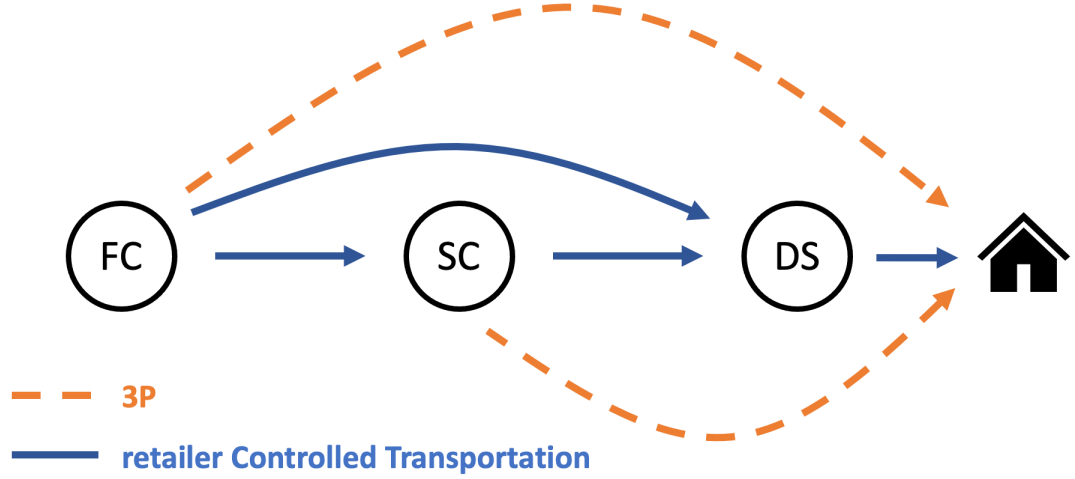


Figure 2-1: This figure shows the relationships between different facilities in the transportation system. For a given shipment from an FC to a customer, there are four feasible routes in this figure: third-party route that ship shipments directly from FC to customer, two retailer controlled routes (FC-SC-DS-customer, FC-DS-customer), and a mixed route where the first half (FC-SC) is shipped by retailer, and the second half is shipped by third-party (SC to customer).

Each element of a route can have limited capacity. For this research, we will express these capacity limits in terms of the maximum number of shipments that can be handled within a specified time window. For instance, the transportation plan may schedule three trucks to travel on an arc between noon and 7 pm; then the capacity on this arc for this time window corresponds to the capacity of these three trucks, which we will state in terms of number of shipments or packages. The time element of the problem contributes to the network complexity in a non-negligible way. We will provide a more detailed description in the next chapter of the time-related considerations and how we account for these in our models. In the next section, we show an illustrative example and use simple examples to demonstrate possible pitfalls of static policies.

2.2 A One-Link Network

Consider a simple network with one FC, one SC, two DSs. The nodes are connected by retailer-controlled transportation and by third-party transportation, as illustrated in Figure 2-2. We denote the current time as time zero, and we consider a time horizon that spans from time 0 to time T. Over this time horizon $[0, T]$, there is limited capacity between the FC and the SC, equal to $u(0)$ packages. We assume that the retailer has committed to this fixed amount of capacity in advance, and that the retailer cannot adjust this capacity within the time horizon $[0, T]$. Furthermore, we assume that the cost for this capacity is sunk, and that there is zero variable cost; for instance, if the capacity corresponds to three trucks on this lane in this time horizon, then we assume the retailer has already committed to paying the cost for these trucks regardless of their utilization. In addition, for simplicity, we assume that the transportation resources between SC and the two DSs are abundant and there is zero variable cost.

For every incoming shipment, the online retailer can route the shipment through its own transportation system with no cost, or ship with the third party from FC directly to customers with a per shipment cost of c_1 for customers served by DS 1 and c_2 for customers served by DS 2. In addition, $c_1 < c_2$. We will make assignment decisions for a fixed time horizon from time 0 to time T. The demand arrival is unknown, and the objective of the online retailer is to make the best use of their limited transportation resource to minimize outbound shipping costs while satisfying the demand. In this example, if the demand arrival between time 0 to time T is more than the capacity of the limited resource, then in hindsight, the online retailer should prioritize the limited free transportation resource to demands with the larger third-party cost (i.e., customers from DS 2). Let us start with introducing some notation:

- $u(t)$: remaining capacity of the shared resource (transportation arc between FC and SC) at time t for $0 < t \leq T$
- $d_i(t)$: actual demand that arrives from DS i in the time interval $(t, T]$

- $D_i(t)$: a random variable for the demand that arrives from DS i in the time interval $(t, T]$

- $\hat{d}_i(t)$: a forecast of the demand that arrives from DS i in the time interval $(t, T]$, and will typically be the expectation of $D_i(t)$

- x_i: the amount of demand from DS i assigned to the retailer-controlled route in $[0, T]$.

- y_i: the amount of demand from DS i shipped by third-party in $[0, T]$.

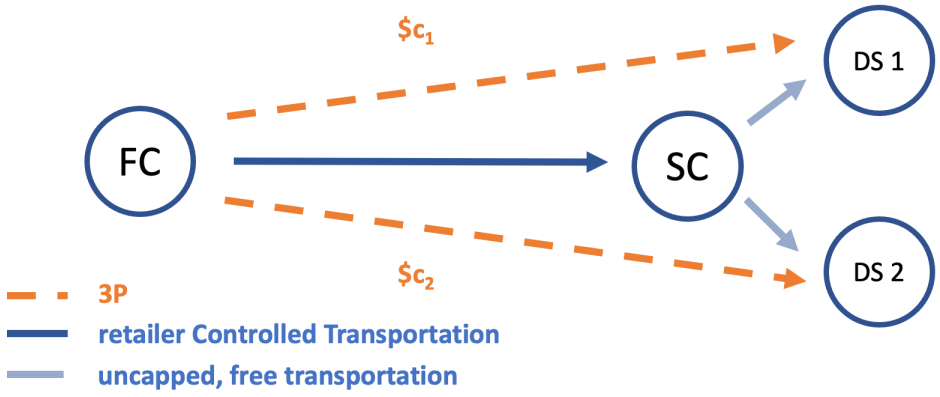


Figure 2-2: The one-link network

We now formulate the hindsight problem where we know the initial capacity $(u(0))$ and we assume that we are given the actual demand $(d_i(0))$ as inputs, and x_i and y_i are decision variables. We obtain the optimal hindsight solution by solving the

following LP:

$$\min \quad \sum_{i=1,2} c_i y_i \tag{2.1a}$$

$$\sum_{i=1,2} x_i \leq u(0) \tag{2.1b}$$

$$x_i + y_i \geq d_i(0) \quad \forall i = 1, 2 \tag{2.1c}$$

$$\mathbf{x}, \mathbf{y} \geq 0 \tag{2.1d}$$

In fact, in this one-link example, we can express the optimal hindsight solution explicitly based on demand and capacity:

$$\begin{aligned}
x_1 &= \min(u(0) - \min(u(0), d_2(0)), d_1(0)) \\
x_2 &= \min(u(0), d_2(0)) \\
y_1 &= d_1(0) - \min(u(0) - \min(u(0), d_2(0)), d_1(0)) \\
y_2 &= d_2(0) - \min(u(0), d_2(0))
\end{aligned}$$

The solution depends on what the demands are. If the total demand arrival from DS 1 and DS 2 does not exceed $u(0)$, i.e., $d_1(0) + d_2(0) \leq u(0)$, then there will be no third-party assignments for both destinations, i.e. $y_1 = y_2 = 0$. For example, if $u(0) = 100$, $d_1(0) = 40$, $d_2(0) = 50$, then $y_1 = y_2 = 0$ and $x_1 = 40$, $x_2 = 50$. If the total demand arrival from DS 1 and 2 exceeds $u(0)$, then the hindsight solution will prioritize the transportation resource to DS 2 demand. For example, if $u(0) = 100$, $d_1(0) = 50$, $d_2(0) = 70$, then $x_2 = 70$, $x_1 = 30$, $y_2 = 0$ and $y_1 = 20$. In the following subsections, we analyze several static policies, and compare them to dynamic policies.

2.2.1 Static Policies

In this section, we study three static policies – greedy policy, conservative policy and threshold-based policy.

The greedy policy assigns every incoming shipment to the cheapest available

route. It assigns the first $u(0)$ shipments to the retailer-controlled transportation since retailer-controlled transportation is cheaper than third-party routes regardless of demand type, and assigns the remaining shipments to third-party routes since there is no more capacity on FC-SC arc. Among all possible demand arrivals (for any number of shipments in any arrival sequence), the ratio of the cost of the greedy policy to the hindsight solution could be as large as c_2/c_1. This worse case would happen when the first $u(0)$ demands come from DS 1, and the remaining $N - u(0)$ demands come from DS 2, where N denotes the total number of demand arrivals, and we assume $N > u(0)$. In this setting, the greedy cost is $c_2(N - u(0))$, the hindsight cost is $c_1 \min(u(0), N - u(0)) + c_2 \max(0, N - 2u(0))$. The competitive ratio reaches maximum when $u(0) < N \leq 2u(0)$:

$$\max_N \frac{c_2(N - u(0))}{c_1 \min(u(0), N - u(0)) + c_2 \max(0, N - 2u(0))} = \frac{c_2}{c_1}$$

Note that the competitive ratio of an online algorithm is simply the approximation ratio achieved by the algorithm, that is, the worst-case ratio between the cost of the solution found by the algorithm and the cost of an optimal solution.

Next, we look at another static policy. The conservative policy reserves the shared resource exclusively for the more valuable shipments, namely those that are more costly to assign to the third party. In this one-link model, it assigns only shipments from DS 2 to the retailer-controlled route. In the worst-case scenario, there are $u(0)$ shipments, and they are all from DS 1. The cost of the conservative policy is, $c_1 u(0)$ while the cost of the optimal solution is 0. The maximum competitive ratio of the conservative algorithm is infinity.

We note that in the case where the FC-SC arc has a non-zero variable cost, the maximum competitive ratio of the static policies changes accordingly. Suppose that the FC-SC arc charges c dollars per package with $0 < c < c_1 < c_2$. Then the maximum competitive ratio of the greedy policy becomes $(c_2 + c)/(c_1 + c)$, while the maximum competitive ratio of the conservative policy becomes $(c_1 + c)/c$.

Finally, we introduce the threshold-based policy that is a hybrid of the greedy and

conservative policy. The threshold-based policy limits the utilization of the shared resource to at most α shipments to the cheaper destination. That is, in this example, the threshold-based policy assigns all shipments to the shared resource, until the number of shipments assigned to DS 1 reaches the threshold, or until the capacity is reached, whichever comes first. If the threshold occurs first, then the policy assigns only DS 2 demand to the shared resource, again until the capacity limit is reached.

It is easy to show that regardless of any α between 0 and $u(0)$, the maximum competitive ratio of the threshold-based policy algorithm is infinity. This happens when the total demand that arrives between $[0, T]$ is less than the initial capacity ($\sum_{i=1,2} d_i(0) < u(0)$) and the total demand from DS 1 (the cheaper destination) that arrives between $[0, T]$ is higher than the threshold ($d_1(0) > \alpha$). In this setting, the cost of the hindsight solution is zero, as there is sufficient capacity to handle all the demand; however, the cost of the threshold-based policy is non-zero since some demand for DS 1 gets assigned to the third party. Hence, this results in an infinite competitive ratio.

The competitive ratios indicate that these static policies perform poorly in worst-case scenarios. In addition to a worst-case analysis, we can also consider the expected regret for these policies. Given the demand distributions, we may be able to compute the expected regret, which is the expected additional cost paid by a policy in comparison to the hindsight solution. The expected regret reflects the probability of making wrong decisions (decisions that in hindsight would be made differently) and the associated costs. We will consider the greedy policy as an example. The greedy policy assigns every shipment to the shared resource until the shared resource is out of capacity. In this simple example, the assignment of demand from DS 2 is always a right decision in hindsight, while the assignment of demand from DS 1 at any time t would be a wrong decision if the remaining capacity at time t is less than the remaining demand from DS 2. In addition, the cost ("regret") of making a mistake is the third-party cost difference $c_2 - c_1$. The expected regret over a horizon of interest (T) can be expressed explicitly in terms of these parameters. For example, suppose that the aggregate demand arrival process is a Poisson process with rate μ, and each

demand comes from DS 1 (DS 2) with probability p $(1-p)$. We denote the number of shipments that have arrived before time t from DS 1 (DS 2) by $\bar{D}_1(t)$ ($\bar{D}_2(t)$), and F_X denotes the CDF of random variable X. We can express the expected regret as

$$\begin{aligned}
\mathbb{E}[\text{regret}] =& \int_{t\in T} p(c_2-c_1)\mathbb{P}[u(t) \le D_2(t)]\mu dt \\
=& p(c_2-c_1)\mu \int_{t\in T} \mathbb{P}[u(0) - (\bar{D}_1(t)+\bar{D}_2(t)) \le D_2(t)]dt \\
=& p(c_2-c_1)\mu \int_{t\in T} \mathbb{P}[D_2(0)+\bar{D}_1(t) \ge u(0)]dt \\
=& p(c_2-c_1)\mu \int_{t\in T} (1-F_{D_2(0)+\bar{D}_1(t)}(u(0)))dt \\
=& p(c_2-c_1)T - p(c_2-c_1)\mu \int_{t=0}^{T} e^{-(pt\mu+(1-p)T\mu)} \sum_{i=0}^{u(0)} \frac{(pt\mu+(1-p)T\mu)^i}{i!}dt \\
=& p(c_2-c_1)T - (c_2-c_1)\sum_{i=0}^{u(0)} \frac{1}{i!}(-\Gamma(i+1,T\mu)+\Gamma(i+1,(1-p)\mu T))
\end{aligned}$$

where Γ is the lower incomplete gamma function. In the second equality, we replace $u(t)$ by the initial capacity minus the demand that arrives before time t. In the fourth equality, since $D_2(0)$ and $\bar{D}_1(t)$ are independent Poisson processes, the sum of these two random variables is a Poisson random variable with mean $pt\mu+(1-p)T\mu$. The fifth equality is by definition of Poisson CDF:

$$F_{D_2(0)+\bar{D}_1(t)}(u(0)) = e^{-(pt\mu+(1-p)T\mu)} \sum_{i=0}^{u(0)} \frac{(pt\mu+(1-p)T\mu)^i}{i!}.$$

The last equality is by definition of the upper incomplete gamma function, where we derive this abbreviated expression by change of variables [12].

This analysis can be readily applied to the conservative policy. The conservative policy assigns only demand from DS 2 to the shared resource until it is out of capacity. In this example, the assignment of demand from DS 2 to the shared resource is always a right decision in hindsight, while the assignment of demand from DS 1 to a third party at any time t would be a wrong decision if the remaining capacity at time t is

greater than the remaining demand from both DSs. In addition, the cost ("regret") of making a mistake is the third-party cost of DS 1 c_1. Therefore, the expected regret can be derived from:

$$\mathbb{E}[\text{regret}] = \int_{t \in T} pc_1 \mathbb{P}[u(t) \geq D_1(t) + D_2(t)] \mu dt$$

One can generate an explicit expression in similar fashion to that for the greedy policy. Similarly, the analysis can be applied to the threshold based controller, which is essentially a policy that switches from greedy to conservative.

2.2.2 Dynamic Policies

In this section, we introduce two dynamic heuristic policies. We note that for this simple network, we can find the optimal dynamic policy with a dynamic program, which we will describe later. The first heuristic policy (which we name the "LP algorithm") is inspired by the algorithm that is currently adopted by the online retailer. The second heuristic policy (which we name the "QP algorithm") is a simplified version of the algorithm we develop further in the thesis.

2.2.2.1 The LP Algorithm (for the One-link Network)

The LP algorithm derives a shadow price of the shared resource (FC-SC arc) at every shipment arrival; then it makes routing decisions with the shadow price by comparing the shadow price of the FC-SC arc to the alternative third-party cost, and picking the cheaper option to ship the shipment. If there is a tie in the costs, we prioritize the retailer-controlled routes over third-party routes.

At every shipment arrival time t, the shadow price is derived from a linear programming problem that finds the optimal solution that minimizes the transportation costs from time t until the end of the horizon. The linear programming formulation

is as follows:

$$\min \quad \sum_{i=1,2} c_i y_i \tag{2.2a}$$

$$\sum_{i=1,2} x_i \leq u(t) \tag{2.2b}$$

$$x_i + y_i \geq \hat{d}_i(t) \quad \forall i = 1, 2 \tag{2.2c}$$

$$x_i, y_i \geq 0 \quad \forall i = 1, 2 \tag{2.2d}$$

where $\hat{d}_i(t)$ is the forecast of demand between time t to time T for DS i; $u(t)$ is the remaining capacity prior to the assignment of the shipment that arrives at time t; the decision variables x_i and y_i are the number of shipments from DS i assigned to shared resource and third-party carriers, respectively. The objective is to minimize the transportation costs. The first constraint is the capacity constraint; the second constraint is the demand constraint. The shadow price of the shared resource is the dual variable of the capacity constraint.

At a high-level, the algorithm produces a large shadow price when demand forecasts are relatively high, and produces a small shadow price when demand forecasts are relatively low. In particular, when the capacity constraint is not tight, by complementary slackness, the shadow price will be zero; when $\hat{d}_2(t) > u(t)$, the shadow price will be c_2; when $\hat{d}_1(t) + \hat{d}_2(t) > u(t)$ and $\hat{d}_2(t) < u(t)$, the shadow price will be c_1.

To understand the performance of the LP algorithm compared with the optimal hindsight solution, we simulate the algorithm on this two-demand-one-resource network over a one-day horizon with $T = 24$hr. The experiment is based on the following assumptions:

- We assume that demand arrival is a Poisson process with rate μ, where for each experiment we set μ by performing a random draw from a normal distribution $N(u(0), \sqrt{u(0)})$. In addition, each demand arrival has probability p from DS 1 and probability $1 - p$ from DS 2.

- At time $t = 0$, the initial demand forecast of the total demand (including the two demand types) is set to the initial capacity of FC-SC arc, i.e. $\hat{d}(0) = u(0)$. For other times t, where $0 < t \leq T$, we update the forecast for the entire day by summing the shipments that arrived before time t ($\bar{d}(t)$) to the initial forecast discounted by $\frac{T-t}{T}$, which corresponds to the initial forecast for the remainder of the day. Then, we multiply this revised demand forecast for the entire day by $\frac{T-t}{T}$ to get an updated forecast for the remainder of the day. In summary, the demand forecast for the remainder of the day at any time $t > 0$ is:

$$\hat{d}(t) = \frac{T-t}{T}(\frac{T-t}{T}\hat{d}(0) + \bar{d}(t))$$

- The demand forecasts for the two demand types (demand from DS 1 and DS 2) for the remainder of the day at time t are:

$$\hat{d}_1(t) = p\hat{d}(t)$$
$$\hat{d}_2(t) = (1-p)\hat{d}(t)$$

Experimental Procedure:

1. Initialize μ with random draw from $N(u(0), \sqrt{u(0)})$

2. Initialize time index at $t = 0$

3. We generate the next demand arrival. Since the demand arrivals are Poisson, the time between two demands are generated by an exponential distribution with mean $1/\mu$, and the demand type is determined randomly with probability p from DS 1 and probability $1 - p$ from DS 2. Let τ be the exponentially distributed interarrival time; then we update the time index $t = t + \tau$

4. We assign the shipment that arrives at time t. For each shipment arrival, we make an assignment decision based on its destination and on the shadow price derived from the LP with the most updated states. That is, the shipment

is assigned to the third-party route if the third-party cost of the shipment is cheaper than the shadow price; the shipment is assigned to the shared FC-SC route if the third-party cost is more expensive. If the third-party cost equals the shadow price, we prioritize retailer-controlled routes over third-party routes.

5. After assigning the shipment we update the states, including the forecast, and the remaining capacity

6. We repeat step 3, 4, 5 until the shipment generated arrives after T

We simulate 100 realizations with $u(0) = 100$, μ is drawn randomly from a normal distribution $N(100, 10)$, $p = 0.5$, $c_1 = 1$, $c_2 = 2$, $T = 24$ hours.

The average cost of the 100 realizations when decisions are made by LP algorithm is 8.74 (with 8.26 standard deviation), while the average cost of the hindsight solution is 4.94 (with 8.17 standard deviation). Figure 2-3 shows the cost of the 100 test cases with respect to the total demand. We note that the hindsight cost is zero when the number of shipments ($d(0)$) is less than or equal to the initial capacity, i.e. $d(0) < 100$, and when $d(0) > 100$, the hindsight cost is always $c_1(d(0) - 100)$ (where $c_1 = 1$) unless the demand from DS 2 is more than 100, which did not occur in any of the 100 realizations.

As noted earlier, the shadow price of the LP algorithm can be one of the three possible values – 0, c_1 or c_2. Figure 2-4 shows the shadow price progression over time of a random test case, where the shadow price is either 0 or c_1. When demand from DS 2 is more than the initial capacity, the shadow price can also be c_2. However, for the assumed demand distribution for the test case, this is very rare, and in most cases, the shadow price is either 0 or c_1.

2.2.2.2 The QP Algorithm (for the One-link Network)

The QP algorithm also derives a shadow price of the shared resource (FC-SC arc) at every shipment arrival, but now the shadow price is derived from a quadratic programming problem. To formulate a quadratic program (QP), we relax the capacity

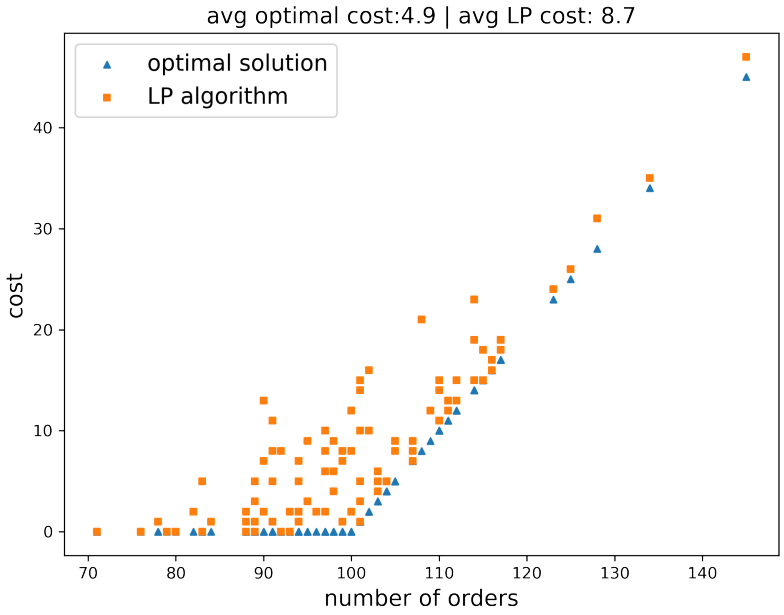


Figure 2-3: Performance of the LP algorithm versus the hindsight solution

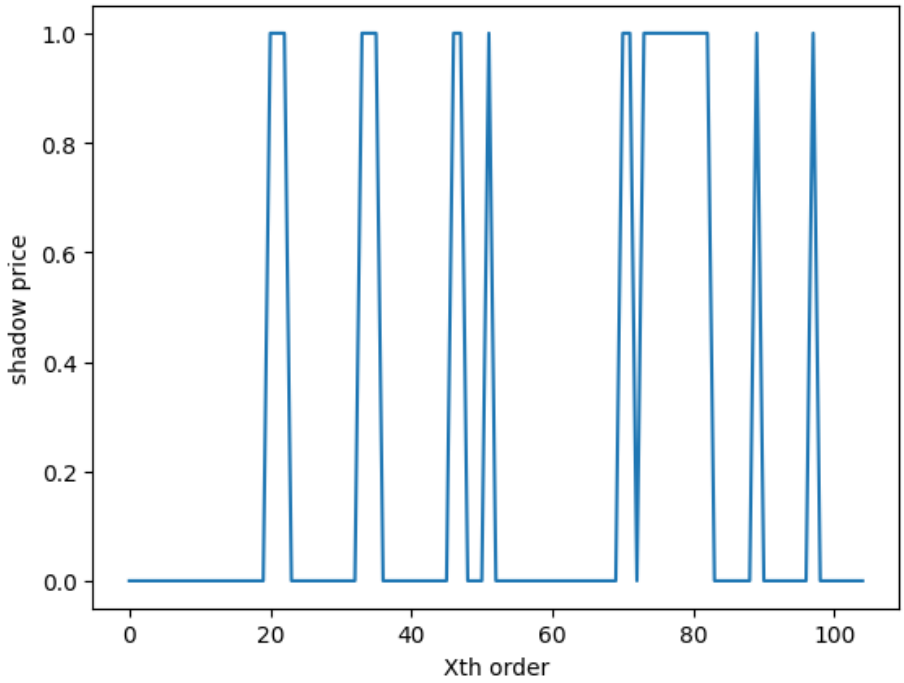


Figure 2-4: Progression of shadow prices from the LP algorithm overtime

constraint, and then set the objective function as proportional to a quadratic penalty

of the flows that exceed a specified flow target ($\bar{f}(t)$):

$$\min \quad \sum_{i=1,2} \frac{1}{2} v(t) g^2 \tag{2.3a}$$

$$\sum_{i=1,2} x_i - g \leq \bar{f}(t) \tag{2.3b}$$

$$x_i \geq \hat{d}_i(t) \quad \forall i = 1, 2 \tag{2.3c}$$

$$\mathbf{x}, \mathbf{g} \geq 0 \tag{2.3d}$$

In the QP, $v(t)$ is a penalty coefficient, $\bar{f}(t)$ is the chosen "flow target", and g denotes the "excess flow" over the target. The flow target ($\bar{f}(t)$) represents how much we should plan to load onto FC-SC arc over the remainder of the time horizon, in light of the demand uncertainty and the relevant costs for third-party option. We expect to set it to be less than the remaining capacity $u(t)$, where the difference acts as a safety buffer. We set the penalty coefficient ($v(t)$) to approximate the expected remaining cost at any time t. We will defer the explanation for how to set and justify the targets and penalty coefficient until chapter 4. The optimal solution of the QP for this one-link example is trivial, where the optimal solution can be expressed explicitly in terms of the capacity and forecasts:

$$x_i = \hat{d}_i(t)$$
$$g = (\sum_{i=1,2} \hat{d}_i(t) - \bar{f}(t))^+$$

By the KKT (Karush–Kuhn–Tucker) condition, the dual variable associated with the FC-SC resource constraint can be expressed explicitly by:

$$\lambda(t) = v(t) g$$

We then set the shadow price on the retailer-controlled route by capping the dual by the maximum third-party cost, i.e.,

$$\min(\max(c_1, c_2), \lambda(t))$$

We make this adjustment to avoid the algorithm from "closing" the retailer-controlled route entirely to all demand type when the dual variable of the shared resource is higher than the most expensive alternative third-party cost. We will explain this adjustment in more detail in the next chapter, where we develop the algorithm. When we encounter a tie in the costs, we prioritize the retailer-controlled routes over third-party routes.

We simulate the QP algorithm on the same 100 test cases by setting target and the penalty coefficient by the following formula:

$$\begin{aligned} \bar{f}(t) &= u(t) - k_1\sqrt{\hat{d}(t)}, \\ v(t) &= k_2 \min(c_1, c_2)\frac{1}{\sqrt{\hat{d}(t)}}, \end{aligned}$$

where k_1 and k_2 are constants set to be 2 and 0.2, respectively, in our simulation. We note that the algorithm's performance depends on the choice of these coefficients. We will explain and justify this setting in the next chapter. We find that the results are much closer to the hindsight solution than the LP algorithm. The average cost of the 100 realizations with the QP algorithm is 6.4 (with 9.7 standard deviation), which is 29.6 percent higher than the average cost of the hindsight solutions 4.94 (with 8.17 standard deviation). In the 100 test cases, the QP algorithm outperforms the LP algorithm in 58 cases, the LP algorithm outperforms QP in 26 cases, and the two algorithms incur the same cost in the remaining 16 cases. Figure 2-5 shows the cost of the 100 test cases with respect to the total number of demand: Figure 2-6 shows the shadow price progression over time of a random test case, where the shadow price structure is more continuous. In this simple example, the QP algorithm performs better than the LP algorithm on average due to its flexible cost structure.

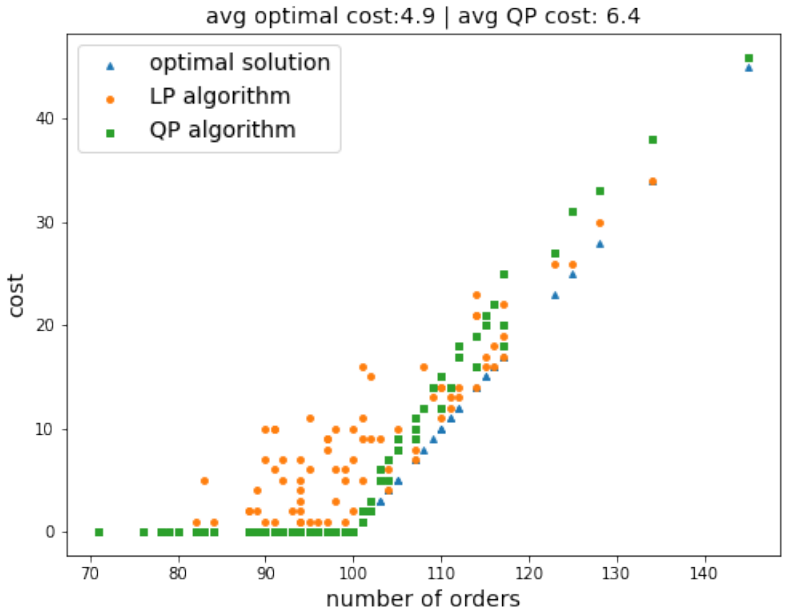


Figure 2-5: Performance of the QP algorithm versus LP algorithm and the hindsight solution

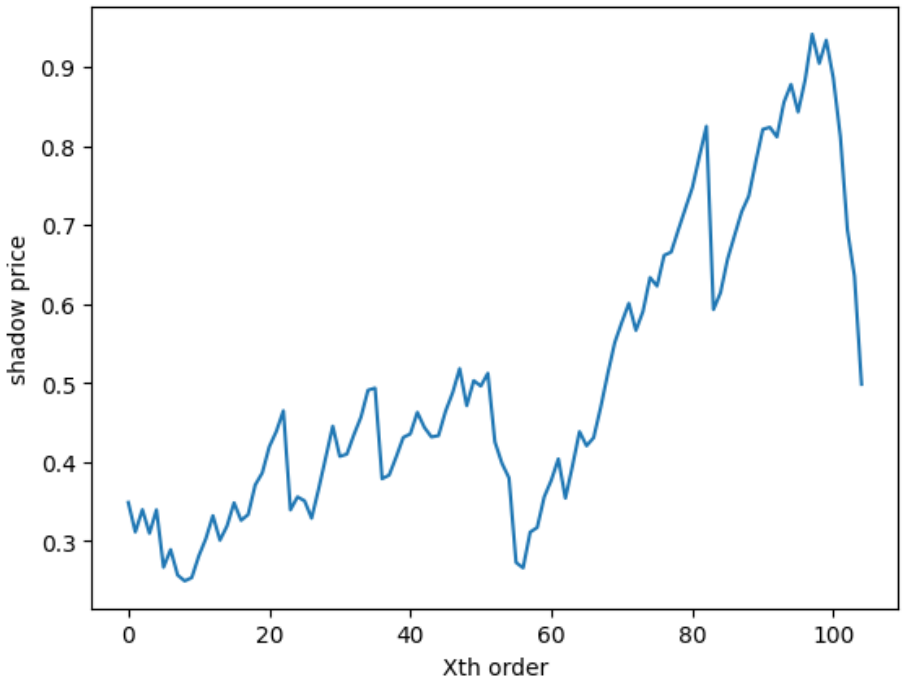


Figure 2-6: Progression of shadow prices from the QP algorithm overtime

In the regime where $\hat{d}_1(t) + \hat{d}_2(t) > u(t)$ and $\hat{d}_2(t) < u(t)$, the shadow price of the LP algorithm can only be c_1, while the QP algorithm could produce higher, equal or lower shadow prices than c_1.

To compare the dynamic policy against the static policy, we also run the same 100 test cases on the two static policies – greedy and conservative. The average cost of the greedy algorithm is 13.91, while the average cost of the conservative algorithm is 50.53. Both are much worse than the dynamic policies. In a later section (section 2.5), we introduce another benchmark where we model the problem as a dynamic programming problem.

2.3 Ad Hoc Truck Options

In the prior case, we assumed that the capacity on the retailer-controlled link is set some time ahead (e.g., a week or two ahead) and can not be changed. This reflects the fact that some number of trucks was scheduled in some earlier time frame and can not be changed. However, in addition to truck schedules set in advance, the online retailer can sometimes add additional trucks during the day with short notice (e.g, within an hour or two, an ad hoc truck can arrive at the facility as an addition to the existing truck schedule) to some transportation link. These trucks are more expensive than trucks scheduled a week ahead, but may be cheaper than third parties if these trucks can be fully-utilized. Therefore, it may be economical for the retailer to add capacity, by calling ad hoc trucks, when facing higher than expected demand. In this section, we assume that the retailer can add ad hoc trucks to the online-retailer controlled link (the FC-SC arc) at any time. We propose two ad hoc truck controllers (heuristics) that decide when and where to add or remove ad hoc trucks to the system. The controllers are triggered whenever the retailer wants to revisit the quantity of scheduled transportation resource.

2.3.1 Threshold-Based Controller

We assume that an additional truck on the FC-SC arc provides a capacity of u^{ah} units and costs c^{ah}. The cost for adding a truck (which we call an ad hoc truck) is greater than the cost for planned trucks, which are scheduled in advance. Nevertheless, when demand within a time period or day exceeds the forecast, adding an ad hoc truck may be justified.

We propose to trigger these ad hoc truck decisions with a threshold that reflects the cost of shipping excess flows by third-party. If the cost of adding an ad hoc truck is cheaper than the projected third-party cost, then we add an ad hoc truck. At any time t we add an ad hoc truck to the arc if the following inequality holds:

$$c^{3p}(f_j(t) - u(t))^+ > c^{\text{ah}}$$

where c^{3p} denotes the alternative (third-party) cost, $f_j(t)$ denotes the current forecast of expected remaining flows on the arc j at time t, $u(t)$ denotes the remaining capacity at time t. The left-hand side of the inequality is the projected cost if we do not add any more capacity on this arc. We note here that the specification for c^{3p} is not obvious, but should depend upon both c_1 and c_2; we will explore numerically various ways for setting c^{3p} in section 2.3.3. The right-hand side is the cost for adding a truck, where we implicitly assume the truck will have sufficient capacity to handle the remaining demand. Hence, the rule is to add a truck if its cost is less than the projected cost for doing nothing.

We can also consider the possibility of canceling a scheduled ad hoc truck. We might remove a truck when the expected cost of shipping the remaining demand with capacity without the ad hoc truck is less than the ad hoc truck cost. In other words, we remove a truck on the arc at time t when the following inequality holds:

$$c^{3p}(f(t) - (u(t) - u^{\text{ah}}))^+ < c^{\text{ah}}.$$

Note that we allow a cancellation of an ad hoc truck only when no shipments have

been assigned to the truck.

2.3.2 DP-Based Controller

We describe here a second heuristic to determine when and where to add or cancel ad hoc trucks in a transportation network. At any time instance t, we can add an ad hoc truck to the FC-SC arc, or remove an ad hoc truck, or do nothing. Therefore, there are 3 possible actions at any time instance. We formulate the ad hoc truck problem as a dynamic program that minimizes the immediate cost of the ad-hoc decision plus the resulting future expected cost:

$$J(S,t) = \min_{a\in\{-1,0,1\}} \{c(S,a) + J(f(S,a),t)\} \tag{2.4}$$

where t is the time of decision instance, S is the system state (which includes remaining capacity and a forecast of the remaining demand), a is the ad hoc truck decision, $c()$ is the cost of action a at state S, $f()$ defines how state S evolves from a given action a, $\{-1,0,1\}$ denotes the three possible actions (i.e., removing a truck, doing nothing or adding a truck). We propose to solve an approximation of the DP by approximating the cost-to-go function ($J(S,t)$) by the objective function $W(S,t)$ of a linear programming problem that minimizes shipping costs while satisfying demand and capacity constraints. For the one-link example, the linear program is the following:

$$W(f(S,a),t) = \min \quad \sum_{i=1,2} c_i y_i \tag{2.5a}$$

$$\sum_{i=1,2} x_i \leq u(t) + a u^{ah} \tag{2.5b}$$

$$x_i + y_i \geq \hat{d}_i(t) \quad \forall i = 1,2 \tag{2.5c}$$

$$x_i, y_i \geq 0 \quad \forall i = 1,2 \tag{2.5d}$$

where u^{ah} is the capacity of an ad hoc truck, a is either -1, 0 or 1. We solve the problem for $a = -1$, $a = 0$ and $a = 1$ to approximate the cost-to-go function for these

actions, and then apply the solution to equation 4.22 and make the ad hoc decision by picking the cheapest option.

2.3.3 One-Link Network with Ad Hoc Truck Options

In this section, we simulate the QP algorithm on the one-link model with the two ad-hoc controllers and compare the results with the hindsight solution. We assume that ad hoc trucks can be added to the FC-SC arc. Each ad hoc truck is 10 dollars (c^{ah}) with capacity of 30 (u^{ah}) shipments. For the threshold based control, we set c^{3p} to be either minimum of the third-party costs ($c^{3p} = \min(c_1, c_2)$), the average of the third-party costs ($c^{3p} = \frac{c_1+c_2}{2}$) or maximum of the third-party costs ($c^{3p} = \max(c_1, c_2)$). For the threshold-based controller, we set the expected flow ($f(t)$) of the FC-SC arc by:

$$f(t) = \sum_{i=1,2} \hat{d}_i(t)$$

We simulate the algorithms on the same 100 instances as described in section 2.2.2. The performance averaged over the 100 instances is listed in table 2.1. We observe that the threshold-based controller performs the best when we set the coefficient to the minimum of third-party costs. Interestingly, the number of ad hoc trucks added by the three different threshold-based controllers are the same, but the timing of when they add the trucks are different.

In hindsight, the online retailer could spend less money on average when there is an ad hoc truck option. The hindsight solution adds zero ad hoc trucks in 86 cases, and one truck in 14 cases; both DP-based and threshold-based controllers (when c^{3p} are set to the minimum of third-party costs) add zero ad hoc trucks in 90 cases, and one truck in 10 cases. In fact, the ad hoc truck decisions made by the two controllers are exactly the same for the 100 instances. This is due to the fact that the threshold-based controller approximates the optimal solution of the LP formulation of the DP-based controller, presumably due to simplicity of the network and the parameter choice. In the next section, we perform the same experiment on a more complicated network,

where we can observe differences between the two controllers.

			QP algorithm	hindsight cost
without ad hoc truck option			6.4 (±9.7)	4.94 (±8.17)
with ad hoc truck option	threshold-based controller	max 3p	7.03 (±4.91)	3.38 (±4.59)
		mean 3p	6.54 (±5.03)	
		min 3p	5.79 (±5.12)	
	DP-based controller		5.79 (±5.12)	

Table 2.1: Average costs of different ad hoc controllers (the numbers in the parentheses are standard deviations).

2.4 A Two-link Network

In reality, the network structure is more complex than the one-link example. Routes are often composed of multiple arcs and nodes (resources), and these resources are often shared by multiple routes. See Figure 2-7 for an illustration of a more complicated transportation network that is representative of reality. If we zoom-in to a FC to SC to DS network (highlighted by red in Figure 2-7), the arc from the chosen FC to the chosen SC will have flows to other DSs, and the arc from the chosen SC to the chosen DS will have other flows originating from other FCs. Eventually we will look at this more complicated network, but for now, to build some intuition and insight, we focus on this two-link reduced network (highlighted by red in Figure 2-7) that incorporates this complexity by modeling not only the flows that are fully fulfilled by the network, but also the flows that are partially fulfilled by the reduced network. Consider a transportation network with 1 FC, 1 SC, 1 DS and two service areas. The nodes are connected by retailer-controlled transportation and third-party transportation as illustrated in Figure 2-8.

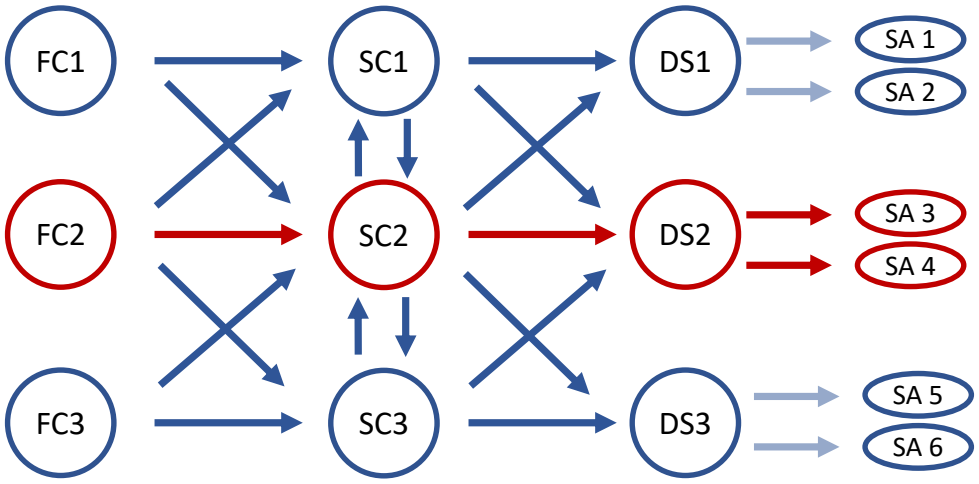


Figure 2-7: Illustration of a complicated network

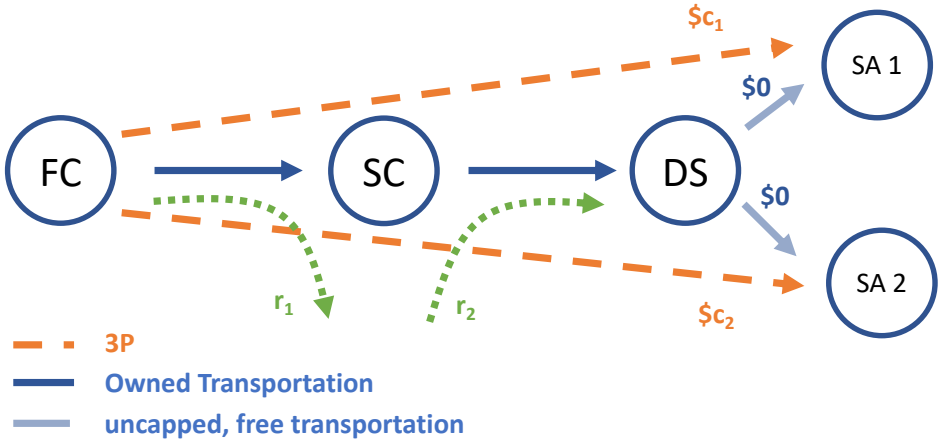


Figure 2-8: Illustration of the two link network. For the internal demand (from service area 1 and service area 2), there are two routes: a third-party route that ships directly from the FC to the customer, a retailer-controlled route from FC to SC to DS to the customer. External demands (illustrated with green lines) on FC-SC (SC-DS) consumes the FC-SC (SC-DS) resource controlled by the retailer.

There are two types of shipments that are handled by this network – "internal

shipment" and "exogenous shipment". Internal shipments are shipments that originate from the FC and that are destined to customers either in service area 1 or in service area 2 (Figure 2-8); these shipments can only be fulfilled by either the online retailer's transportation route or by third-party companies with a cost of c_1 for demand from service area 1 and c_2 for demand from service area 2. The online retailer's transportation route starts at the FC, goes to the SC, and then to the DS; from the DS, the shipment is delivered by a last-mile carrier to a customer at one of the service areas. In addition, we assume that the two retailer-controlled arcs serve other demand, which we refer to as "exogenous shipments". The exogenous shipments on the FC-SC arc are shipments that originate from the FC and transported first to the SC, and then go from the SC to other DSs for delivery to other service areas. Similarly, we have other exogenous shipments on the SC-DS arc. These represent shipments that come from other FCs to the SC, are then shipped to the DS, and will go to customers in service area 1 or service area 2. For this analysis, we follow the same assumptions as the one-link model with the following additions:

- We model the external demand on the upper (FC-SC) and the lower (SC-DS) as two independent Poisson processes with rate γ_1 and γ_2, respectively.

- The exogenous demand, by default, will be shipped by the retailer-controlled resources. If the retailer-controlled route is out of capacity, then the excess exogenous shipments will be charged a cost (c^{ex}) per shipment.

- We assume that all retailer–controlled arc and node costs are sunk costs that have been paid, and incur no variable costs. In addition, last-mile delivery from DS has zero cost. (Alternatively, we can interpret the third-party cost of c_1 for demand from service area 1 and c_2 for demand from service area 2 to be the actual cost net of the last-mile delivery cost from the DS)

We will illustrate the performance of the LP algorithm and the QP algorithm on this network. We first describe the experimental setup, and then how we implement each algorithm. We then provide results, with a comparison to the hindsight solution.

2.4.1 Experimental Setups

For the test cases that we simulate, we set the following parameters:

- Forecasted internal demand rate: $\mu^0 = 100$.
- Each internal shipment has probability $p_1 = 0.5$ from service area 1, and probability $p_2 = 0.5$ from service area 2.
- The exogenous demand rate on upper (γ_1^0) and lower arc (γ_2^0) arc are both 200 units per day.
- Initial arc capacities on upper ($u_1(0)$) and lower arc ($u_2(0)$) are both 300 units.
- Third-party cost of satisfying internal demand from service area 1 and service area 2: $(c_1^{3p}, c_2^{3p}) = (1, 2)$.
- The variable cost for any excess exogenous shipment (when retailer-controlled resources are not available) on FC to SC arc, and for any excess exogenous shipment on SC to DS arc are both $c^{\text{ex}} = 2$ per package.
- Ad hoc trucks can be added to FC-SC or SC-DS retailer-controlled arc separately. One ad hoc truck per arc cost $c^{ah} = 10$, and the capacity is $u^{ah} = 30$ units.
- Time horizon: $T = 1$ day.

For each experiment, the process is as follows:

1. At the start of every instance, we generate the actual internal demand rate (μ) and exogenous demand rates (γ_j) from normal distributions with mean and standard derivation associated with the forecasted demand rate:

$$\mu \sim N(\mu^0, \sqrt{\mu^0})$$
$$\gamma_j \sim N(\gamma_j^0, \sqrt{\gamma_j^0})$$

Note that μ^0 and γ_j^0 are the expected demand rates, and therefore, we assume this is known information for the assignment controllers.

2. Initialize time index at $t = 0$

3. We generate the next internal shipment. Since we assume the demand arrivals are Poisson, the time from the previous demand to this demand is generated by an exponential distribution with mean $1/\mu$ and the demand type is determined randomly with probability p from service area 1 and probability $1 - p$ from service area 2. Let τ be the exponentially distributed interarrival time; then we update the time index $t := t + \tau$.

4. We generate the number of exogenous demands that arrive between t and the previous internal shipment arrival $\rho(t)$ (or time 0 if it is the first shipment) from a Poisson distribution with rate $\gamma_j(t - \rho(t))/T$ for each arc j. The remaining capacity on arc j is then updated by the realized exogenous demand. If the number of exogenous demand is more than the remaining capacity, then we set the remaining capacity on the arc to zero, and we incur a cost of c^{ex} for each external shipment in excess of the arc capacity.

5. Re-solve the QP (or LP) to update the shadow price. Note that we update the demand forecast of the remaining horizon based on the initial demand forecasts and on the observed shipments:

$$\hat{e}_j(t) = \frac{T-t}{T}\left(\frac{T-t}{T}\gamma_j^0 + \bar{e}_j(t)\right) \tag{2.6}$$

$$\hat{d}_i(t) = \frac{T-t}{T}\left(\frac{T-t}{T}p_i\mu^0 + \bar{d}_i(t)\right) \tag{2.7}$$

where $\hat{e}_j(t)$ is the demand forecast of external demand on arc j at time t, $\hat{d}_i(t)$ is the demand forecast of internal demand from service area i at time t, $\bar{e}_j(t)$ is the observed shipment arrival of external demand on arc j up to time t, and $\bar{d}_i(t)$ is the observed shipment arrival of internal demand from service area i up to time t.

6. We assign the internal demand to the retailer-controlled transportation or 3P based on the updated shadow prices. Note that if any link along the path has no remaining capacity, then the package must be assigned to 3P.

7. If the test case allows ad hoc trucks, we decide whether to add ad hoc trucks to the arcs. We then update the states accordingly.

8. Repeat step 3 to 7 until the next shipment arrives after time T, and the simulation is terminated

2.4.1.1 The LP Algorithm (for the Two-link Network)

At each shipment arrival time t, the LP algorithm derives shadow price for upper arc (FC-SC) and lower arc (SC-DS) from the following linear program:

$$\min \quad \sum_{i=1,2} c_i y_i \tag{2.8a}$$

$$\sum_{i=1,2} x_i \leq (u_j(t) - \hat{e}_j(t))^+ \quad \forall j = 1, 2 \tag{2.8b}$$

$$x_i + y_i \geq \hat{d}_i(t) \quad \forall i = 1, 2 \tag{2.8c}$$

$$\mathbf{x}, \mathbf{y} \geq 0 \tag{2.8d}$$

where $\mathbf{x}$ is internal shipments assigned to the retailer-controlled route, $\mathbf{y}$ is internal shipments assigned to the third-party route. The shadow price of the upper ($\lambda_1(t)$) and lower arcs ($\lambda_2(t)$) are the dual variables of their associated resource constraint (2.8b). The shadow price of the retailer-controlled route is the sum of the shadow prices of upper and lower arcs with a maximum value capped at the most expensive third-party cost:

$$\min(\max(c_1, c_2), \lambda_1(t) + \lambda_2(t))$$

The algorithm then makes order fulfillment decisions by comparing the shadow price with the third-party cost, and picks the cheaper option to ship the shipment. If

we encounter a tie, we adopt a different tie-breaking rule in this section, where we prioritize the retailer-controlled routes over third party routes.

2.4.1.2 The QP Algorithm (for the Two-link Network)

At each shipment arrival time t, the QP algorithm derives shadow prices for the upper arc (FC-SC) and for the lower arc (SC-DS) from the following quadratic program:

$$\min \quad \sum_{j=1,2} \frac{1}{2} v_j(t) g_j^2 \tag{2.9a}$$

$$\sum_{i=1,2} x_i + \hat{e}_j(t) - g_j \leq \bar{f}_j(t) \quad \forall j = 1,2 \tag{2.9b}$$

$$x_i \geq \hat{d}_i(t) \quad \forall i = 1,2 \tag{2.9c}$$

$$\mathbf{x}, \mathbf{g} \geq 0 \tag{2.9d}$$

The targets and penalty coefficients are set by the following formula:

$$\bar{f}_j(t) = u_j(t) - k_1 \sqrt{\hat{e}_j(t) + \sum_{i=1,2} \hat{d}_i(t)} \quad \forall j = 1,2$$

$$v_j(t) = k_2 \min(c_1, c_2) \frac{1}{\sqrt{\hat{e}_j(t) + \sum_{i=1,2} \hat{d}_i(t)}} \quad \forall j = 1,2$$

where k_1 and k_2 are constants set to be 2 and 0.2, respectively, in our simulation. We note that the algorithm's performance depends on the parameter choice of these coefficients, and the parameters adopted in this chapter are hand-picked, where we do not try to find the best set of parameters for the tests. We will explain and justify this setting in the next chapter. We note that the flow targets are set slightly lower than capacity, where the gap is a function of the uncertainty in the number of shipments that could utilize the arc, including demand from service area 1 and 2 and the exogenous demand on their associated arc; the penalty coefficient is set according to the third-party costs and the internal and external forecasts. We derive the shadow price of the upper and lower arc from the dual variables of their associated capacity constraint in the QP. For the two-link example, by KKT condition, the dual variables

$(\lambda_j(t))$ can be expressed explicitly:

$$\lambda_j(t) = v_j(t)(\sum_{i=1,2} \hat{d}_i(t) + \hat{e}_j(t) - \bar{f}_j(t))^+.$$

The shadow price of the retailer-controlled route is the sum of the upper and lower arc's dual variables with a maximum value of the most expensive third-party cost:

$$\min(\max(c_1, c_2), \lambda_1(t) + \lambda_2(t))$$

Like the LP algorithm, the QP algorithm makes order fulfillment decision by comparing the shadow price with the third-party cost, and picks the cheaper option to ship the shipment. If we encounter a tie, we prioritize the retailer-controlled routes over third party routes.

2.4.1.3 Parameter Choice of the Ad Hoc Truck Controllers

For the threshold-based controller, at every shipment arrival time t, we add an ad hoc truck to arc j if the following inequality holds:

$$c_j^{3p}(\hat{e}_j(t) + \sum_{i=1,2} \hat{d}_i(t) - u_j(t))^+ > c_j^{\text{ah}}$$

where c_j^{3p} is set to the minimum of the third-party costs, i.e. $min(c_1, c_2)$. In addition, if cancellation of ad hoc trucks are allowed, at every shipment arrival time t we cancel an ad hoc truck on resource j if these three conditions are satisfied:

1. The following inequality holds:

$$c_j^{3p}(\hat{e}_j(t) + \sum_{i=1,2} \hat{d}_i(t) - u_j(t) - u_j^{ah})^+ < c_j^{\text{ah}}$$

2. Remaining capacity of arc j is more than an ad hoc truck capacity, i.e. $u_j(t) > u_j^{ah}$. This condition is to guarantee that no ad hoc trucks are cancelled if they already have shipments assigned to them.

3. The accumulated number of ad hoc trucks added to resource j is at least one. This condition is to make sure that we do not cancel pre-scheduled trucks; in other words, we only cancel the ad hoc trucks that are added in the current time period.

For the DP-based controller, at every shipment arrival time t, we approximate the cost-to-go function by the cost of the following LP:

$$J((S,\mathbf{a}),t) \approx W((S,\mathbf{a}),t) = \min \quad \sum_{i=1,2} c_i y_i + \sum_{j=1,2} c^{\text{ex}} y_j \tag{2.10a}$$

$$\sum_{i=1,2} x_i + x_j \leq u_j(0) + a_j u^{\text{ah}} \quad \forall j \tag{2.10b}$$

$$x_i + y_i \geq \hat{d}_i(t) \quad \forall i \tag{2.10c}$$

$$x_j + y_j \geq \hat{e}_j(t) \quad \forall j \tag{2.10d}$$

$$\mathbf{x}, \mathbf{y} \geq 0, \text{ integer} \tag{2.10e}$$

where $a_j = 1, 0$ or -1 depending on the ad hoc truck actions, $\hat{e}_j(t)$ denotes the exogenous demand forecast on arc j at time t, $\hat{d}_i(t)$ denotes the demand forecast from service area i at time t, x_i (y_i) denotes the number of service area i shipments shipped by retailer controlled route (third-party route), x_j (y_j) denotes the number of exogenous shipments on arc j shipped by retailer controlled route (third-party route).

2.4.2 Hindsight Solution

In this simple example, if we know the internal and exogenous demands, we can calculate the minimum (hindsight) transportation costs by a simple linear programming given the total exogenous and internal demand are known:

$$\min \quad \sum_{i=1,2} c_i y_i + \sum_{j=1,2} c^{\text{ex}} y_j + \sum_{j=1,2} c^{\text{ah}} a_j \tag{2.11a}$$

$$\sum_{i=1,2} x_i + x_j \leq u_j(0) + a_j u^{\text{ah}} \quad \forall j \tag{2.11b}$$

$$x_i + y_i \geq d_i \quad \forall i \tag{2.11c}$$

$$x_j + y_j \geq e_j \quad \forall j \tag{2.11d}$$

$$\mathbf{x}, \mathbf{y}, \mathbf{a} \geq 0, \text{ integer} \tag{2.11e}$$

$$\tag{2.11f}$$

where e_j denotes the total number of exogenous demands on arc j, d_i denotes the total number of demands from service area i, x_i (y_i) denotes the number of service area i shipments shipped by retailer controlled route (third-party route), x_j (y_j) denotes the number of exogenous shipments on arc j shipped by retailer controlled route (third-party route), a_j denotes the number of ad hoc trucks added to arc j. Note that if the ad hoc truck options are not allowed, we set $\mathbf{a} = 0$.

2.4.3 Results of Experiments

In this section, all test cases are based on parameters and procedures described in section 2.4.1, except for the sensitivity test, for which we will specify the different parameters (or process) if any. We compare the performance of different algorithms by simulating a number of realizations (50 instances) that leads to conclusions. Then, we perform sensitivity tests on the best performing algorithm to understand the robustness of the algorithm under different scenarios.

2.4.3.1 The QP algorithm is better than the LP algorithm

First, we compare the LP and QP algorithms' performances by simulating the two algorithms on the two-link network without the option of adding ad hoc trucks for simplicity. We simulate both algorithms on the same 50 instances. The average and

standard deviation of costs are summarized in the following table:

	LP algorithm	QP algorithm	hindsight
mean	20.88	19.20	14.10
std	17.08	21.34	16.61

On average, the QP algorithm performs slightly better than the LP algorithm. In addition, among the 50 test cases, QP performs better than LP in 35 out of 50 cases. To make a more statistically sound judgment, we perform a two-sample t-test by the following procedure. First, we create a null hypothesis that is counter to our belief. In this case, the null hypothesis is that the LP algorithm is better than the QP algorithm:

$$H_0 : \mu^{(LP-QP)} < 0,$$

where μ^{LP-QP} is the average difference between the cost from the LP and the cost from the QP algorithm (cost of LP subtracting cost of QP). We assume that $\mu^{(LP-QP)}$ is approximately normally distributed, and we perform t-test to the test statistics. If the p-value of the test is small, we reject the null hypothesis with confidence. The average cost difference of the 50 samples is 1.68 ($\hat{\mu}^{LP-QP}$) and the standard deviation of the cost difference is 6.54 (s^{LP-QP}). The t-test statistic is:

$$T = \sqrt{N}\frac{\hat{\mu}^{LP-QP} - 0}{s^{LP-QP}} = \sqrt{50}\frac{1.68}{6.54} = 1.817$$

The p-value derived from the t-statistics:

$$\mathbb{P}(t_{50} > 1.817) = 0.0376$$

where t_{50} denotes a t-distribution with 50 degrees of freedom. The small p-value suggests that our experiments show strong evidence that the QP algorithm is better than the LP algorithm with the 50 instances.

2.4.3.2 The DP-based controller and the threshold-based controller

Next, we allow ad hoc trucks and repeat the experiment to compare the two ad hoc truck controller's performances. In this experiment, we run the two controllers on the same 50 instances, and the assignment decisions are both made by the QP algorithm. For simplicity, we do not allow cancellation of ad hoc trucks, i.e., once an ad hoc truck is added to the system, it cannot be removed. We run 50 instances with two different controllers and the mean and standard deviation of the costs are:

	DP-based	threshold-based	hindsight
mean	15.60	16.08	8.96
std	10.62	10.59	9.92

The average cost with the DP-based controller is lower than the threshold-based controller. However, among the 50 test cases, DP-based controller performs better than threshold-based controller in 5 cases, performs equally in 40 cases, perform worse in 5 cases. Again, we perform t-test to the test statistics following the same procedure in the previous section. The p-value of our test is 0.117, which doesn't show a strong evidence that one controller is better than the other.

2.4.3.3 Allowing Cancellation of Ad Hoc Trucks Reduces Costs

We simulate 50 test cases with the same assignment engine (the QP algorithm) and the same ad hoc truck controller (DP-based controller) but now with or without the option of ad hoc truck cancellation. The mean and standard deviation of the costs are:

	without cancellation	with cancellation	hindsight
mean	15.60	11.22	8.96
std	10.62	11.99	9.92

Among the 50 test cases, with cancellation performs better than without cancellation in 22 cases, performs equally in 21 cases, performs worse in 7 cases. The test statistics show that the algorithm's performance is better with cancellation than without, with a small p-value 4.17e−5.

2.4.3.4 Sensitivity Tests

Based on the above experiments, we find that the algorithm that performs the best on this test case for a two-link network is the one that assigns shipments with QP algorithm, and that makes ad hoc truck decisions with the DP-based controller with the option of cancelling ad hoc trucks. In this section, we perform sensitivity tests to understand the robustness of this algorithm (QP + DP-based ad hoc controller + with cancellation) under different scenarios, and we refer to this version of algorithm as " the algorithm" in this section.

The two-link network differs from the one-link model by having external flows. The first sensitivity test is on the external demand rates. We vary the external demand forecast (γ_j^0) of both arcs between 100 and 300 with step size of 25, i.e. $\gamma_j^0 = 100, 125, 150, \cdots, 300$. There are 9^2 combinations, and we simulate 10 instances for each combination. Figure 2-9 and Figure 2-10 are the heat map of average hindsight costs and the average costs by the algorithm, respectively.

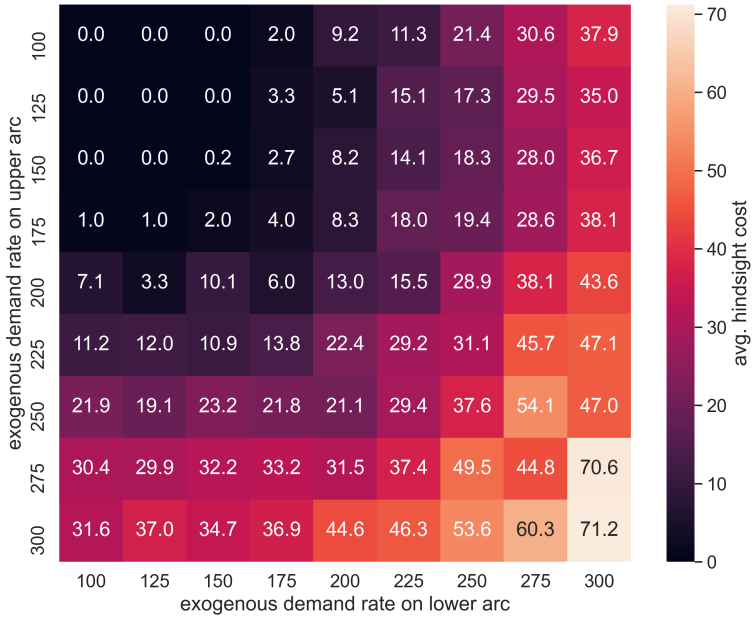


Figure 2-9: Average hindsight costs under different exogenous demand rates

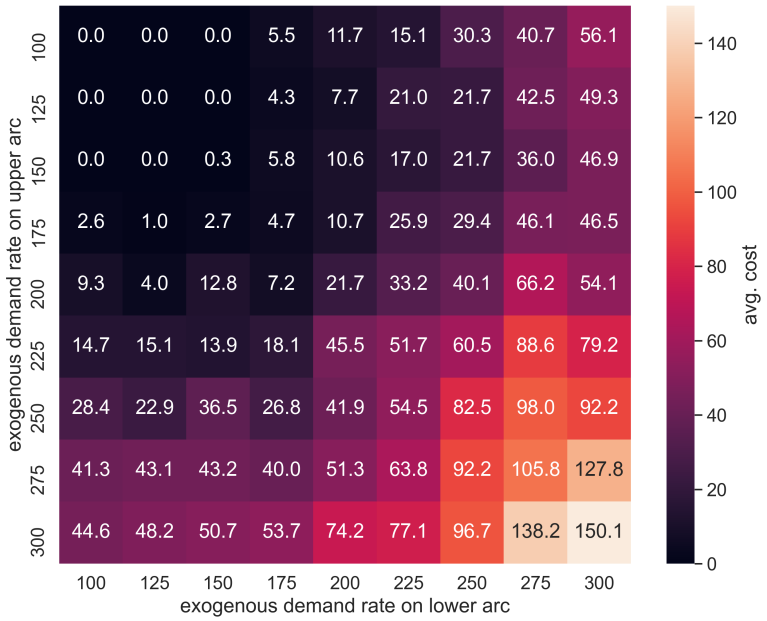


Figure 2-10: Average cost by the algorithm under different exogenous demand rates

As expected, the average hindsight cost and the average cost of the algorithm increase as the exogenous demand rate increases. To compare the two algorithms, we calculate the mean percentage error (MPE):

$$\text{MPE} = \frac{\text{avg. cost by the algo.} - \text{avg. hindsight cost}}{\text{avg. hindsight cost}}$$

for each test case. Figure 2-11 shows MPE under different exogenous demand rates. The results are quite robust to different combinations of exogenous demand rates within groups. When one of the exogenous demand rate is more than 200, the percentage errors are in the range of 24% to 138%. When one of the exogenous demand rate is less than 200, the mean percentage errors are mostly in range from 20% to 60%. Note that the controller does not have the knowledge of the actual external demand rates (γ_j) but the controller knows the expected external demand rate (γ_j^0) and it updates the forecasts based on observed demand arrivals linearly with equation (2.6). Therefore, the controller's performance decreases as the forecast becomes less

accurate. From the 10 test cases, we observed that our algorithm performs the worst in the cases where the actual demand rates (γ_j) are much higher than the expected demand rate (γ_j^0) and the demand arrival is relatively sparse in the beginning, and relatively dense toward the end. This leads to poor forecasts for external demands, which leads to poor ad hoc truck decisions by the algorithm.

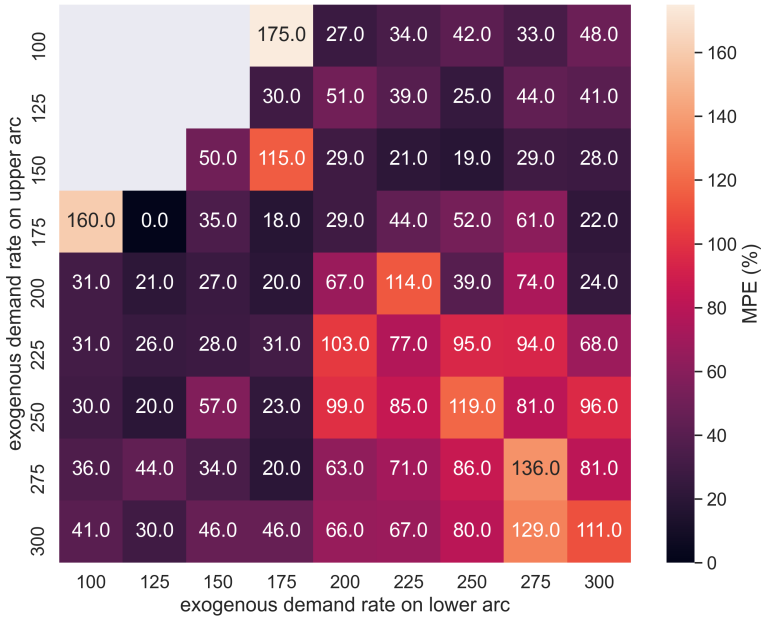


Figure 2-11: MPE under different exogenous demand rates (Note that the blanked space are the ones with zero hindsight cost, where cost difference fail to provide a meaningful value due to zero denominator).

In addition, the number of ad hoc trucks added by our algorithm is less than the hindsight solution in average. In figure 2-12 and figure 2-13, we show the average difference (algorithm minus hindsight) in the number of ad hoc trucks added. We observe that all scenarios are non-positive, and no scenario has more than 2-truck difference in average on any arc.

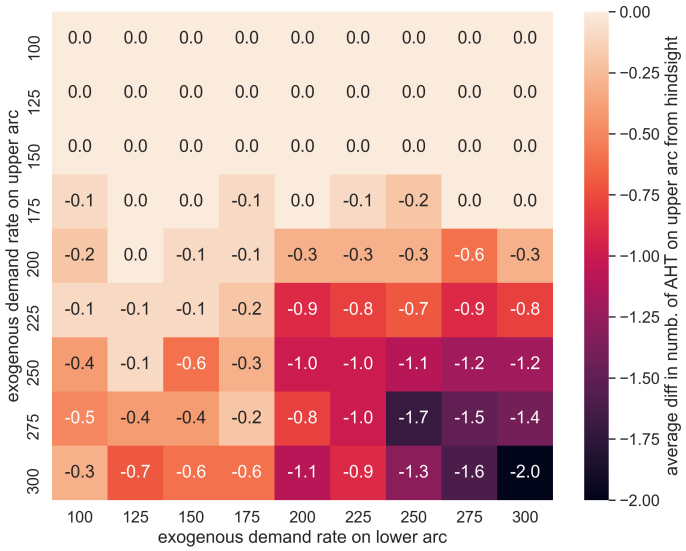


Figure 2-12: Average difference in the number of ad hoc trucks added by the algorithm and the hindsight solution on upper arc

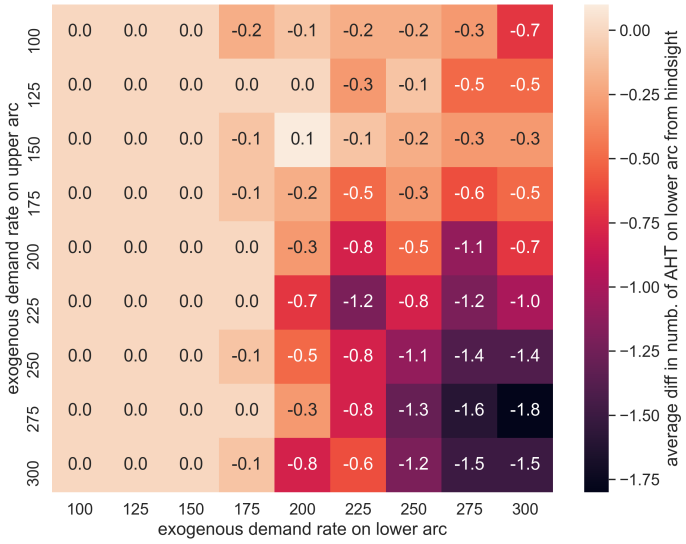


Figure 2-13: Average difference in the number of ad hoc trucks added by the algorithm and the hindsight solution on lower arc

The next sensitivity test is on internal demand rates. We vary the internal demand forecasts from 30 units/day to 200 units/day with 10 units/day increment. Figure

2-14 shows the average hindsight cost and the average cost incurred by our algorithm. The gap between the average cost of our algorithm and the hindsight solution increases slightly as the internal demand rate increases. However, the mean percentage error stays roughly constant when internal demand rate is greater than 100. The final

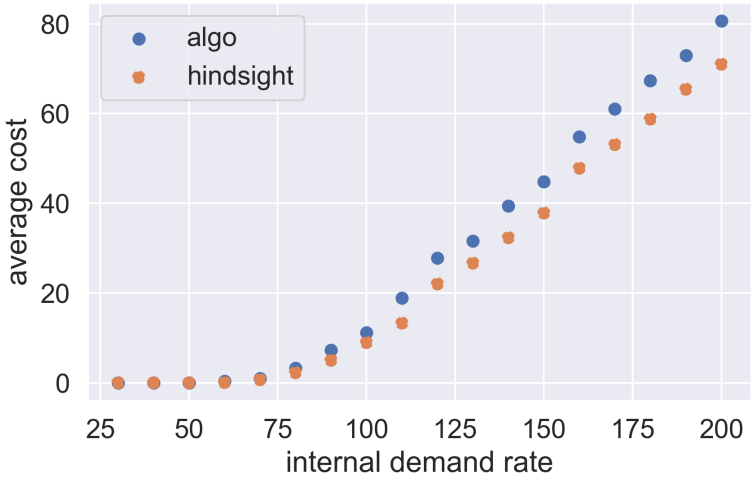


Figure 2-14: Average cost of the algorithm under different internal demand rates

sensitivity test is on ad hoc truck costs. We vary the ad hoc truck costs from 5 dollars to 15 dollars with 1 dollar increment. Figure 2-15 shows the average hindsight cost and the average cost incurred by our algorithm. Again, we observe a slight increase in the average cost difference. However, the mean percentage error stays below 30% for all cases.

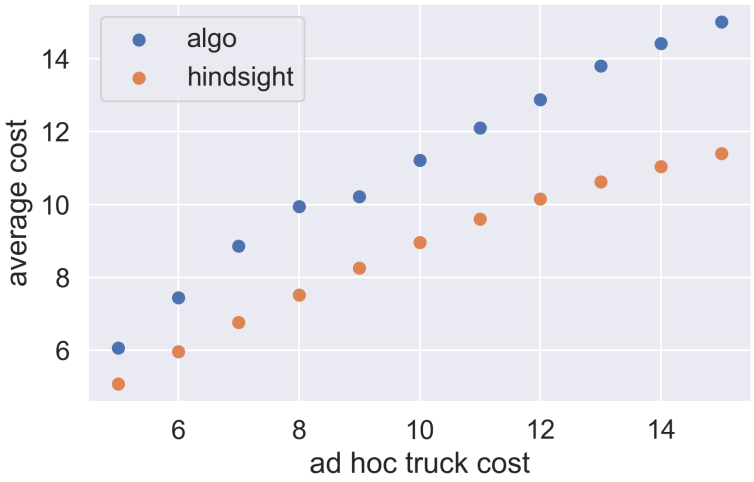


Figure 2-15: Average cost of the algorithm under different ad hoc truck costs

2.5 DP Formulation of the One and Two Link Network

In the one-link and two-link example, the demand arrivals follows a known probability distribution. Therefore, we could determine the optimal expected cost by formulating the problem as a dynamic programming problem. In this section, we introduce the dynamic programming formulation for both the one and two link network, which provides another benchmark for the order fulfillment algorithm's performance.

2.5.1 DP of the One-link Model

In the one-link model, the demand arrival follows a Poisson process with rate μ and each demand has probability p_1 from service area 1, and p_2 from service area 2. The initial capacity of the link is denoted by $u(0)$. We divide the look-ahead period into N infinitesimal time periods, such that the probability of having more than one demand arrival is negligible. The dynamic programming problem takes the form:

$$J_t(u) = (1 - \frac{\mu}{N})J_{t+1}(u) + \frac{\mu}{N}\sum_{i=1,2} p_i \min(J_{t+1}(u-1), J_{t+1}(u)) + c_i) \tag{2.12}$$

where $t = 0, 1, \cdots N$, u is the remaining capacity of the link, J_t is the optimal cost at step t. The optimal expected cost ($J_0(u(0))$) over the one-day horizon with initial capacity $u(0)$ can be estimated by the above formula with boundary condition $J_N(u) = 0$ for $u \geq 0$ and boundary condition $J_N(u) = \infty$ with $u < 0$. Note that the first term is the expected cost for the case where there is no shipment at time step t, the second term is the expected cost of the case where there is a shipment to service area 1 or service area 2. If ad hoc trucks are allowed, we modify equation (2.12) by adding another term in the minimization which signifies the case where an ad hoc truck is added to the transportation link:

$$\begin{aligned} J_t(u) = &(1 - \frac{\mu}{N}) J_{t+1}(u) \\ &+ \frac{\mu}{N} \sum_{i=1,2} p_i \min(J_{t+1}(u-1), J_{t+1}(u)) + c_i, J_{t+1}(u - 1 + u^{\text{ah}}) + c^{\text{ah}}) \end{aligned}$$

To compare with simulation results in table 2.1, we calculate the optimal expected cost from our DP formulation by plugging in the same parameters: $u(0) = 100$, $\mu = 100$, $p_1 = p_2 = 0.5$ $c^{\text{ah}} = 10$, $u^{\text{ah}} = 30$. With $N = 1440$ (which leads to one minute time interval), the optimal expected cost with ad hoc truck option is 3.55, and 4.41 without ad hoc truck option, which are both similar to the hindsight cost in table 2.1.

2.5.2 DP of the Two-link Model

In the two-link model, the demand arrival also follows a Poisson process with rate μ and each demand has probability p_1 (p_2) from service area 1, (service area 2). The initial capacity of the upper and lower links are $u_1(0)$ and $u_2(0)$, respectively. External demand consumes the upper and lower arc resource with Poisson rate γ_1 and γ_2, respectively. We, again, divide the look-ahead period into N infinitesimal time periods, such that the probability of having more than one demand arrival (including both internal and external) is negligible. The dynamic programming problem takes

the form:

$$\begin{aligned} J_t(u_1,u_2) &= (1-q_0-q_1-q_2)J_{t+1}(u_1,u_2) \\ &\quad + q_0\Big(\sum_{j=1,2} p_j \min(J_{t+1}(u_1-1,u_2-1), J_{t+1}(u_1,u_2)+c_j^{3p})\Big) \\ &\quad + q_1 \min(J_{t+1}(u_1-1,u_2), J_{t+1}(u_1,u_2)+c^{\text{ex}}) \\ &\quad + q_2 \min(J_{t+1}(u_1,u_2-1), J_{t+1}(u_1,u_2)+c^{\text{ex}}) \end{aligned} \tag{2.13}$$

where $q_0 = \frac{\mu}{N}$ is the probability of having an internal demand, $q_1 = \frac{\gamma_1}{N}$ ($q_2 = \frac{\gamma_2}{N}$) is the probability of having an external demand on upper (lower) arc. The optimal cost $J_0(u_1(0), u_2(0))$ can be calculated through equation (2.13) with boundary condition $J_N(\cdot,\cdot) = 0$ with $u_1 \geq 0$ or $u_2 \geq 0$ and boundary condition $J_t(u_1,u_2) = \infty$ for all t with $u_1 < 0$ or $u_2 < 0$. If ad hoc trucks are allowed, equation (2.13) becomes:

$$\begin{aligned} J_t(u_1,u_2) &= (1-q_0-q_1-q_2)J_{t+1}(u_1,u_2) \\ &\quad + q_0\Big(\sum_{j=1,2} p_j \min(J_{t+1}(u_1-1,u_2-1), J_{t+1}(u_1,u_2)+c_j^{3p}, \\ &\quad J_{t+1}(u_1-1+u^{\text{ah}},u_2-1)+c^{\text{ah}}, J_{t+1}(u_1-1,u_2-1+u^{\text{ah}})+c^{\text{ah}}, \\ &\quad J_{t+1}(u_1-1+u^{\text{ah}},u_2-1+u^{\text{ah}})+2c^{\text{ah}})\Big) \\ &\quad + q_1 \min(J_{t+1}(u_1-1,u_2), J_{t+1}(u_1-1+u^{\text{ah}},u_2)+c^{\text{ah}}) \\ &\quad + q_2 \min(J_{t+1}(u_1,u_2-1), J_{t+1}(u_1,u_2-1+u^{\text{ah}})+c^{\text{ah}}) \end{aligned}$$

Note that we assume that at most one ad hoc truck can be added to each link in each time step.

Again, we calculate the optimal expected cost from the DP formulation by plugging in the same parameters used in the previous experiment: $u_1(0) = u_2(0) = 100$, $\mu = 100$, $p_1 = p_2 = 0.5$ $c^{\text{ah}} = 10$, $\gamma_1 = \gamma_2 = 200$, $u^{\text{ah}} = 30$. With $N = 1440$ (which leads to one minute time interval), the optimal expected cost with ad hoc truck option is 10.3, and 16.8 without ad hoc truck option, which are both slightly higher than the hindsight cost (8.96 with ad hoc truck option, 14.1 without ad hoc truck option).

2.6 Summary

So far, we have introduced the order fulfillment problem with two simple examples (the one and two link network). We introduced both static and dynamic algorithms and test these algorithms in a stylized setting. The test results in the two-link model suggest that the QP algorithm with DP-based ad hoc controller and truck cancellation outperforms other algorithms with statistical significance. The sensitivity tests on this algorithm also suggest that the algorithm is robust to a range of demand and cost inputs. Finally, we introduce the dynamic programming formulation for these two examples, which provides another benchmark for the order fulfillment algorithm's performance.

Chapter 3

The Transportation Network and Capacity Planning

In the previous chapter, we gave two small examples, derived from the online retailer's transportation network, to introduce the order fulfillment problem. In this chapter, we provide a more detailed description of the transportation network, then formulate a capacity planning problem. For this formulation, we need to explain how the "time" element contributes greatly to the complexity of the problem. In section 3.1, we introduce our definition of resources and routes in our model, and we provide an example to illustrate these definitions. In section 3.2, we introduce the notion of commodities, and provide a guideline for capacity planning.

3.1 Network Representations

3.1.1 Definition of Resources

We consider a transportation network that has three types of nodes: fulfillment centers (FCs), sortation centers (SCs), delivery stations (DSs) as explained in chapter 2. From now on, we index the FCs by u, SCs by v and DSs by w. The network connects these nodes with three types of retailer-controlled transportation arcs: FC-SC $((u, v))$, SC-DS $((v, w))$, FC-DS $((u, w))$. From each DS, the retailer relies on local

delivery resources to perform last-mile delivery of packages to customers. In addition to the retailer-controlled transportation resources, the retailer may assign some packages to third-party carriers that can pickup packages at certain FCs and SCs, and then deliver the packages to customers' doorstep. In this section, we explain how we define these transportation resources in our model.

In the one and two link network in chapter 2, we assume that facility nodes do not have capacity limits. In addition, we assume that the transportation arcs are available for the entire time period (e.g., for 24 hours in the example), and that the delivery deadline for all shipments is the end of the time period. In this setting with these assumptions, there is no "time" element, which leads to a simple definition for an arc resource. That is, for each transportation arc that connects two facilities or nodes, we only have a single arc resource. However, in reality, most facilities do not operate continuously and do have limits in terms of number of packages they could process in a limited time frame. On each transportation arc, trucks depart and arrive at certain time frames. In addition, shipments can have different delivery deadlines. Therefore, in order to meet the delivery deadline for a shipment, the model has to account for multiple time periods within a day and identify routes that consider the relevant time delays and time-dependent constraints through the network. The operational planning for the transportation network depends on a set of time deadlines at each facility, known as *critical pull times* (CPT). The purpose of these critical pull times is to help coordinate the steps necessary to deliver each shipment. For instance, for a link from an FC to an SC, the critical pull time relates the picking time of a shipment to the departure time for its first transportation leg. For example, suppose the CPTs for the link each day are 8 AM, 5 PM and 11 PM. Then if a shipment gets picked between 8 AM and 5 PM, then it will depart on a truck no later than 5 PM. But if a shipment gets picked shortly after 5 PM, then it may be delayed until 11 PM before departing on a truck.

Transportation planning is specified in terms of these CPTs. The transportation plan will be specified in terms of the number of truck departures for each CPT. In the above example, the transportation plan might be for 10 trucks to go on the link

between 8 AM and 5 PM, 6 trucks to go between 5 PM and 11 PM, and 3 trucks scheduled for between 11 PM and 8 AM next day. The exact departure times will depend on how quickly trucks get filled up, which depends on the shipment picking. Similarly, at a SC, the critical pull times will coordinate the inbound flow with the outbound flow; if a CPT at the SC is noon, then any inbound flow that arrives and get sorted prior to noon will depart as outbound flow at noon. At the DS, the CPTs correspond to when delivery vehicles depart for last mile delivery of customer shipments. The purpose of these critical pull times is to help coordinate the steps necessary to deliver each shipment.

We use the CPTs to define the arc resources in the transportation network. For each transportation arc or link (u, v), we define a resource (u, v, t) for each CPT in the relevant planning horizon, where t signifies the CPT. We understand the capacity of resource (u, v, t) to be the capacity on the arc (u, v) scheduled for the time segment (s, t) where s is the CPT immediately prior to CPT t. The transportation plan determines the capacity level (number of trucks) for each of these link resources. For each node or facility, there are limited number of packages that can be processed per time segment. Therefore, we may have node resources (v, t), (w, t) which signify both the type of node (SC, DS) and the end time of a time segment (ex: a labor time shift). The capacity plan determines a capacity level for each of these resources, in terms of how many units the resource can process within a time segment.

Aside from the retailer-controlled transportation, the capacity plan may need to account for limits on third-party deliveries. The online retailer signs contracts with third-party carriers in advance that may specify a maximum quantity that can be assigned to the carrier from a pick-up location, which can be either an FC or SC. We model the third-party resource by its pick-up location and a pick-up time window.

In summary, the set of resources of the transportation network (J) is composed of a set of arc resources (J^{arc}), a set of node resources (J^{node}), and a set of third-party resources (J^{tp}): $J = J^{\text{arc}} \cup J^{\text{node}} \cup J^{\text{tp}}$. We will use index j to denote a resource.

3.1.2 Definition of Routes

A route is composed of arc and node resources. We will consider four different route patterns in the transportation network, which differ in terms of resources:

- direct route: one unit of FC-DS resource and one unit of DS resource.
- indirect route: one unit of FC-SC resource, one unit of SC resource, one unit of SC-DS resource, and one unit of DS resource (We note that in practice the indirect route might contain one or more SCs.)
- third-party route: one unit of FC-3P resource, whereby a shipment is transferred to a third-party carrier at the FC; the third-party carrier then delivers the shipment to the customer.
- mixed route: one unit of FC-SC resource, one unit of SC resource, and one unit of SC-3P resource; this is similar to the third-party route, but the transfer to the third-party carrier occurs at a SC.

We note that we do not consider any FC-related resources in the formulation of the middle-mile routes. This is because we assume that for any incoming shipment an existing controller makes the FC assignment decision, and that this controller accounts for any FC related resources in making its assignment decision. Hence, we do not consider FC-related resources in the transportation network planning model. In addition, we do not consider the resources for last mile delivery (delivery from DS to customer's doorstep), which is needed for completing the direct and indirect routes. The last mile delivery might come with a cost, which can be included in the model.

The resources along a route need to be time-compatible, i.e., the timestamp of an upstream resource should not be too far or too close from the timestamp of a downstream resource. If the timestamp of the upstream resource is too far (early) from the downstream resource, then packages shipped by the route might dwell in the system for too long; if the timestamp of the upstream resource is too close, then

the packages shipped by the route might not be able to make it to the next resource on time.

In order to create a set of routes that is time-compatible, we make several assumptions. First, we assume that we know the minimum dwell or process time of a shipment at every node facility, and we know the transit time on every transportation arc. Second, we assume that each order travels in the network according to these dwell and transit times.

We then create routes based on the dwell and transit time information. We specify a route by its path and by the resources it uses. The resources along the route are time-compatible if with normal transit and process times, a shipment can meet each downstream timestamp provided that upstream shipments occur at their respective timestamp. In addition, we will consider only "no wait" routes: once a shipment arrives at a node, it will depart on an outbound arc at the next earliest CPT. This is a practical requirement in that delaying a shipment means that the facility will have to hold or store the shipment for some period of time, for which there will be a cost and/or required storage space; furthermore, there will usually be no value from delaying a shipment. In summary, for a route to be time-compatible, we require every upstream resource (with timestamp t_1) and the downstream resource (with timestamp t_2) to satisfy the following conditions:

- $t_1 + \delta < t_2$, where δ is the dwell or transit time to travel from the upstream resource to the downstream resource
- t_2 is the earliest resource of the kind that satisfies this condition

For instance, a typical route might be composed of the following resources: $(u, v, t_1) \rightarrow (v, t_2) \rightarrow (v, w, t_3) \rightarrow (w, t_4)$; this signifies a path from FC u to SC v to DS w, where t_1 is the CPT for link (u, v), t_2 is the end time of a labor shift at SC v, t_3 is the CPT for link (v, w), and t_4 is the CPT at DS w. The resources along the route satisfy the following conditions:

1. t_1 plus the travel time on link (u, v) is less than t_2, and t_2 is the earliest resource on node (v) that satisfies this condition

2. t_2 plus dwell time at SC v is less than t_3, and t_3 is the earliest resource on arc (v, w) that satisfies this condition

3. t_3 plus the travel time on link (v, w) is less than t_4, and t_4 is the earliest resource on node (w) that satisfies this condition

We create routes (indexed with r) according to the four route patterns, and the routes are formed with resources with compatible timestamps. The set of routes of the transportation network (R) is composed of the set of direct routes (R^{direct}), the set of indirect routes ($R^{indirect}$), the set of third-party routes (R^{tp}), and the set of mixed routes (R^{mixed}):

$$R = R^{direct} \cup R^{indirect} \cup R^{tp} \cup R^{mixed}$$

So far, we have been describing the details of forming routes with time-compatible resources. Next, we want to point out that, in real-time execution, to examine if a route is feasible for a shipment, we do not need to know all the timestamps of resources. Instead, as long as the following three conditions are satisfied, a route will be feasible to a shipment:

- The origin and the destination of the route matches with the shipment's origin FC and destination (DS for direct and indirect routes, service area for third-party and mixed routes).

- The arrival time for the shipment is earlier than the timestamp of the first resource of a route. That is, let the shipment arrival time be a, the first timestamp of a route be t. Then $a \leq t$.

- The promise time for the shipment is later than the timestamp of the last resource of a route (plus the travel time of last mile delivery from DS to the customer if considering a direct or indirect route. However, in our experiment, we assume this travel time is negligible). That is, let the shipment promise time be p, the last timestamp of a route be t. Then $t \leq p$.

3.1.3 Modeling a Two-Link Network

In this section, we model a two link network with our definition of resources and routes. Figure 2-1 shows an illustration of a two-link network. Table 3.1 shows the daily CPTs and the transit and dwell times of each resource. In this model, there are

resource type	daily CPT	transit (dwell) time
sc node	1PM, 3PM	1
ds node	6PM, 8PM	1
fc-sc arc	8AM, 10AM	4
sc-ds arc	3PM, 5PM	2
fc-ds arc	9AM, 12PM (noon)	6
fc third-party	9:30AM	Nan
sc third-party	2:30PM	Nan

Table 3.1: Daily CPTs and dwell and transit times of resources in the example network.

12 resources in total per day:

- SC node resource: $j(\text{SC}, 1\text{PM})$, $j(\text{SC}, 3\text{PM})$
- DS node resource: $j(\text{DS}, 6\text{PM})$, $j(\text{DS}, 8\text{PM})$
- FC-SC arc resource: $j(\text{FC}, \text{SC}, 8\text{AM})$, $j(\text{FC}, \text{SC}, 10\text{AM})$
- SC-DS arc resource: $j(\text{SC}, \text{DS}, 3\text{PM})$, $j(\text{SC}, \text{DS}, 5\text{PM})$
- FC-DS arc resource: $j(\text{FC}, \text{DS}, 9\text{AM})$, $j(\text{FC}, \text{DS}, 12\text{PM})$
- FC third-party resource: $j(3\text{P@FC}, 9{:}30\text{AM})$
- SC third-party resource: $j(3\text{P@SC}, 2{:}30\text{PM})$

There are four types of routes from the FC to the customer destination, including a direct route (FC $\rightarrow$ DS), a indirect route (FC $\rightarrow$ SC $\rightarrow$ DS), a third party route (3P resource at FC) and a mixed route (FC $\rightarrow$ SC $\rightarrow$ 3P). There are more than four feasible routes since there are multiple CPTs for some arcs. To form a feasible time-compatible route, we start with the first resource, which differs depending on the route type. If the route type has a downstream resource with a CPT, then we

identify the earliest downstream resource that is time compatible. For example, to form a direct route, the first resource is a FC-DS resource. Suppose that we consider the FC-DS resource with 9AM CPT, 9AM plus the travel time on FC-DS (which is 6 hours) is 3PM in the afternoon, and 3PM plus the dwell time at DS (which is 1 hour) is 4PM in the afternoon. After 4PM, the next earliest CPT at the DS is 6PM. Thus, the arc resource (FC, DS, 9AM) connected to the node resource (DS, 6PM) forms a feasible direct route. We note that for indirect route, and mixed routes, the SC timestamp stands for the start of the labor time shift. In this example, there are two SC reosurces a day at 1PM and 3PM, and the dwell time at the SC is 1 hour. Anything that arrives before 1 PM will get sorted between 1 and 2 PM, and then be ready for truck departures from 2 PM on. And anything that arrives between 1PM and 3PM will get sorted between 3 and 4PM, and can depart on any truck after 4 PM.

In this manner, we can construct the six feasible routes for this example, which we list as follows:

- two direct routes: $r(j(\text{FC}, \text{DS}, \text{9AM}), j(\text{DS}, \text{6PM}))$ and $r(j(\text{FC}, \text{DS}, \text{12PM}), j(\text{DS}, \text{8PM}))$
- two indirect routes: $r(j(\text{FC}, \text{SC}, \text{8AM}), j(\text{SC}, \text{1PM}), j(\text{SC}, \text{DS}, \text{3PM}), j(\text{DS}, \text{6PM}))$, $r(j(\text{FC}, \text{SC}, \text{10AM}), j(\text{SC}, \text{3PM}), j(\text{SC}, \text{DS}, \text{5PM}), j(\text{DS}, \text{8PM}))$
- one third-party routes: $r(j(\text{3P@FC}, \text{9:30AM}))$
- one mixed routes: $r(j(\text{FC}, \text{SC}, \text{8AM}), j(\text{SC}, \text{1PM}), j(\text{3P@SC}, \text{2:30PM}))$

3.2 Capacity Planning

If we take a snapshot of the online retailer's current network, we see trucks running between facilities, shipments being processed within facilities, and third-party trucks come and go from the FCs and SCs to pickup shipments for delivery. Capacity planning is an activity that determines the quantity and timing of the resources in

the network; this includes the number of trucks to schedule on each arc and the staffing levels for each shift for each facility. A capacity plan determines the shipping capacity on each transportation arc, and the throughput capacity (package process rate) of each facility node. In addition, a capacity plan can account for a limit on the number of packages that each third-party carrier can handle, which is usually specified by contract. Note that we assume that the network structure, i.e., node locations, active transportation lanes, third-party pick up points and all the CPTs are predetermined, and capacity planning is based on this fixed network structure.

The capacity plan has as input the demand forecasts over a fixed period of time ($t = t^{\text{start}}$ to $t = t^{\text{end}}$), and then determines how many retailer-controlled resources to plan to handle the forecast of demand that arrives in this period of time. In this section, we introduce the notion of commodities, and provide a detailed description of the planning problem and the required inputs. (We note that the process of identifying relevant resources ($j(t^{\text{start}}, t^{\text{end}})$) and routes are non-trivial; readers can refer to appendix B for more details.)

3.2.1 Definition of Commodities

The online retailer has detailed information associated with every historical shipment, including the time information of the shipment (the time the shipment was placed, promise time), the dimension (length, width, height) of the package, shipment content (SKUs and quantities), the route that the shipment was shipped by, and the set of feasible routes for the shipment. We define “commodity" as an aggregate of the shipments for the purpose of capacity planning. The features for demand aggregation include origin FC, destination DS or service area (depending on desired granularity), package dimension, arrival time (the time the order was assigned to the FC) and promise time of the shipment. For discrete features, e.g., origin FC, destination DS, we simply group by distinct elements. For continuous features, e.g., arrival time, promise time, and package dimension, we create discrete thresholds to form the grouping. Therefore, the number of possible commodities depends on the granularity of the discrete thresholds of the continuous features. We note that the level of aggregation is

important and should be chosen carefully for the best planning results. In particular, if the granularity is too small, the accuracy of demand forecast at the commodity-level can be arbitrarily bad; if the granularity is too large, the capacity planning result will be too gross to be useful, despite having a perfect forecast. In summary, each commodity k is specified by (u, d, g, a, p) where u denotes the origin FC, d denotes the destination DS, g denotes the dimension group, a is the arrival time group, and p is the promise time group. We denote the set of commodities with arrival time within the planning horizon $t = t^{\text{start}}$ to $t = t^{\text{end}}$ by $K(t^{\text{start}}, t^{\text{end}})$. Note that in our project, we assume all packages are in the same dimension group, and therefore, we will omit this dimension hereafter.

Next, to formulate a planning model, we generate the set of feasible routes for each commodity. A route (r) is feasible to a commodity (k) if the following three conditions are satisfied:

- The route and the commodity have the same origin FC and destination DS

- The arrival time for the commodity is earlier than the first CPT for the route. That is, let the shipment arrival time in commodity k be $a(k)$, the first CPT of the route r be t, and suppose the dwell time at the FC is γ. The route is feasible if $a(k) + \gamma \leq t$

- The commodity can be delivered on time by the route, i.e., the promise time of the commodity is later than the delivery time of the route; this is determined by the CPT for the DS for retailer-controlled routes, and is specified by contract for the third-party carrier routes.

For each commodity (k), we identify all the routes in the relevant route set $(R(t^{\text{start}}, t^{\text{end}}))$ that satisfy the three conditions above to create the feasible route set (R_k) for commodity k.

3.2.2 Problem Formulation

For the capacity planning problem, we assume that for each transportation arc resource, there is a fixed cost for each truck that is scheduled on the resource. In addition, we assume that there is a route-specific variable cost for each shipment of a commodity assigned to the route. For example, consider a mixed route in which a shipment goes from an FC to a SC, and then is transferred to a third party for delivery; the variable cost includes any variable process cost for handling the shipment at the FC and at the SC, plus the cost paid to the third party for the delivery. The goal of the planning problem is to minimize the total cost, which is the sum of the fixed cost for scheduling trucks, plus variable transportation and processing costs while satisfying the demand forecast over some fixed time interval ($t = t^{\text{start}}$ to $t = t^{\text{end}}$); the solution is the optimal capacity of the resources.

In the capacity planning problem, we need to account for possible boundary effects. For instance, at the start of the planning horizon t^{start}, there may be shipments in the system that will require resources. Similarly, at the end of the planning horizon t^{end}, there may be shipments not yet delivered that will require resources in subsequent time periods.

We provide two ways of formulating the capacity planning problem, where the two formulations address the boundary problem differently. In the first formulation, we assume that we can estimate and account for the resources required by the shipments that arrive outside the planning horizon. In the second formulation, we assume that there is demand regularity and that we can then set a capacity plan that repeats over some time cycle. For example, suppose that the planning horizon is a week, and we forecast the same demand pattern for each week. Then, we develop a capacity plan that repeats weekly. In this case, the boundary conditions at the start of the planning horizon match the boundary conditions at the end of the planning horizon. We then determine the capacity plan that assures this “wrap-around" condition. We start with introducing a list of notation for both formulations:

Sets

- J: set of relevant resources, note that in this planning problem $J = J(t^{start}, t^{end})$
- J^{tp}: set of third-party resources, which is a subset of J
- K: set of relevant commodities, note that in this problem $K = K(t^{start}, t^{end})$
- R: set of routes, note that in this planning problem, $R = R(t^{start}, t^{end})$
- R_k: set of routes feasible to commodity k, which is a subset of R.
- R_j: set of routes that utilize resource j, which is a subset of R.
- R^{tp}: set of third-party routes, which is a subset of R.
- R^{mixed}: set of mixed routes, which is a subset of R.

Parameters

- π_j: cost per unit of resource on resource j, either arc or node
- u_j: capacity in packages per unit of resource j, for either an arc or node
- c_{kr}: variable cost of commodity k on route r. Note that for simplicity, we assume here that for the retailer-controlled route $c_{kr} = 0$; hence, for now, c_{kr} just includes any relevant third-party cost for the route. This could be easily changed to allow a non-zero variable cost for any retailer-controlled route.
- d_k: demand forecast of commodity k
- p_j: the amount of resource j that is committed or reserved for serving out-of-the-horizon demand

Variables

- x_{kr}: amount of shipments in commodity k assigned to route r
- n_j: number of units for resource j, which are trucks on arcs and are staff level on nodes

3.2.2.1 Formulation 1

The objective is to minimize the transportation and processing costs, which is composed of truck costs on retailer-controlled arcs, plus variable processing and third-party costs on routes. The first constraint is the demand constraint, where commodities are matched with routes. The second constraint is the capacity constraint, where the sum of the reserved resource and the assigned commodities on the resource is less than the planned capacity. We note that the capacity of each resource is termed in units of packages. This formulation assumes there is no upper limit on the use of third-party routes; if there is a limit, it is easy to add a constraint.

$$\begin{aligned}
\min \quad & \sum_{j \in J} \pi_j n_j + \sum_{k \in K, r \in R_k} c_{kr} x_{kr} \\
\text{s.t.} \quad & \sum_{r \in R_k} x_{kr} = d_k \quad \forall k \in K \\
p_j + & \sum_{k \in K} \sum_{r \in R_k \cap R_j} x_{kr} \leq n_j u_j \quad \forall J / J^{tp} \\
& x_{kr} \geq 0 \quad \forall r \in R_k,\ k \in K \\
& n_j \geq 0,\ \text{integer} \quad \forall j \in J / J^{tp}
\end{aligned}$$

3.2.2.2 Formulation 2

First, we introduce the concept of "counterpart resources". A counterpart resource of a resource j within time horizon t^{start} to t^{end} is a resource that has the same physical location, but the CPT is one week apart (suppose that our planning horizon is one week) from resource j. In addition, both resource j and its counterpart are in the relevant resource set $J(t^{\text{start}}, t^{\text{end}})$. For example, suppose that we are planning for the demand over a one-week horizon from July 4, 00:00 to July 10, 23:59. The relevant resource of this horizon on a direct arc from a FC u to DS v are: $(u, v, \text{July 4}),\ (u, v, \text{July 5}), \cdots, (u, v, \text{July 11})$. The counterpart of resource $(u, v, \text{July 4})$ is $(u, v, \text{July 11})$ since they are one week apart, while other resources $((u, v, \text{July 5}),\ (u, v, \text{July 6}), \cdots, (u, v, \text{July 10}))$ on the link have no counterpart in

this planning horizon.

We create a set $J(j)$ for every resource j in set $J(t^{\text{start}}, t^{\text{end}})$. For any resource j that has a counterpart resource in set J, $J(j)$ contains the resource j and its counterpart. (In our example, $J((u, v, \text{July } 11)) = J((u, v, \text{July } 4)) = \{(u, v, \text{July } 4), (u, v, \text{July } 11)\}$.) For every resource j that does not have a counterpart resource in $J(t^{\text{start}}, t^{\text{end}})$, $J(j)$ only contains itself, i.e. $J(j) = \{j\}$. In summary, $|J(j)| = 1$ if the resource doesn't have a counterpart; $|J(j)| = 2$ if the resource has a counterpart; there is no other value of $|J(j)|$ for all $j \in J$. In the capacity planning problem, we charge the resources and their counterpart together. In addition, we create a set of resources (J') that has a counterpart resource in set J and its counterpart resource's CPT is one week earlier than itself. Then, we create a unique set of resources $\hat{J} = J/J'$ that represents a weekly capacity plan. The purpose of identifying $\hat{J}$ is to avoid double counting the cost of resources with counterparts in the objective function. We note that the composition of routes need not be changed since commodities arrive in the beginning and the end of the horizon despite charged to different route (and resources), should be charged together, which creates the desired "wrap-around" condition. For example, suppose the planning horizon is a week (from Monday to Sunday). Orders that arrive on Sunday might have feasible routes that utilize resources with next Monday timestamp. For our purpose, these routes need to be modified by replacing the resources with next week's timestamp by its counterpart resource with this week's timestamp. In this example, the resources with timestamp on next Monday will be replaced by the resource (with same physical attributes such as origin and destination) with timestamp on this Monday. With this change of route composition (,i.e., replacing out-of-horizon resources with resources within the horizon), we force the orders arrived over the finite horizon to be charged resources within the horizon.

Now, we are ready to formulate the capacity planning problem:

$$\min \quad \sum_{j \in \hat{J}} \pi_j n_j + \sum_{k \in K, r \in R^{tp} \cap R^{mixed}} c_{kr} x_{kr}$$

$$\text{s.t.} \sum_{r \in R_k} x_{kr} = d_k \quad \forall k \in K$$

$$\sum_{J(j)} \sum_{k \in K} \sum_{r \in R_k \cap R_j} x_{kr} \leq n_j u_j \quad \forall j \in \hat{J}$$

$$x_{kr} \geq 0 \quad \forall r \in R_k,\ k \in K$$

$$n_j \geq 0,\ \text{integer} \quad \forall j \in \hat{J}$$

Again, the objective is to minimize the transportation and processing costs, which are composed of truck costs, variable processing and third-party costs. The first constraint is the demand constraint, where commodities are matched with routes. The second constraint is the capacity constraint, where the sum of the utilization of resources within the counterpart set is less than the planned capacity.

The output of both formulations are the optimal assignments (x^*_{kr}) of commodities to routes, and the number of resources (n^*_j), for both arcs and nodes in the network. To be more specific, for the retailer-controlled resources, the capacity of a resource j is $n^*_j u_j$; for the third-party resources, the desired capacity to procure from the carrier is $\sum_{k \in K} \sum_{r \in R_k \cap R_j} x^*_{kr}$ for resource j

So far, we have provided a guideline for modeling and planning for a typical transportation network of the online retail. In the next chapter, we introduce the online algorithm that aims to make the best use of this given capacity plan by making smart order fulfillment decisions.

Chapter 4

The QP Algorithm for the Order Fulfillment Problem

In the previous chapter, we describe how an online retailer can plan its transportation resources ahead of time in light of its demand forecasts. This transportation plan then constrains or guides the real-time order fulfillment decisions for the online retailer. In real-time execution, the online retailer needs to choose dynamically a route to fulfill each customer's shipment in such a way that minimizes the incremental outbound shipping costs for the given transportation plan. This can be achieved by making order fulfillment decisions that efficiently balance the utilization of the retailer-controlled planned capacities with the third-party shipping options, as well as with the possibility of augmenting the capacity plan with ad hoc trucks.

In this chapter, we develop order fulfillment heuristics that make the online routing decision for each incoming shipment for a given transportation capacity plan. In addition, we develop heuristics for adding ad hoc trucks to the planned transportation resources.

4.1 Problem Formulation

We formulate the order fulfillment (route decision) problem as a dynamic programming problem that minimizes the incremental outbound shipping cost plus the ex-

pected future costs over a finite horizon. We assume that for each shipment, an upstream controller assigns an FC to fulfill the shipment. Hence, we now have a route decision problem. Each arriving shipment needs to be delivered from its assigned FC to a specific location by a promised time. The action space is the set of routes that can satisfy the shipment. The state of the system is defined by the remaining capacity of every resource, by the forecasts of the remaining demand for the planning horizon, and by the current time. We express the cost-to-go function as:

$$J(S, o) = \min_{r \in R(o,S)} (C(r, o) + E_{\tilde{o}}[J(f(S, r), \tilde{o})]), \tag{4.1}$$

where S is the system state, o is the incoming shipment, r is the route decision, $C()$ is the cost to fulfill shipment o by route r, $f()$ defines how the state evolves, $R(o, S)$ is the set of feasible routes of shipment o at state S. The expectation is taken over the next incoming shipment denoted by $\tilde{o}$. For our purposes, the salient features of a shipment are the origin FC, and the destination DS or service area, and the promise time. Another important feature is the physical size or volume of the order, which could affect third-party delivery costs and the volume occupied in trucks. However, for the current research, we do not consider this and assume all orders are uniform in volume and weight.

Solving the above dynamic program is intractable. The system state includes the remaining capacity on each resource, as well as demand for every pair of FC and DS (or service area). The dimension of the state space grows with the number of resources, which is on the order of 100's for small networks. A realistically sized problem over a one-week horizon may have as many as 100,000 resources.

To develop an operational heuristic, we approximate the cost-to-go function by the cost of a quadratic program. The objective of the quadratic program is to minimize the expected incremental cost to the transportation plan while satisfying the demand forecast over a finite planning horizon:

$$E_{\tilde{o}}[J(S, \tilde{o})] \approx W^{QP}(S). \tag{4.2}$$

where W^{QP} denotes the optimal value for the objective function of the quadratic programming problem. We will describe and develop the quadratic program (QP) in the next section. Based on this approximation, we consider making the routing decision by the following heuristic for shipment o, that arrives at state S:

$$r^* = \underset{r \in R(o,S)}{\arg\min}(C(r,o) + W^{QP}(f(S,r))) \tag{4.3}$$

The above heuristic requires solving one quadratic programming problem per feasible route per shipment; this is likely to require an unrealistic amount of computational time for realistic networks. Therefore, we make another approximation to further simplify the heuristic. We use the dual variables from the quadratic program to approximate the change in the QP's objective function from a route choice r for a given state S:

$$W^{QP}(f(S,r)) \approx W^{QP}(S) + \sum_{j \in J_r} \lambda_j. \tag{4.4}$$

where J_r is the set of resources that route r consumes and λ_j is the dual value for resource j, obtained from solving $W^{QP}(S)$. This is more computationally realistic, as at most one optimization is solved at each order occasion. Based on this approximation, we now choose the route r as follows:

$$r^* = \underset{r \in R(o,S)}{\arg\min}(C(r,o) + \sum_{j \in J_r} \lambda_j) \tag{4.5}$$

We note that in Chapter 2 we enforce a cap on the shadow prices to prevent excluding the assignment of retailer-controlled routes. This will not happen in the general framework we introduce in this chapter, due to the inclusion of unconstrained third-party options for each commodity. As a consequence, the third-party costs for the commodities will constrain the shadow prices on the associated resources.

4.2 The quadratic program formulation

The intent of the quadratic program is to develop an estimate of the expected incremental transportation costs over a finite horizon, given a transportation plan and a demand forecast. We use τ to denote the current time and the look-ahead period $[\tau, \tau + T^h]$, where T^h is the horizon length. In addition to the current time τ, the system state includes $\{d_k^\tau\}$, which are the forecasts for commodity k for the remaining demand in the look-ahead period $[\tau, \tau + T^h]$, and $\{\bar{f}_j^\tau\}$, which are the flow targets for resource j for the look-ahead period $[\tau, \tau + T^h]$. We will state first the formulation of the quadratic program, and then explain the motivation, as well as how we set the flow targets and the objective function coefficients. We note that for now we do not consider any options to use ad hoc trucks to increase transportation capacity. We first specify the notation as follows:

Sets

- K^τ: set of commodities for time horizon $[\tau, \tau + T^h]$
- J^τ: set of relevant resources for time horizon $[\tau, \tau + T^h]$
- R^τ: set of routes
- R_k: set of routes feasible to commodity k
- R_j: set of routes that utilize resource j

Parameters

- d_k^τ: demand forecast of commodity k at time τ
- $\bar{f}_j^\tau$: a flow target on resource j at time τ
- v_j^τ: penalty coefficient on resource j at time τ
- c_{kr}: per package cost of commodity k shipped through route r

Variables

- x_{kr}: number of shipments of commodity k assigned to route r
- g_j: excess flows on resource j

We now formulate the QP:

$$W^{QP}(\mathbf{d}, \bar{\mathbf{f}}, \tau) = \min \quad \frac{1}{2} \sum_{j \in J^\tau} v_j^\tau g_j^2 + \sum_{k \in K^\tau} \sum_{r \in R_k} c_{kr} x_{kr} \tag{4.6a}$$

$$\text{s.t.} \quad \sum_{r \in R_k} x_{kr} = d_k^\tau \quad \forall k \in K^\tau \tag{4.6b}$$

$$\sum_{r \in R_j} \sum_{k \in K_r} x_{kr} - g_j \leq \bar{f}_j^\tau \quad \forall j \in J^\tau \tag{4.6c}$$

$$x_{kr}, g_j \geq 0 \quad \forall k, r, j \tag{4.6d}$$

The first set of constraints (4.6b) ensures that the assignments satisfy the remaining demand for each commodity. For the second set of constraints (4.6c), we assume that we have a flow target for each resource. This flow target would typically be the amount of capacity remaining for the resource, net of some safety buffer to allow for demand uncertainty (or forecast errors). For planning purposes, we try to assign shipments to routes in a way that will not exceed the flow targets for each resource. However, in the QP we allow for the assignments to exceed these flow targets, which results in excess flow for that resource; the second set of constraints effectively define this excess flow (g_j) for each resource. We discuss in the next section how to set the flow targets ($\bar{f}_j^\tau$) for each resource j.

The objective function has two elements. The first (quadratic) term in the objective function approximates the expected incremental transportation cost that will be incurred if the assignments exceed the flow target on a resource. We discuss below a justification for the quadratic term, as well as how to set the penalty coefficient. The second (linear) term captures any variable costs associated with the assignment. To be more specific, the QP allows for every route assignment to incur a per package cost denoted by c_{kr}. For the retailer-controlled route, we assume that the transportation plan determines the number of trucks that traverse each transportation lane, and that the fixed cost for these trucks is the dominant cost for any shipment on these lanes.

Hence, we regard the cost for the middle-mile delivery (i.e., from FC to DS) as a sunk cost. In this case, we will use c_{kr} to capture only the last mile cost. For the routes that utilize a third party, we set c_{kr} to represent the per package cost charged by the third party.

We note that the parameters and sets depend on the current time τ and need to be updated at every resolve of the QP. We will discuss later in the chapter how frequently we re-solve the QP as well as how to execute the updates.

4.3 Justification of the Objective Function

4.3.1 Case 1: Stair-case Cost Structure

For the QP, we want to somehow account for uncertainty in the demand. Due to this demand uncertainty, there will be uncertainty in the assignment decisions, which results in uncertainty in the resource utilization.

We will use a simple example to motivate the QP formulation and will consider two cases that differ based on the cost assumptions. We consider a link or transportation lane, which corresponds to a resource j. For the first case, we suppose that for the given transportation plan, the actual incremental cost on the resource is given as follows (We will drop resource indices now until they become necessary.):

$$c(f) = \begin{cases} 0 & \text{for} \quad f \leq u \\ \pi & \text{for} \quad f > u \end{cases} \tag{4.7}$$

where f denotes the actual flow on the resource, and u denotes the planned capacity on the resource, according to the current transportation plan. Effectively, we are assuming that the cost for the planned capacity u is a sunk cost. Then π is the incremental cost if the actual flow exceeds the plan and might be set to the cost for adding additional capacity, e.g., an additional truck on the link. For instance, the current transportation plan assumes N trucks are scheduled on this link over the look-ahead period, which provides a capacity to ship u units; if the actual flow on the

link is above u, then the retailer will schedule an ad hoc truck and incur an additional cost for one more truck. To keep the presentation simple, we assume one truck will be sufficient to handle any excess flow.

Now, as of the current time τ, we view the flow on the resource on the look-ahead period as a random variable, denoted by F. We will not explicitly denote the dependence on τ, so as to not complicate the presentation. If we know the probability distribution of this random variable, then we can express the expected cost for this resource as:

$$\mathbb{E}[c(F)] = \int_{x=u}^{\infty} \pi\phi(x)dx = \pi\mathbb{P}[F > u], \tag{4.8}$$

where $\phi(x)$ denotes the probability density function. We propose to use a quadratic term in QP to approximate the expected cost in (4.8). This approximation depends on a number of assumptions. First, we assume that the expectation of the flow F_j on resource j will equal the corresponding solution from the QP. That is, we assume:

$$\mathbb{E}[F_j] = \sum_{r \in R_j} \sum_{k \in K_r} x_{kr} = f_j \tag{4.9}$$

where we define f_j as the planned flow of resource j from the solution to the QP. As an explanation, we expect the heuristic, given in equation (4.5), to guide the online decisions so that the actual flow on each resource aligns with the planned flow from the QP. By using the shadow prices from the QP, the heuristic adjusts the prices for different routes, so as to make assignments that are aligned to the optimal solution of the QP.

Second, we assume that we can estimate the standard deviation σ_j for the flow F_j on any resource j; we expect this can be done empirically. Third, in stating the QP, we propose to approximate the expected cost in equation (4.8) as follows:

$$\mathbb{E}[c(F_j)] = \int_{x=u_j}^{\infty} \pi_j\phi(x)dx = \pi\mathbb{P}[F_j > u_j] \approx \frac{v_j}{2}((f_j - \bar{f}_j)^+)^2 \tag{4.10}$$

where we have expressed the excess flow of resource j in terms of the assigned flow,

namely: $g_j = (f_j - \bar{f}_j)^+$. We will show some numerical examples later in this section to motivate this approximation. The final step of this approximation is to set the flow target ($\bar{f}_j$) and the penalty coefficient (v_j) for each resource j. We propose to do this by fitting the quadratic function through two points.

One point is $f_j = \bar{f}_j$ and the value of the RHS of (4.10) is zero. Thus, we then want to set the flow target $\bar{f}_j$ so that when the assigned flow equals the flow target ($f_j = \bar{f}_j$), the LHS of (4.10) also equals to zero or near zero. To do this, we see from equation (4.10) that we need to set the flow target so that $\pi\mathbb{P}[F_j > u_j] \approx 0$. We propose to set the flow target for resource j as $\bar{f}_j = u_j - z\sigma_j$ for a safety factor z. Then when $f_j = \bar{f}_j$, we have by assumption that $\mathbb{E}[F_j] = f_j = \bar{f}_j = u_j - z\sigma_j$; therefore, $\mathbb{E}[F_j] + z\sigma_j = u_j$. Thus, we will set z so that the probability of exceeding the capacity is likely small, which results in the LHS of (4.10) being near zero. For instance, suppose that we assume the F_j has a normal distribution, and suppose that we set the safety factor $z = 2$; then the probability of exceeding the capacity is 0.023.

For the other point, we suppose that the assigned flow equals the planned capacity, $f_j = u_j$ and thus, $\mathbb{E}[F_j] = u_j$. If we assume that the probability distribution of F_j is symmetric, then the expected cost in (4.8) is $\pi_j/2$, equal to the LHS of (4.10) for $f_j = u_j$. To equate the RHS of (4.10) to $\pi_j/2$ at $f_j = u$, we note that for $\bar{f}_j = u_j - z\sigma_j$, we have $f_j - \bar{f}_j = z\sigma_j$. Thus, we have $\frac{\pi_j}{2} = \frac{v_j(z\sigma_j)^2}{2}$ and find that we should set $v_j = \frac{\pi_j}{(z\sigma)^2}$.

In summary, for incremental cost given by (4.8), we set the parameters in the QP for each resource as:

$$\begin{aligned} \bar{f}_j &= u_j - z\sigma_j \\ v_j &= \frac{\pi_j}{(z\sigma_j)^2} \end{aligned} \tag{4.11}$$

Here, we observe that there is a dependence on the current time τ. In particular, we expect that our estimate of the standard deviation of the flow on the resource will depend on the current time, or more specifically, on the remaining time for the resource. We denote this dependence as σ_j^τ. To be consistent with this, we then

express the parameters as:

$$\begin{aligned} \bar{f}_j^\tau &= u_j^\tau - z\sigma_j^\tau \\ v_j^\tau &= \frac{\pi_j}{(z\sigma_j^\tau)^2} \end{aligned} \tag{4.12}$$

To implement, we will need to estimate this standard deviation for each resource at each re-solve time, as well as set a safety factor; in the computational work we will experiment with these settings.

We illustrate this approximation paradigm through a simple numerical example. Consider a network that has a single resource that is feasible to all the incoming demands and that there is no third-party available. The capacity of the resource is $u = 100$, and additional capacity can be added to the resource at a cost of $\pi = 10$ for additional 10 units. The flow on the resource is a normally distributed random variable $F \sim (\mu, \sigma)$. The expected cost at different μ with fixed standard deviation ($\sigma = 10$) can be computed with equation (4.8). The result is plotted in dotted line in Figure 4-1.

Next, we calculate the quadratic cost approximation at different μ to examine its resemblance to the expected cost. According to our approximation paradigm, the quadratic cost of the resource is: $\frac{1}{2}v((f - \bar{f})^+)^2$, where v is the penalty cost, f is the optimal solution of QP and $\bar{f}$ is the flow target of the resource. We set v and $\bar{f}$ according to (4.12) with $z = 2$ and $\sigma = 10$, which gives us $\bar{f} = u - z\sigma = 80$, and $v = \pi/(z\sigma)^2 = 0.025$. The quadratic cost at different μ is illustrated with solid line in Figure 4-1.

From Figure 4-1, we observe that the approximation is quite good up to $\mu = 100$, the capacity level; beyond this, the approximation deviates quite a bit with increased values of μ. (As a side note, we expect the most relevant range is up to and around the capacity level.) In Figure 4-2, we vary the safety factor and the standard deviation of flows to show how these parameters in the quadratic function affect the approximation.

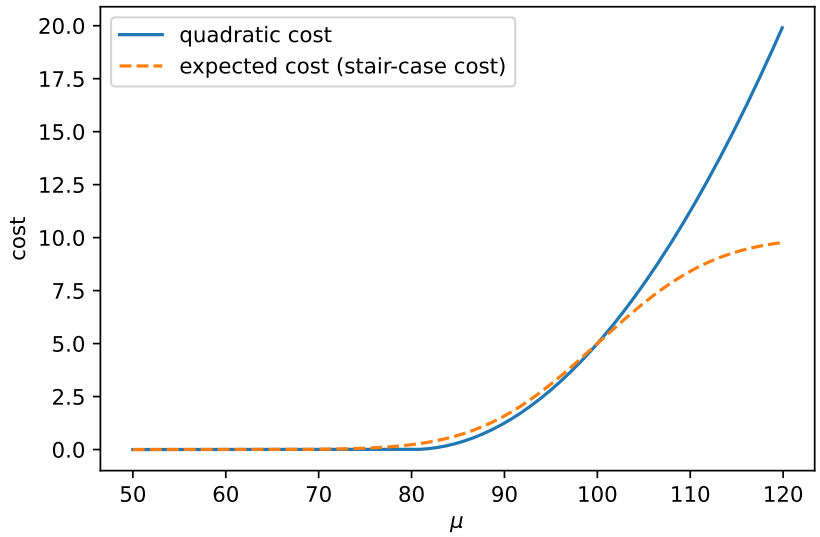


Figure 4-1: Resemblance of quadratic cost and the expected costs when cost is a staircase function.

4.3.2 Case 2: Linear Cost Structure

In this section, we will repeat the development from the prior section, but now with a different assumption on the incremental cost. We consider a resource j, which could be a link or transportation lane, or a network node. For this case, we suppose that for the given transportation plan, the actual incremental cost on the resource is given as follows (we will drop resources indices now until they might become necessary):

$$c(f) = \begin{cases} 0 & \text{for} \quad f \leq u \\ \pi(f-u) & \text{for} \quad f > u \end{cases} \tag{4.13}$$

where f denotes the actual flow on the resource, and u denotes the planned capacity on the resource, according to the current transportation plan. Again, we are assuming that the cost for the planned capacity u is a sunk cost. Now, however, the incremental cost when the actual flow exceeds the capacity, is linear in the excess flow; the cost factor π is the cost per shipment for fulfilling each shipment in excess of the capacity. For instance, the current transportation plan might assume N trucks are scheduled on this link over the look-ahead period, which provides a capacity to ship u units;

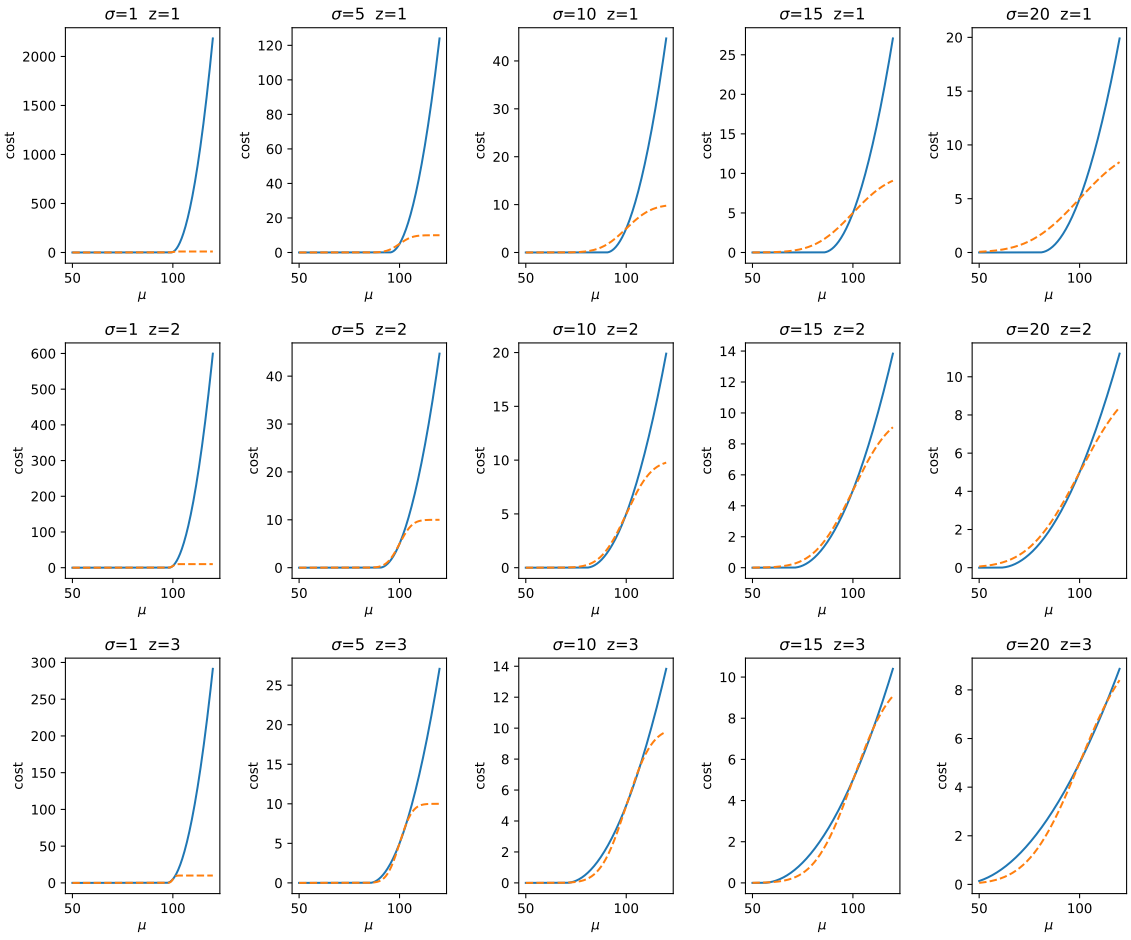


Figure 4-2: Resemblance of quadratic cost and expected ad hoc cost with different values of standard deviation and safety factors. (case 1)

if the actual flow on the link is above u, then the retailer will assign each excess shipment to a third-party carrier, at an incremental cost of π per shipment. To keep the presentation simple, we assume the third party has sufficient capacity to handle any excess flow.

Now, as of the current time τ, we view the flow on the link on the look-ahead period as a random variable, denoted by F. We do not make explicit this dependence on τ, to simplify the presentation. If we know the probability distribution of this random variable, then we can express the expected cost for this link as:

$$\mathbb{E}[c(F)] = \int_{x=u}^{\infty} \pi(x-u)\phi(x)dx, \tag{4.14}$$

where $\phi(x)$ denotes the probability density function. We propose to use a quadratic term in QP to approximate the expected cost in equation (4.14). This approximation depends on the same assumptions as for the first case. First, we assume that the expectation of the flow F_j on resource j will equal the corresponding solution from the QP. That is, we assume:

$$\mathbb{E}[F_j] = \sum_{r \in R_j} \sum_{k \in K_r} x_{kr} = f_j \tag{4.15}$$

where f_j is the planned flow of resource j from the solution to the QP. As explanation, we will use the QP to determine the shadow prices or dual values for the resources. We expect that these values will try to guide the decisions so that the actual flow on each resource aligns with the planned flow. Thus, we expect that the heuristic will result in the planned flow being a good estimate of the expected flow on a resource.

Second, we assume that we can estimate the standard deviation σ_j for the flow F_j on the resource; we expect this can be done empirically. Third, in stating the QP, we propose to approximate the expected cost in equation (4.14) as follows:

$$\int_{x=u_j}^{\infty} \pi_j (x - u_j) \phi(x) dx \approx \frac{v_j}{2} ((f_j - \bar{f}_j)^+)^2 \tag{4.16}$$

where we have expressed the excess flow of resource j in terms of the assigned flow, namely: $g_j = (f_j - \bar{f}_j)^+$.

Similar to the first case, we propose to set the penalty coefficient for each resource by fitting the quadratic function through two points. One point is when the planned flow equals exactly the flow target $f_j = \bar{f}_j$ and the value of the RHS of (4.16) is zero. We then want to set the flow target $\bar{f}_j$ so that the LHS of (4.16) also equals to zero or near zero. To do this, we propose to set the flow target for resource j as $\bar{f}_j = u_j - z\sigma_j$ for a safety factor z. Then $f_j = \bar{f}_j$ implies that $\mathbb{E}[F_j] = f_j = \bar{f}_j = u_j - z\sigma_j$, and therefore, $\mathbb{E}[F_j] = u_j - z\sigma_j$. With the assumption that the flow is normally distributed,

the LHS of equation (4.16) becomes:

$$\int_{x=u_j}^{\infty} \pi_j \phi(x-u_j)dx = \pi_j \sigma_j L(\frac{u_j - \mathbb{E}[F_j]}{\sigma_j}) = \pi_j \sigma_j L(z), \tag{4.17}$$

where $L()$ is the loss function for the standard normal random variable. The above expression is near zero for typical values for the safety factor. As an example, with $z = 2$, $\mathbb{E}[c(F_j)] = 0.008\pi_j\sigma_j$. We will treat this as effectively being zero.

For the other point, suppose $f_j = u_j$ and thus, $\mathbb{E}[F_j] = u_j$. The RHS of (4.16) is $\frac{1}{2}v_j(\sigma_j z)^2$. With the assumption that the flow is normally distributed, the LHS of (4.16) is:

$$\int_{x=u_j}^{\infty} \pi_j \phi(x-u_j)dx = \pi_j \sigma_j L(\frac{u_j - \mathbb{E}[F_j]}{\sigma_j}) = \pi_j \sigma_j L(0) = 0.399\pi_j\sigma_j. \tag{4.18}$$

By equating the two, we find that $v_j = \frac{0.798\pi_j}{z^2\sigma_j}$. In summary, for incremental cost given by (4.14), we set the parameters in the QP for each resource as:

$$\begin{aligned} \bar{f}_j &= u_j - z\sigma_j \\ v_j &= \frac{0.798\pi_j}{z^2\sigma_j} \end{aligned} \tag{4.19}$$

Again, we expect that our estimate of the standard deviation of the flow on the resource will depend on the current time (τ), or more specifically, in the remaining time for the resource. We denote this dependence as σ_j^τ. To be consistent with this, we express the parameters as:

$$\begin{aligned} \bar{f}_j^\tau &= u_j^\tau - z\sigma_j^\tau \\ v_j^\tau &= \frac{0.798\pi_j}{z^2\sigma_j^\tau} \end{aligned} \tag{4.20}$$

We test this approximation paradigm on a one-resource network, which is the same as the example in case 1 but with different cost structure. The single resource has capacity $u = 100$, and additional capacity can be added incrementally by unit: one additional unit cost 5 dollars ($\pi = 5$). The flow of the resource is, again, a normally distributed random variable $F \sim (\mu, \sigma)$. The expected cost with the linear cost struc-

ture can be estimated by equation (4.14) given the mean (μ) and standard deviation (σ) of the flow. The dotted line in Figure 4-3 shows the expected cost with different μ with fixed standard deviation ($\sigma = 10$).

Next, we calculate the quadratic cost (the objective function of the QP) and examine its resemblance to the expected cost. According to our approximation paradigm, the quadratic cost of the resource is: $\frac{1}{2}v((f - \bar{f})^+)^2$, where v is the penalty costs, f is the optimal solution of QP and $\bar{f}$ is the flow target of the resource. By certainty equivalence, the expected cost of the quadratic function is $\frac{1}{2}v((\mu - \bar{f})^+)^2$. We set v and $\bar{f}$ according to (4.19) with $z = 2$ and $\sigma = 10$, which gives us $\bar{f} = u - z\sigma = 80$, and $v = 0.798\pi/z^2\sigma = 0.09975$. The expected quadratic cost at different μ is illustrated with a solid line in Figure 4-3. The approximation works well when μ is under the capacity ($u = 100$). In Figure 4-4, we vary the safety factor and the standard deviation of flows to show how different parameters in the quadratic function affects the approximation.

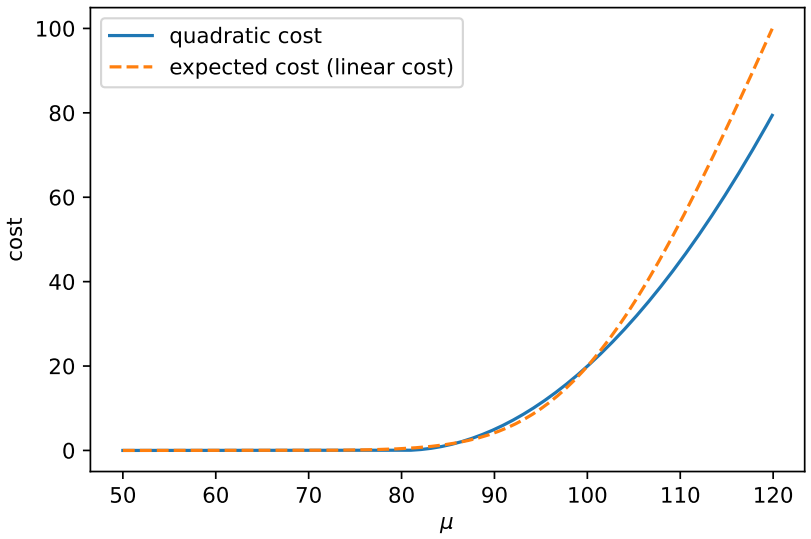


Figure 4-3: Resemblance of quadratic cost and the expected costs when cost is a linear function (case 2).

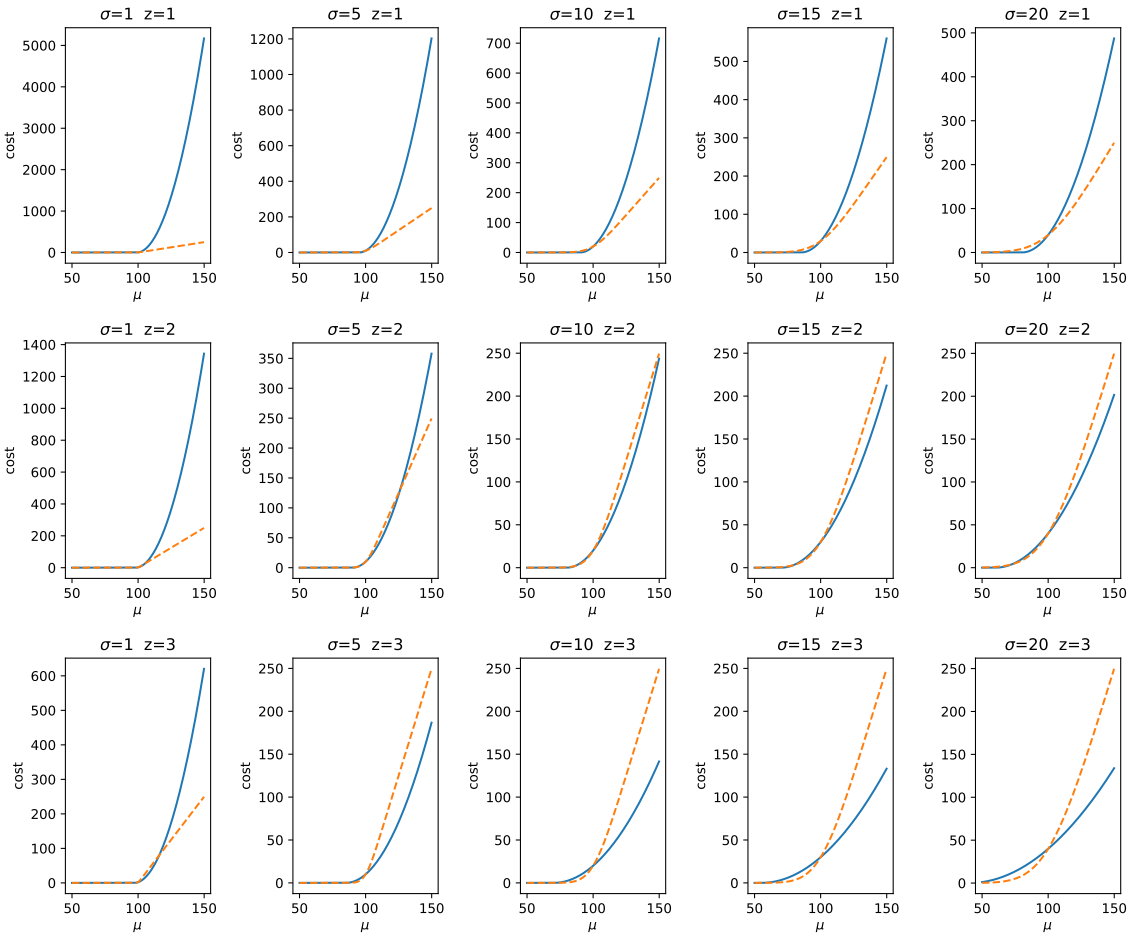


Figure 4-4: Resemblance of quadratic cost and expected ad hoc cost with different values of standard deviation and safety factors. (case 2)

4.4 Summary of the Order Fulfillment Algorithm

To apply our order fulfillment algorithm, one needs to decide the frequency of shadow price update and the length of the time horizon (T^h) of the quadratic programming problem. We will explore this decision empirically. At a high-level, we note that the length of the time horizon (T^h) should at least cover resources that are "relevant" to the current instance. Furthermore, we expect that the more frequent the shadow prices are updated, the more "accurate" will be the values of resources; thus, we expect a marginal improvement as the update frequency increases.

The next step is to construct the quadratic programming problem according to the network structure and demand forecasts, which includes identifying relevant resources,

routes, and commodities. Depending on the cost structure of the resources, we pick one of the two approximation paradigm to set the QP parameters (target flows and penalty coefficients).

In real-time execution, order fulfillment decisions are based on the most updated shadow prices, where routes are picked according to (4.5). Shadow prices are updated in the background at a desired frequency with the QP, where the parameters (including forecasts, capacity of resources... etc.) are being updated at every resolve.

In the next chapter, we test our algorithm in a large network inspired from real-world data collected by our industrial partner.

4.5 Hindsight Solution of the Order Fulfillment Problem – A Benchmark

We provide a simple benchmark for evaluating online order fulfillment algorithms. In the next chapter, we test different algorithms over a finite horizon, and evaluate their performance by average per package costs of the fulfilled packages over the finite horizon. The hindsight solution assumes that the shipment arrivals are known, and then we can solve the following transportation LP with resource capacities relevant to the one-day horizon:

$$\min \quad \sum_{s\in S}\sum_{r\in R_s} c_{sr}x_{sr} \tag{4.21a}$$

$$\text{s.t.} \quad \sum_{r\in R_s} x_{sr} = 1 \quad \forall s \in S \tag{4.21b}$$

$$\sum_{r\in R_j}\sum_{s\in S} x_{sr} \le u_j \quad \forall j \in J \tag{4.21c}$$

$$x_{sr} \text{ is binary} \quad \forall s, r \tag{4.21d}$$

notations

- S: set of shipments

- R: set of routes
- J: set of resources
- R_s: set of routes feasible to shipment s
- R_j: set of routes contains resource j
- x_{sr}: binary variable, 1 if shipment s is assigned to route r, 0 if not.
- c_{sr}: cost of shipping shipment s by route r
- u_j: capacity of resource j

The total cost of the hindsight solution is the objective of the transportation LP. We note that the hindsight solution is the best benchmark for any online fulfillment algorithm. In fact, we can aggregate the shipments into commodities where shipments in each commodity has the same set of feasible routes, and the hindsight solution can be solved at commodity-level instead of at shipment-level.

4.6 Ad hoc Controllers

So far, we have been assuming that the online-retailer has a planned-ahead capacity plan that is fixed over a finite horizon. In reality, when the demand deviates from the planned capacity by a lot, adding/removing planned resources becomes a crucial cost-saving mechanism. Motivated by this, we propose two ad hoc truck controllers that make online capacity modification decisions by adding/removing ad hoc trucks to arc resources. We note that the cost for adding a truck is greater than the cost for planned trucks, which are scheduled in advance. The controllers are not tested numerically in the realistic network presented in the next chapter, but this mechanism is an important avenue for future research.

4.6.1 The Threshold-based Controller

We assume that an additional truck to arc resource j provides a capacity of u_j^{ah} units and costs c_j^{ah}. We propose to trigger these ad hoc truck decisions with a threshold that reflects the cost of shipping excess flows by third-party. If the cost of adding an ad hoc truck is cheaper than the projected third-party cost, then we add an ad hoc truck. At any time t we add an ad hoc truck to the arc if the following inequality holds:

$$c_j^{3p}(f_j(t) - u_j(t))^+ > c_j^{ah}$$

where c_j^{3p} denotes the alternative (third-party) cost on the arc resource j, $f_j(t)$ denotes the current forecast of expected remaining flows on the arc resource j at time t, $u_j(t)$ denotes the remaining capacity at time t on the arc resource j. The left-hand side of the inequality is the projected cost if we do not add any more capacity on this arc. We note here that the specification for c_j^{3p} is not obvious for many middle mile arcs (e.g., an arc from an FC to a SC), but should depend upon the third-party costs (of various origin-destination pairs) in the transportation network. We provide two possible ways of estimating this cost in Appendix C for general networks. The right-hand side is the cost for adding a truck, where we implicitly assume the truck will have sufficient capacity to handle the remaining demand. Hence, the rule is to add a truck if its cost is less than the projected cost for doing nothing. We can also consider the possibility of canceling a scheduled ad hoc truck. We might remove a truck when the expected cost of shipping the remaining demand with capacity without the ad hoc truck is less than the ad hoc truck cost. In other words, we remove a truck on arc resource j at time t when the following inequality holds:

$$c_j^{3p}(f_j(t) - (u_j(t) - u_j^{ah}))^+ < c_j^{ah}.$$

Note that we allow a cancellation of an ad hoc truck only when no shipments have been assigned to the truck.

4.6.2 The DP-based Controller

The threshold-based controller makes local ad hoc truck decisions for each arc, i.e., the decision of adding/removing resources of an arc resource does not depend on the decision of other arc resources. We describe here a second heuristic that makes ad hoc truck decision globally for every arc resource. At any time instance t, we can add an ad hoc truck to any arc resource, or remove an ad hoc truck, or do nothing. Adding (removing) an ad hoc costs c_j^{ah} ($-c_j^{ah}$), and there is no cost for doing nothing. Therefore, there are 3 possible actions at any time instance for an arc. We formulate the ad hoc truck problem as a dynamic program that minimizes immediate cost of the ad-hoc decision plus the resulting future expected cost:

$$J(\mathbf{S},t) = \min_{\mathbf{a}\in\{-1,0,1\}^M} \{c(\mathbf{a}) + J(f(\mathbf{S},\mathbf{a}),t)\} \tag{4.22}$$

where t is the time of decision instance, $\mathbf{S}$ is the system state (which includes remaining capacity ($\mathbf{u}$) and a forecast of the remaining demand ($\mathbf{d}$)), $\mathbf{a}$ is a vector (of -1,0 or 1) with size M that denotes the ad hoc truck decision, $c()$ is the cost of action a, $f()$ defines how state S evolves from a given action a, $\{-1,0,1\}$ denotes the three possible actions (i.e., removing a truck, doing nothing or adding a truck), M is the number of arcs that allows addition and removal of trucks. We propose to solve an approximation of the DP by approximating the cost-to-go function ($J(\mathbf{S},t)$) by the objective function $W^{AH}(\mathbf{S},t)$ of a linear programming problem that minimizes shipping costs while satisfying demand and capacity constraints:

$$W^{AH}(\mathbf{S},t) = \min \quad \sum_{k\in K^t}\sum_{r\in R_k} c_{kr}x_{kr} \tag{4.23a}$$

$$\text{s.t.} \quad \sum_{r\in R_k} x_{kr} = d_k^t \quad \forall k \in K^t \tag{4.23b}$$

$$\sum_{r\in R_j}\sum_{k\in K_r} x_{kr} \le u_j(t) + a_j u_j^{ah} \quad \forall j \in J^t \tag{4.23c}$$

$$x_{kr} \ge 0 \quad \forall k,r \tag{4.23d}$$

where u_j^{ah} is the capacity of an ad hoc truck. In theory, we would need to solve the problem for all possible $\mathbf{a}$ to approximate the cost-to-go function for these actions, and then apply the solution to equation 4.22 and make the ad hoc decision by picking the cheapest option. Note that the action space grows exponentially as the number of resources, whereby we need to solve the LP 3^M times in order to make an ad hoc truck decision. However, because $c(\mathbf{a})$ is linear in the action space, we can approximate $J(\mathbf{S}, t)$ by the solution of the following MIP:

That is, this method is equivalent to solving a mixed-integer program that contains ad-hoc truck decision variables. To be more specific, solving the following mixed-integer program results in the same ad hoc decisions:

$$\min \quad \sum_{k \in K^t} \sum_{r \in R_k} c_{kr} x_{kr} + \sum_{j \in J^t} c_j^{ad} a_j \tag{4.24a}$$

$$\text{s.t.} \quad \sum_{r \in R_k} x_{kr} = d_k^t \quad \forall k \in K^t \tag{4.24b}$$

$$\sum_{r \in R_j} \sum_{k \in K_r} x_{kr} \leq u_j(t) + a_j u_j^{ah} \quad \forall j \in J^t \tag{4.24c}$$

$$x_{kr} \geq 0 \quad \forall k, r \tag{4.24d}$$

$$a_j = \{-1, 0, 1\}, \tag{4.24e}$$

where the decision variable a_j correspond to the ad hoc truck decisions on each arc j. However, we note that this equivalence requires the convexity of the ad hoc truck cost function. In our case, the cost function is linear, therefore, the method can be simplified to solving (4.24) in one shot instead of solving multiple (4.23) for different ad hoc decisions.

Chapter 5

Experiments on Realistic Networks

To test the practicality and performance of our work, we conduct experiments on a series of numerical examples modeled after a subnetwork of an online retailer's fulfillment network. We replay the historical order arrivals (or variants of it) to test our fulfillment algorithm in comparison to other benchmark algorithms. We measure their performances with the total transportation cost to fulfill all orders that arrive within a fixed time horizon. Our code is made publicly available at `https://github.com/PinyiChen/Order_Fulfillment_Dynamic_Route_Selection`.

In section 5.1, we provide details on the setup of a numerical example, which we refer to as the "base-case", and on the historical shipment data. Then, we analyze the performance of our algorithm against a benchmark (greedy) algorithm. The "base case" is deliberately simplified as the intent of the analysis is to get a more detailed understanding of the QP's behavior. In section 5.2 we create another test case that has a more complicated network topology than the base case. This test case has two SCs, which results in a larger set of routes for each order; the test case also has multiple service areas for each DS, which results in additional trade-offs between third-party options and retailer–controlled routes. Since this test case is more general and is closer to reality, we perform more extensive sensitivity tests on the test case and compare the performance with the "LP algorithm", which is an algorithm that mimics the online-retailer's current operation.

5.1 The Base Case

5.1.1 Identifying a Self-contained Subnetwork

The experiment is performed on a self-contained subnetwork, which is composed of 13 FCs, 1 SC, 5 DSs and 164 service areas. A service area is a geographic region that is served by a dedicated DS, and for which the third-party delivery cost to customers in the service area is considered the same. Table 5.1 shows the number of service areas that each DS serves. For this system, we had access to the entire set of orders that were assigned to the SC over a 17-day period in 2020. For every order, we had the following information: the **FC** from which the order was shipped, the **set of feasible routes** for the order, and the **route** (chosen from the feasible set) that was assigned to the order. From this 17-day order assignment data, we find that 95% of the shipments handled by the SC came from the 13 FCs and were destined to the 5 DSs. Thus, a large percentage of the shipments in the data set utilized the transportation resources in the subnetwork. The 164 service areas are chosen based on two conditions: each service area was primarily served by one of the 5 DSs, and there were more than 100 shipments to the service area over the 17-day horizon.

DS	number of service areas
DS1	18
DS2	79
DS3	25
DS4	30
DS5	12

Table 5.1: Number of service areas each DS serves

Next, we identify the network topology (arcs between nodes) from the historical shipment data set. An arc resource is considered "active" if the resource was part of a feasible route for at least one shipment. In this 13-FC, 1-SC, 5-DS network, every FC has an active arc to the SC, and every DS has an active arc from the SC; however, only a subset of the possible direct (FC to DS) arcs are active. The 9 active direct arcs can be found in table 5.2, which are the FC-DS pairs with non-zero values. We

assume that a third-party option is available from every FC and from the SC to every destination. In the next section, we build a base case from the 13-FC, 1-SC, 5-DS network.

5.1.2 Input Creation

We select a single day of shipments for testing the order fulfillment algorithms. This shipment data set contains shipments that satisfy the following conditions:

- The shipment "arrived" [1] within a one-day horizon (between 00:00:00 and 23:59:59 on a chosen day)
- The shipment was shipped from one of the 13 FCs
- The shipment was delivered to one of the 164 service areas

We reduce the number of shipments by a scale factor, and the total number of shipments is 11,518 after reduction. Each shipment comes with arrival time, origin FC, destination service area, and the DS associated with the service area. We assume that each service area is assigned to a single DS for service (which corresponds to reality), and that the third-party delivery costs are the same for all service areas served by a DS (which is a simplification that we will relax with the test case in section 5.2). We also ignore the physical volume of the order for simplicity. Based on this one-day worth of shipments, we create the inputs for the base case, which include relevant resources, routes, demand forecasts and shipment arrivals. In this section, we provide a detailed description of these inputs.

5.1.2.1 Resources and Capacities

As we mentioned in section 3.1.1, retailer-controlled resources are defined by facilities and timestamps. For simplicity, we assume that every arc resource has a single CPT each day:

- FC-SC resources daily CPT: 5:00

[1] The arrival time is defined by the time a shipment is assigned to a FC by an existing controller.

- SC-DS resources daily CPT: 15:00
- FC-DS resources daily CPT: 12:00

We consider resources with CPTs on two days – day 1, and day 2, with the delivery promise time being midnight of day 2 for all orders. Therefore, each arc has two resources in the one-day horizon experiment. For instance, for an arc between FC u and SC v, there are two resources, one with a timestamp (u, v, 5:00 day 1) and the other with timestamp (u, v, 5:00 day 2). For the node resources, there may be a capacity restriction related to the node that are time dependent. However, in this example, we are assuming the capacity of the node resources are infinity so there is no need to create time stamps for the nodes.

We set the capacity of the arc resources based on the quantity of the shipment data. For each arc type, we create the capacity with different rules:

- FC-SC resources: We take 70% of the demand forecast from the associated FC and split this by day one and day two with 2-8 ratio. For instance, if demand forecast from a FC u is 200 units, then we split 140 units (70% of 200) between the day one and day two resources with a 2-8 ratio. Thus, the capacity of the arc between FC u and SC v is 28 units for the first day resource (u, v, 5:00 day 1), and 112 units for the second day resource (u, v, 5:00 day 2).

- FC-DS resources: We take 30% of the demand forecast from the FC to the DS, and then split this by day one and day two with 2-8 ratio. For instance, if demand forecast from a FC u to DS w is 200 units, then the capacity of the arc between FC u and SC w is 12 units for the first day resource (u, w, 12:00 day 1), and 48 units for the second day resource (u, w, 12:00 day 2).

- SC-DS resources: We set the capacity for each SC to DS resource equal to the demand forecast for the DS for both day one and day two; in effect, we do not constrain this resource in the base case.

We deliberately set the resource capacities to only constrain the upper and direct arc resources, so that the other resources in the network are not constraining. This

allows us to focus on the resource allocation on the upper and direct arcs. Table 5.2 summarize the capacities of FC-SC and FC-DS resources. Furthermore, we set no capacity limits on third-party resources in this test case.

	upper arc	direct arc					
FC\SC or DS	SC1	DS3	DS2	DS1	DS4	DS5	Total
FC1	705	0	0	52	0	0	757
FC2	669	0	0	0	0	0	669
FC3	649	0	0	0	0	0	649
FC4	1685	135	352	56	145	0	2373
FC5	80	0	0	0	0	0	80
FC6	991	0	158	0	96	26	1271
FC7	364	0	0	0	0	0	364
FC8	297	0	0	0	0	0	297
FC9	339	0	0	0	30	0	369
FC10	244	0	0	0	0	0	244
FC11	1135	0	0	0	0	0	1135
FC12	380	0	0	0	0	0	380
FC13	524	0	0	0	0	0	524

Table 5.2: Resource Capacity of each upper (FC-SC) and lower (FC-DS) arc. We note that the arcs with 0 capacity are not active.

5.1.2.2 Routes

For simplicity, we consider only three types of routes – direct, indirect and third-party route. (We omit the mixed route, which will be considered in the later section.) As described in section 3.1.2, a route is feasible as long as the resources that are required are time compatible. In this base case, we create the routes assuming that dwell and transit times are effectively zero ($\epsilon = 1$ second). The estimated arrival time (EAT) of retailer-controlled routes are the last resource's CPT plus ϵ. We note that since the node resources have unlimited capacity, the node resources can be omitted from route formation. In other words, indirect routes are composed of a FC-SC arc resource and a SC-DS arc resource, the direct routes only have a FC-DS arc resource. In addition, all the routes we form for the base case have an EDT earlier than the promise (day 2, midnight), i.e., all the routes can deliver all the shipments on time. In this base

case, there are 148 retailer-controlled routes, which includes 65 indirect routes and 9 direct routes on day one and day two each.

5.1.2.3 Commodities and Forecasts

In section 3.2.1, we introduced the notion of commodities, which is an aggregated group of shipments for the purpose of demand forecasts. Each commodity is defined by an origin FC, a destination DS, a dimension group, an arrival time segment and a promise time. For the base case, we assume that there are 13 FCs, 5 DSs, 1 dimension group, 3 arrival time segments (day 0, midnight[2] to 5AM, day 1, 5AM day 1 to noon and day 1, noon to day 1, midnight), 1 promise time (day 2, midnight). We note that these arrival time segments are created based on the first resource CPT of the routes, such that orders that arrive within an arrival time segment have the same set of feasible routes. For example, consider an origin FC and a destination DS, for which there are four feasible retailer-controlled routes – two direct and two indirect routes. The first resource CPT for the two direct routes are day 1, noon and day 2, noon. The first resource CPT for the two indirect routes are day 1, 5AM and day 2, 5AM. All four routes can meet the promise time (day 2, midnight). Then any demand for this FC, DS pair that arrives before day 1, 5AM can be assigned to any of the four routes; any demand that arrives between day 1, 5AM and day 1, noon has three feasible routes (two direct routes; and the second-day indirect route); any demand that arrives between day 1, noon and day 2, 5 AM has two feasible routes (the second-day indirect route and the second-day direct route).

Next, we create the demand forecasts for each commodity. For each (FC, DS) pair, we set its daily demand forecast equal to the shipment data for day 1. Table 5.3 summarize the demand forecasts of commodities.

Then, we distribute the daily forecast of each (FC, DS) pair to its associated commodities proportional to the length of the arrival time segment. For example, suppose a (FC, DS) pair has a daily forecast of 100 units, and we have 3 arrival time

[2]By convention we associate midnight with the preceding day; so day X, midnight means the end of day for day X

FC\DS	DS3	DS2	DS1	DS4	DS5	Total
FC1	187	365	175	210	71	1008
FC2	220	323	77	281	55	956
FC3	224	336	82	232	53	927
FC4	452	1174	188	484	109	2407
FC5	26	69	0	15	5	115
FC6	362	528	121	320	86	1417
FC7	98	235	59	96	32	520
FC8	86	184	37	90	28	425
FC9	96	202	54	100	33	485
FC10	82	143	40	61	22	348
FC11	268	763	118	356	116	1621
FC12	108	228	64	109	33	542
FC13	435	234	0	58	21	748
Total	2644	4784	1015	2412	664	

Table 5.3: Demand forecast of each FC-DS pair

segments (day 0, midnight to day 1, 5AM, day 1, 5AM to noon and day 1, noon to day 1, midnight). The first commodity with a 5-hour arrival time segment has a forecast of $\frac{5}{24}100$ units, the second commodity with a 7-hour arrival time segment has a forecast of $\frac{7}{24}100$ units, the third commodity with arrival time segment day 1, noon to midnight has a forecast of $\frac{12}{24}100$ units. For each commodity, we identify the set of feasible routes by checking the three conditions listed in section 3.2.1.

5.1.2.4 Costs

We assume that the cost of retailer-controlled node and arc resources are sunk (paid in advance) ; therefore, every retailer-controlled route has a \$0 cost. However, any third-party costs are real costs. We estimate the cost of third-party routes by the distance between the origin FC and the destination DS with the following linear equation:

$$\text{cost} = \$0.01 \times \text{distance (mile)} + \$2 \tag{5.1}$$

We note that the distance is estimated based on the longitude and latitude data of the facilities with the Haversine formula, which determines the great-circle distance

between two points. The algorithm's performance is evaluated by the total cost incurred over the one-day horizon, which consists of just the third-party costs in this case. Table 5.4 summarize the third-party costs for each (FC, DS) pair.

FC\DS	DS3	DS2	DS1	DS4	DS5	Avg
FC1	3.43	3.12	2.92	4.03	3.90	3.48
FC2	4.34	4.04	3.78	4.95	4.82	4.39
FC3	4.03	4.27	4.62	4.54	4.49	4.39
FC4	2.15	2.30	2.80	2.72	2.60	2.51
FC5	2.06	2.46	2.96	2.61	2.50	2.52
FC6	2.12	2.38	2.88	2.79	2.68	2.57
FC7	5.48	5.14	4.81	6.06	5.93	5.48
FC8	2.09	2.47	2.97	2.75	2.64	2.58
FC9	2.08	2.50	3.00	2.60	2.50	2.54
FC10	4.96	5.40	5.90	4.87	4.93	5.21
FC11	2.17	2.62	3.12	2.64	2.55	2.62
FC12	6.02	5.72	5.41	6.63	6.50	6.06
FC13	6.00	6.44	6.94	5.76	5.85	6.20
Avg	3.61	3.76	4.01	4.07	3.99	3.89

Table 5.4: Third party costs of each FC-DS pair

5.1.2.5 Parameters for the QP Algorithm

In implementing the QP algorithm, we will periodically resolve the QP in order to update the shadow prices. At each shadow price update time instance τ, we need to specify the target flows ($\bar{f}_j^\tau$) and penalty costs (v_j^τ) for each resource, and the updated demand forecasts (d_k^τ) for each commodity, which are inputs for the QP. As developed in section 4.3.2, we set the target flows ($\bar{f}_j^\tau$) and penalty costs (v_j^τ) by the following formulas:

$$\begin{aligned} \bar{f}_j^\tau &= u_j^\tau - z\sigma_j^\tau \\ v_j^\tau &= \frac{0.798\pi_j}{z^2\sigma_j^\tau} \end{aligned} \tag{5.2}$$

The incremental costs (π_j) captures the per package cost for exceeding capacities on each resource. We estimate the incremental cost for arc resources by:

$$\pi_j = \gamma \times (\$0.01 \times \text{distance} + \$2)$$

where $(0.01 \times \text{distance} + 2)$ is the formula for deriving third-party costs and γ is the discount factor of shipping with retailer-controlled resources versus shipping through third-party. In the base case, we set $\gamma = 1$. We set the standard deviation by: $\sigma_j^\tau = \alpha u_j^\tau$, where u_j^τ is the remaining capacity of resource j at resolve time τ, α is like a coefficient of variation, and we set it as $\alpha = 0.1$. The remaining capacities (u_j^τ) are known, and are readily determined by deducting the assignments up to time τ from the initial capacity. For the safety factor, we set it to be $z = 2$ in the base case. We note that when the remaining capacity of a resource j is zero, the standard deviation will be zero, which leads to infinity penalty coefficient (v_j) according to equation (5.2). To avoid this, the resources with zero remaining capacity should be removed at the QP resolve instance.

For the demand forecasts, we assume that the forecast of the remaining demand is proportional to the original forecast; hence, it is a demand rate multiplied by the remaining time when the demand rate is inferred from the original forecast. To be more specific, for a commodity k with arrival time segment t_k^{start} to t_k^{end}, the updated demand forecast d_k^τ at shadow price update time instance τ is:

$$d_k^\tau = \begin{cases} d_k^0 & \text{when} \quad \tau \le t_k^{\text{start}} \\ \frac{\tau - t_k^{\text{start}}}{t_k^{\text{end}} - t_k^{\text{start}}} d_k^0 & \text{when} \quad t_k^{\text{start}} < \tau < t_k^{\text{end}} \\ 0 & \text{when} \quad \tau \ge t_k^{\text{end}} \end{cases} \tag{5.3}$$

where d_k^0 is the initial demand forecast.

In summary, at every shadow price update instance τ, we update demand forecasts according to equation (5.3), penalty costs (v_j^τ), target flows (f_j^τ) according to (5.2) with fixed safety factor (z) and incremental costs (π_j) and time-dependent standard deviations $(\sigma_j^\tau = 0.1 u_j^\tau)$. Then, we solve the QP formulation (4.6) with these parameters to update the resource shadow prices (λ_j^τ). The fulfillment decisions are based on equation (4.5) with these updated shadow prices.

In our testing process, we can encounter ties in cost in (4.5) when applying equation (4.5) in the QP algorithm. We observed that a good tie-breaking rule is crucial

to an algorithm's success. The tie-breaking rule that we apply throughput the chapter makes intuitive sense: when there is a tie in costs, we first pick the route with resources that expire on an earlier day, i.e., the route with the earliest first resource CPT; if there is still a tie, we prioritize routes by category so that the less flexible resources get utilized first, i.e., we prioritize direct routes (FC-DS) over indirect routes (FC-SC-DS) over third-party routes. In the base case, we update the shadow prices 10 times/day.

5.1.2.6 10 Randomized Shipment Arrivals

We create 10 single-day shipment-arrival data sets from the commodity forecasts, where the number of orders for each FC-DS pair is equal to the one-day forecast for that pair. For each shipment-arrival data set, we first generate the same number of shipments for each commodity, where shipments have the same FC, DS, and promise time as the commodity. Next, we generate the arrival time of each shipment by drawing from a uniform distribution over the arrival time segment of its associated commodity. Finally, the feasible routes of each shipment are identified based on the new arrival time by checking the three conditions listed in section 3.2.1 where we replace "commodity" by "shipment" in the description. We note that these 10 data sets are different in terms of arrival sequences (arrival times) but they all match the demand forecasts perfectly.

5.1.3 A Comparison with the Greedy Algorithm

We make order assignment decisions with our (QP) algorithm and the greedy algorithm (a benchmark) on these 10 data sets of shipment arrival instances. The greedy algorithm makes fulfillment decisions based on route costs, while the QP algorithm makes fulfillment decisions based on the route costs and shadow prices. In addition, when there is a tie in costs, the greedy algorithm breaks the tie based on the same tie-breaking rule as the QP algorithm (described in section 5.1.2.5). We note that the greedy algorithm will always prioritize retailer-controlled routes over third-party

routes since the retailer-controlled routes in the base case have zero cost, while the costs of third-party routes are non-zero (estimated based on distances, see eq (5.1)). Other assumptions we made for the simulations are:

- Every order fulfillment decision needs to be made at the epoch of shipment arrival, i.e., there is no wait time allowed for route decisions.
- After an order fulfillment decision is made, the resources for the route are immediately reserved and are committed to the shipment based on the decision.

We evaluate our algorithm with the average per package cost, which is the total transportation cost (equal to the total third-party route costs since the retailer-controlled routes are free in the base case) over the one-day horizon divided by the number of packages. In all 10 instances, our algorithm outperforms the greedy algorithm. The average cost of our algorithm is 0.779 with a standard deviation of 0.002, the average cost of the greedy algorithm is 0.818 with a standard deviation of 0.001, and the hindsight cost is 0.755. (The hindsight cost is calculated based on full knowledge of demand and capacity, which is described in detail in section 4.5). As we mentioned in the previous section, there are only two types of resources that have limited capacity – upper (FC to SC) and direct (FC to DS) arcs. In all 10 cases, the less-flexible direct resources are 100 percent utilized by both the greedy and our algorithm, due to the tie-breaking rule. Hence, the utilization of limited upper arc resources is the only element that distinguishes the two algorithm's performances.

To understand the utilization of each upper arc in detail, we calculate for each FC the percent of its indirect shipments that go to each DS. We note that all upper arc resources have no remaining capacity with both algorithms. Then, for each (FC, DS) pair, we divide the total number of indirect shipments from the FC to the DS by the total number of indirect shipments shipped from the FC (i.e., the number of shipments from the FC to the SC). Table 5.5 lists the average utilization percentage (u.p.) of the upper arcs (i.e, the shipments from the FC to the SC) by different destination DS over the 10 instances. The cost of shipping through third-party from a given FC to a given DS is also listed in Table 5.5 from cheap to expensive. From the table,

we observe that the QP algorithm allocates a higher percent of the limited upper arc resources to the shipments that are more "expensive", i.e., the DSs that have a higher third-party cost. That is, the QP assigns a higher percentage of its limited capacity for indirect shipments to the demand at DSs with expensive alternatives (third-party costs).

We note that for the upper arc from FC4, the QP algorithm and the greedy algorithm have the same utilization percentages. We observed that the shadow price of this upper arc resource is always less than or equal to the minimum third-party cost (i.e., \$2.15 to DS3) over the one-day horizon for every test case with the QP algorithm. This is because the resource capacity of this upper arc is relatively sufficient considering its demand. Therefore, both algorithms make order fulfillment based on the same tie-breaking rules, which leads to the same fulfillment decision for the indirect shipments from FC4.

FC	DS	QP u.p. (%)	greedy u.p. (%)	3p cost
FC1	DS1	2.46	11.29	2.92
	DS2	37.43	38.58	3.12
	DS3	22.99	19.86	3.43
	DS5	9.01	7.70	3.90
	DS4	28.11	22.57	4.03
FC2	DS1	0.00	8.16	3.78
	DS2	28.18	33.39	4.04
	DS3	26.94	23.05	4.34
	DS5	6.83	5.80	4.82
	DS4	38.06	29.60	4.95
FC3	DS3	11.76	24.10	4.03
	DS2	37.95	36.52	4.27
	DS5	6.66	5.84	4.49
	DS4	32.10	24.67	4.54

	DS1	11.54	8.88	4.62
FC4	DS3	18.41	18.41	2.15
	DS2	47.80	47.80	2.30
	DS5	6.39	6.39	2.60
	DS4	19.70	19.70	2.72
	DS1	7.69	7.69	2.80
FC5	DS3	11.12	23.00	2.06
	DS2	68.12	59.00	2.46
	DS5	5.12	4.50	2.50
	DS4	15.62	13.50	2.61
FC6	DS3	29.48	32.93	2.12
	DS2	32.55	31.79	2.38
	DS5	5.48	5.11	2.68
	DS4	20.90	19.34	2.79
	DS1	11.60	10.84	2.88
FC7	DS1	0.00	11.32	4.81
	DS2	46.35	45.25	5.14
	DS3	21.54	19.07	5.48
	DS5	7.69	5.99	5.93
	DS4	24.42	18.38	6.06
FC8	DS3	8.82	20.20	2.09
	DS2	47.37	43.77	2.47
	DS5	7.27	6.46	2.64
	DS4	25.49	20.71	2.75
	DS1	11.04	8.86	2.97
FC9	DS3	11.83	22.06	2.08
	DS2	49.88	45.37	2.50
	DS5	8.11	7.14	2.50

	DS4	16.46	13.54	2.60
	DS1	13.71	11.88	3.00
FC10	DS4	4.96	16.55	4.87
	DS5	3.36	6.68	4.93
	DS3	24.38	23.77	4.96
	DS2	52.01	41.06	5.40
	DS1	15.28	11.93	5.90
FC11	DS3	4.88	16.72	2.17
	DS5	4.36	6.87	2.55
	DS2	54.31	47.35	2.62
	DS4	27.42	21.66	2.64
	DS1	9.03	7.40	3.12
FC12	DS1	0.00	11.55	5.41
	DS2	41.66	41.95	5.72
	DS3	24.24	20.10	6.02
	DS5	7.55	6.08	6.50
	DS4	26.55	20.32	6.63
FC13	DS4	0.00	7.83	5.76
	DS5	0.55	2.81	5.85
	DS3	60.46	58.04	6.00
	DS2	38.99	31.34	6.44

Table 5.5: Utilization percentage of each upper arc (defined by FC) to different destination (defined by DS) with the QP algorithm and the greedy algorithm. We note that there is no demand between FC5 and DS1; and no demand between FC13 and DS1.

To quantify the "value" of deliveries that an upper arc achieved over the one-day horizon, we introduce the notion of "savings", which is the third-party cost of shipping all orders shipped by the upper arc minus the cost of shipping all orders through retailer-controlled route (which we assumed to be zero in this base case). (We note

that maximizing savings for the online retailer is equivalent to minimizing the revenue of the third party, but their profits may not be diminished, depending on their pricing structure.) Consider a FC to SC arc, and consider all the indirect shipments that are assigned by the QP to this arc; we then define the saving from the assignments to the FC to SC arc to be the sum of what the third-party cost of each of these shipments would have been. For instance, if $S(u, v)$ denotes the set of shipments assigned to the upper arc (u, v), and the third party cost of shipping shipment $s \in S(u, v)$ is denoted by c_s, then the savings of the upper arc (u, v) is $\sum_{s \in S(u,v)} c_s$. Table 5.6 shows the average savings of each upper arc over the 10 instances, where every upper arc (except for the upper arc from FC4) has higher savings with the QP algorithm than the greedy algorithm. The percentage difference (p.d.) (the difference between the QP's and the greedy's savings normalized by the greedy's savings) ranges from 0 to 4.41 percent on each arc.

upper arc FC	FC1	FC2	FC3	FC4	FC5	FC6
greedy	2403	2959	2808	4065	192	2421
QP	2462	3036	2842	4065	196	2429
p.d. (%)	2.45	2.62	1.19	0	2.45	0.3

upper arc FC	FC7	FC8	FC9	FC10	FC11	FC12	FC13
greedy	1956	747	844	1277	2931	2267	3208
QP	1995	769	861	1298	3002	2302	3239
p.d. (%)	1.98	3.02	2.12	1.7	2.43	1.56	0.96

Table 5.6: Average savings of upper arcs with QP and greedy algorithms.

Let's take a closer look at one of the upper arcs to understand how the QP algorithm reserves resources for demand with high third-party costs. By design, the shadow prices of all the resources except the upper and direct arcs are zero, since these resources have more capacity than demand. Therefore, in the QP algorithm, the cost of an indirect route is equal to the shadow price of its associated upper arc. When the shadow price is lower than or equal to the third-party cost, the shipment will be shipped by the indirect route; when the shadow price is higher than the third-party

cost, the shipment will be shipped by the third-party route. Figure 5-1 shows the shadow price of the upper arc that starts from FC8 over the one-day horizon of one instance. We note that there are no direct routes from this FC, and thus shipments from FC8 to any destination have only two route options: indirect retailer-controller route or third-party route.

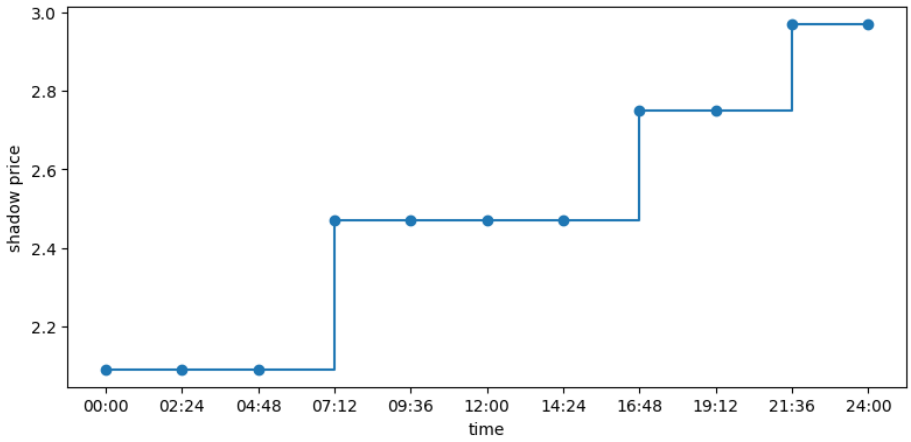


Figure 5-1: Shadow price of the upper arc (FC8, SC1) over the one-day horizon.

The third-party cost to ship from FC8 to different DSs ranges from 2.09 to 2.97 (Table 5.4). Between 0:00 to 7:12 on day 1, the shadow price is 2.09; any demand arrivals in this time period will be shipped by the retailer-controlled route. Between 7:12 to 16:48 on day 1, the shadow price is 2.47; in this time period, shipments to DS3 and DS2 are shipped by third-party, while shipments to other DSs are assigned to the retailer-controlled route. Between 16:48 to 21:36 on day 1, the shadow price is 2.75; in this period, shipments to DS3, DS2, DS5 are shipped by third-party, while shipments to DS1 and DS4 are assigned to the retailer-controlled route. Between 21:36 to the end of day 1, the shadow price is 2.97. In this time period, shipments to DS3, DS2, DS5, DS4 are shipped by third-party, while shipments to DS1 are assigned to the retailer-controlled route. By doing so, the QP algorithm gradually saves or protects the limited resources on the upper arc for the more expensive shipments. Over the one-day horizon, the QP diverts the less expensive demand to third-party by the dynamically updated shadow prices.

In summary, the QP algorithm makes better fulfillment decisions than the greedy

algorithm by allocating limited retailer–controlled resources to serve the more expensive demand, where the "expense" of a demand is its third-party costs. For this base case, we observe that the QP algorithm saves an average of 5% in cost relative to the greedy algorithm. In addition, the greedy algorithm is 0.063 more expensive than hindsight, while QP is 0.024 higher than hindsight. Therefore, the QP achieves most of the potential improvement by shrinking the gap by more than 60%.

5.2 Extension from Base Case – A Realistic Test Case

In this section, we extend the transportation network of the base case to a more realistic transportation network that incorporates other complexities existing in the actual network structure. First, we include another SC so that the network now has more than one SC, and consequently more indirect routes for each commodity. Second, each DS serves multiple service areas; we then define the third-party costs and demand forecasts at the service-area-level instead of at the DS-level. Third, we include mixed routes which, are formed by both retailer-controlled resources and third-party resources. Finally, we impose capacity limits on every resource. In section 5.2.1, we provide a detailed description of the test case. In section 5.2.2, we run the QP algorithm on this test case with different QP parameters, and perform sensitivity analysis on the test case to understand the algorithm's performance at different extremes. In section 5.2.3, we compare the performance of our algorithm with the LP algorithm.

5.2.1 Input Creation

5.2.1.1 Network Structure

In the previous section, we build the base case on the self-contained subnetwork with 13 FCs, 1 SC, 5 DSs. In addition, the network is fully connected. All the FCs are connected to the only SC and the SC is further connected to all the DSs. For the

extension, we add another SC to the network[3], and each of these two SCs is not connected to each FC and DS. This construction increases the number and diversity of routes that are available for each order. We connect each SC to 8 FCs and to 3 DSs, where we chose the connections subject to connectivity requirements: each FC is connected to at least one SC and each DS is connected to at least one SC. Table 5.7 shows the connections between FCs (or DSs) and SCs. The direct arcs available are set the same as in the base case. Table 5.8 shows the direct connections between FCs and DCs. (We label the FCs and DSs the same as we did for the base case 5.1.) We note that in this network structure, some demand (from certain FC to certain DS) has no retailer-controlled route, and therefore, has to go by third party.

		SC	
		SC1	SC2
FC	FC1	√	
	FC2	√	
	FC3	√	
	FC4	√	
	FC5	√	
	FC6	√	√
	FC7	√	√
	FC8	√	√
	FC9		√
	FC10		√
	FC11		√
	FC12		√
	FC13		√
DS	DS1	√	
	DS2	√	
	DS3	√	√
	DS4		√
	DS5		√

Table 5.7: Active indirect arcs (FC to SC or DS to SC)

In addition, each DS serves a set of service areas (ranging from 12 to 79 service areas per DS), where there is a single DS for each service area. We assume that the last-mile delivery between DSs and service areas is accomplished by a contract

[3]The added SC is located in this subnetwork region.

		DS				
		DS1	DS2	DS3	DS4	DS5
FC	FC1	√				
	FC2					
	FC3					
	FC4	√	√	√	√	
	FC5					
	FC6		√		√	√
	FC7					
	FC8					
	FC9				√	
	FC10					
	FC11					
	FC12					
	FC13					

Table 5.8: Active direct arcs (FC to DS)

carrier at some service-area specific cost. The costs will be explained in more detail in section 5.2.1.5.

5.2.1.2 Resources and Capacities

As we mentioned in section 3.1.1, retailer-controlled resources are defined by facilities and timestamps – a node resource is defined by a node facility and a labor shift timestamp, and an arc resource is defined by an origin facility, a destination facility and a CPT. In reality, CPTs on a node or arc are often repeated daily, if not, periodically. For simplicity, we assume that every node resource and every arc resource have a single daily timestamp, and we set a reasonable timestamp for each resource type:

- FC-SC resources daily CPT: 5:00
- SC resources daily CPT: 10:00
- SC-DS resources daily CPT: 15:00
- DS resources daily CPT: 20:00
- FC-DS resources daily CPT: 12:00

We consider these arc resources to have CPTs on two days – day 1, and day 2, with the order promise time being day 2, midnight. Therefore, each arc has two resources in the one-day horizon experiment. For instance, for an arc between FC u and SC v, there are two resources, one with a timestamp (u, v, 5:00 day 1) and the other with timestamp (u, v, 5:00 day 2). We set the capacity of the resources based on the quantity of the in-network-shipment. For each resource type, we create the capacity with different rules:

- FC-SC resources: $70/n\%$ of demand from the associated FC split by day one and day two with 2-8 ratio, where n is the number of SCs connected to the FC. For instance, if demand forecast from a FC u is 200 units, and the FC is connected to both SCs, indexed by v_1 and v_2 then the capacities of the first-day arc resources (u, v_1, 5:00 day 1) and (u, v_2, 5:00 day 1) are both $200 \times 70/2\% \times 20\% = 14$ units; similarly, the capacity of the second-day arc resources (u, v_1, 5:00 day 2) and (u, v_2, 5:00 day 2) are both $200 \times 70/2\% \times 80\% = 56$ units.

- FC-DS resources: 20% of demand from the associated FC and DS split by day one and day two with 2-8 ratio; thus, the day-one capacity is 4% of the demand, and the day-two capacity is 16% of the demand.

- SC-DS resources: $70/n\%$ of demand from the associated DS split by day one and day two with 2-8 ratio, where n is the number of SCs connected to the DS. If n= 1 (2), then the day-one capacity is 14% (7%) of the demand, and the day-two capacity is 56% (28%) of the demand.

- SC resources: 40% of all the demand (shipments in total) split by day one and day two with 2-8 ratio. Thus, for each SC, the day-one capacity is 920 shipments, and the day-two capacity is 3679 shipments.

- DS resources: 90% of demand from the associated DS split by day one and day two with 2-8 ratio; thus, the day-one capacity is 18% of the demand, and the day-two capacity is 72% of the demand.

Table 5.9 summarizes the capacities of FC-SC and FC-DS arc resources for the two days, and table 5.10 summarizes the capacities of SC-DS arc resources. Furthermore, we set no capacity limits on third-party resources (FC (or SC) to service areas) in this extended base case.

	upper arc		direct arc					
FC\SC or DS	SC2	SC1	DS3	DS2	DS1	DS4	DS5	Total
FC1	0	705	0	0	35	0	0	740
FC2	0	670	0	0	0	0	0	670
FC3	0	648	0	0	0	0	0	648
FC4	0	1685	90	235	38	96	0	2144
FC5	0	75	0	0	0	0	0	75
FC6	495	495	0	105	0	64	17	1176
FC7	183	183	0	0	0	0	0	366
FC8	149	149	0	0	0	0	0	298
FC9	336	0	0	0	0	20	0	356
FC10	240	0	0	0	0	0	0	240
FC11	1135	0	0	0	0	0	0	1135
FC12	380	0	0	0	0	0	0	380
FC13	523	0	0	0	0	0	0	523

Table 5.9: Resource capacity of upper (FC-SC) and direct (FC-DS) arcs. We note that the arcs with 0 capacity are not active.

	lower arc				
SC\DS	DS3	DS2	DS1	DS4	DS5
SC1	926	3340	712	0	0
SC2	926	0	0	1681	461

Table 5.10: Resource capacity of lower (SC-DS) arcs. We note that the arcs with 0 capacity are not active.

5.2.1.3 Routes

We consider four types of routes – direct, indirect, **mixed** and third-party route. As described in section 3.1.2, a route is feasible as long as the resources that are required are time compatible. In this extended base case, we create the routes assuming that dwell and transit times are effectively zero(ϵ = 1 second). The estimated delivery

time (EDT) of retailer-controlled routes are the last resource's CPT (DS CPT) plus ϵ. All the routes that we create for the extended base case have an EDT earlier than the promise (day 2, midnight), i.e., all the routes can deliver all the shipments on time. There are 7674 routes in total, 114 of them are retailer-controller routes, 2184 are third-party routes, and 5376 are mixed routes.

5.2.1.4 Commodities and Forecasts

In section 3.2.1, we fintroduced the notion of commodities, which is an aggregate group of shipments for the purpose of demand forecasts. Each commodity is defined by an origin FC, a destination **service area**, a dimension group, an arrival time segment and a promise time. For this extended base case, we assume that there are 13 FCs, 164 service areas, 1 dimension group, 3 arrival time segments (day 0, midnight to day 1, 5AM, day 1, 5AM to noon, and day 1, noon to midnight), 1 promise time (day 2, midnight). We note that these arrival time segments are created based on the first resource CPT of the routes, such that orders that arrive within an arrival time segment have the same set of feasible routes. We note that for some commodities, there will be more feasible routes when the FC and DS is connected to both SCs. In particular, for commodities from FC6, FC7 or FC8 to DS 3, they could have twice the amount of feasible indirect routes than other commodities.

Next, we create the demand forecasts for each commodity. For each (FC, DS) pair, we set its daily demand forecast equal to the shipment data for day 1. In Table D.1 (Appendix D) we provide the daily demand forecasts of each (FC, service area) pair. Then, we distribute the daily forecast of each (FC, service area) pair to its associated commodities proportional to the length of the arrival time segment. For example, suppose a (FC, service area) pair has a daily forecast of 100 units, and we have 3 arrival time segments (day 0, midnight to day 1, 5AM, day 1, 5AM to noon and day 1, noon to midnight). The first commodity with a 5-hour arrival time segment has a forecast of $\frac{5}{24}100$ units, the second commodity with a 7-hour arrival time segment has a forecast of $\frac{7}{24}100$ units, and the third commodity with a 12-hour arrival time segment has a demand forecast of $\frac{12}{24}100$ units. For each commodity, we identify the

set of feasible routes by checking the three conditions listed in section 3.2.1. In this test case, there are 4128 commodities (We note that there are 124 service areas and 13 FCs, which creates 1612 origin-destination (OD) pairs. For each OD pair, there could be 2 or 3 arrival segments depending on whether the OD pair has direct route or not, which adds up to 4128 commodities in total).

5.2.1.5 Costs

In this extended base case, we assume that the cost of retailer-controlled node and arc resources are sunk (paid in advance), and there is a service-area specific last-mile cost. The retailer-controlled route has a flat rate of \$1 per package plus an off-set cost that is service-area specific. This off-set cost is incorporated into the third-party costs. For example, consider a service area that has a last-mile delivery cost of \$1.5 (which is \$0.5 higher from the flat rate, which we term the off-set); then we subtract the off-set of \$0.5 from the third-party cost of this service area. With this specification, in real-time route assignment, each retailer-controlled route has a \$1 base cost while the third-party cost is the net third-party cost, namely the actual third-party cost reduced by the service-area-specific offset of the last-mile delivery. We acknowledge that this is an overly complex set up, but the intent is to demonstrate that the model could accommodate any service-area specific costs of the retailer's route.

The third-party costs are estimated by the distance between the origin facility (either an FC or an SC) and the destination service area with the following linear equation:

$$\text{cost} = \$0.01 \times \text{distance (mile)} + \$2$$

We determine the distance between a facility and a service area with the Haversine formula, which determines the great-circle distance between two points. For each service area, we set the location of a service area to its centroid. In this test case, we assume that we have third party option from all FCs to all service areas, and from all SCs to all service areas (even though some DSs are not connected to the SC, therefore, there might not be an indirect route to a service area through the SC, but

there is a third-party option).

Again, we evaluate the algorithm's performance by the total cost incurred over the one-day horizon. In Table 5.11 we summarize the maximum and minimum values of the third-party costs from an origin FC (or SC) to the destination service areas served by a DS.

origin/destination		DS3	DS2	DS1	DS4	DS5
FC	FC1	3.66/3.29	3.27/2.97	3.03/2.88	4.18/3.84	4.11/3.79
	FC2	4.56/4.19	4.17/3.88	3.91/3.7	5.1/4.76	5.03/4.71
	FC3	4.09/3.9	4.52/4.16	4.77/4.45	4.69/4.39	4.53/4.34
	FC4	2.37/2.12	2.52/2.2	2.96/2.58	2.88/2.53	2.79/2.47
	FC5	2.22/2.04	2.68/2.37	3.12/2.74	2.77/2.41	2.65/2.35
	FC6	2.35/2.01	2.6/2.27	3.05/2.66	2.95/2.59	2.83/2.53
	FC7	5.7/5.33	5.25/4.96	4.97/4.7	6.2/5.88	6.15/5.83
	FC8	2.27/2.02	2.68/2.35	3.13/2.74	2.91/2.55	2.78/2.49
	FC9	2.23/2.03	2.72/2.41	3.17/2.78	2.77/2.41	2.63/2.35
	FC10	5.03/4.75	5.62/5.29	6.07/5.68	4.96/4.71	4.99/4.71
	FC11	2.28/2.02	2.84/2.51	3.28/2.9	2.8/2.46	2.63/2.4
	FC12	6.24/5.87	5.83/5.55	5.56/5.31	6.77/6.44	6.71/6.39
	FC13	6.09/5.77	6.66/6.34	7.1/6.72	5.85/5.6	5.92/5.6
SC	SC1	2.27/2.01	2.67/2.34	3.12/2.74	2.91/2.55	2.77/2.48
	SC2	2.59/2.08	2.46/2.02	2.96/2.37	2.93/2.62	2.9/2.58

Table 5.11: Third party costs of the 2-SC network. (maximum/minimum)

5.2.1.6 Parameters for the QP Algorithm

To implement the QP algorithm, we need to periodically resolve the QP in order to update the shadow prices. At each shadow price update time instance τ, we need to specify the target flows ($\bar{f}_j^\tau$) and penalty costs (v_j^τ) for each resource, and to update demand forecasts (d_k^τ) for each commodity, which are inputs for the QP. As developed in section 4.3.2, we set the target flows ($\bar{f}_j^\tau$) and penalty costs (v_j^τ) by the following formulas:

$$\begin{aligned} \bar{f}_j^\tau &= u_j^\tau - z\sigma_j^\tau \\ v_j^\tau &= \frac{0.798\pi_j}{z^2\sigma_j^\tau} \end{aligned} \tag{5.4}$$

The incremental costs (π_j) captures the per package cost for exceeding capacities on each resource. We estimate the incremental cost for arc resources by:

$$\pi_j = \gamma \times (\$0.01 \times \text{distance} + \$2)$$

where $(0.01 \times \text{distance} + 2)$ is the formula for deriving third-party costs and γ is the discount factor of shipping with retailer-controlled resources versus shipping through third-party. In this extended base case, we set $\gamma = 1$. The incremental costs for node resources are set by a fixed cost $\pi_j = \$0.3$. We set the standard deviation by: $\sigma_j^\tau = \alpha u_j^\tau$, where u_j^τ is the remaining capacity of resource j at resolve time τ, α is like a coefficient of variation, and we will numerically search for a good α in section 5.2.2.1. The remaining capacities (u_j^τ) are known, and are readily determined by deducting the assignments up to time τ from the initial capacity. We note that when the remaining capacity of a resource j is zero, the standard deviation will be zero, which leads to infinity penalty coefficient (v_j^τ) according to equation (5.4). To avoid this, the resources with zero remaining capacity should be removed at the QP resolve instance. For the safety factor, we will numerically search for a good value in section 5.2.2.1.

For the demand forecasts, we assume that the forecast of the remaining demand is proportional to the original forecast; hence, it is a demand rate multiplied by the remaining time, where the demand rate is inferred from the original forecast. To be more specific, for a commodity k with arrival time segment t_k^{start} to t_k^{end}, the updated demand forecast d_k^τ at shadow price update time instance τ is:

$$d_k^\tau = \begin{cases} d_k^0 & \text{when} \quad \tau \le t_k^{\text{start}} \\ \frac{\tau - t_k^{\text{start}}}{t_k^{\text{end}} - t_k^{\text{start}}} d_k^0 & \text{when} \quad t_k^{\text{start}} < \tau < t_k^{\text{end}} \\ 0 & \text{when} \quad \tau \ge t_k^{\text{end}} \end{cases} \tag{5.5}$$

where d_k^0 is the initial demand forecast.

In summary, at every shadow price update instance τ, we update demand forecasts according to equation (5.5), penalty costs (v_j^τ), target flows (f_j^τ) according to (5.4)

with a safety factor (z^*), the incremental costs (π_j), and time-dependent standard deviations ($\sigma_j^\tau = \alpha^* u_j^\tau$). (We will set z^* and α^* based on the numerical tests in section 5.2.2.1.) Then, we solve the QP formulation (4.6) with these parameters to update the resource shadow prices (λ_j^τ). The fulfillment decisions are based on equation (4.5) with these updated shadow prices.

In our test process, we can encounter ties in cost when applying the QP algorithm. We observed that a good tie-breaking rule is crucial to an algorithm's success. The tie-breaking rule that we apply throughput the chapter makes intuitive sense: when there is a tie in costs, we first pick the route with resources that expire on an earlier day, i.e., the route with the earliest first resource CPT; if there is still a tie, we prioritize routes by category so that the less flexible resources get utilized first, i.e., we prioritize direct routes (FC-DS) over indirect routes (FC-SC-DS) over mixed routes, over third-party routes. In this extended base case, we update the shadow prices 10 times/day; we perform a sensitivity test on the shadow price update frequency in section 5.2.2.1.

5.2.2 Sensitivity Tests

5.2.2.1 Tuning of the QP parameters

A user needs to set several parameters to run the QP algorithm. These parameters are: shadow price update frequency, safety factor (z), standard deviation (σ_j^τ), and incremental cost (π_j) of each resource. In this section, we explore a subset of these parameters and leave the rest for future research.

The incremental cost (π_j), intuitively, represents the incremental cost for a shipment when the resource reaches its capacity; for instance it could be the cost of a third-party option, if this option exists. This information is not always available, since only certain transportation arcs have known third-party costs. In Appendix C, we provide two possible ways of estimating incremental costs for retailer-controlled resources in the network. In our experiments, we fix the incremental costs according to the description in section 5.2.1.6, and we do not numerically explore other options. However, our preliminary experiments have shown that the algorithm's performance

is sensitive to the incremental costs; therefore, inferring reasonable incremental costs is an important avenue of future research.

We could estimate the standard deviations (σ_j^τ) from historical data by sampling flow rates over a desired time period, and then derive the standard deviations accordingly. This could be an iterative process since the standard deviations impact the shadow prices which impacts the assignments, and presumably the flow rates on the resources. Therefore, we expect the standard deviations would need to be updated periodically while applying our algorithm. In our experiment, we set the standard deviation (σ_j^τ) to be linear to the remaining capacity of the resource at every shadow price resolve time instance τ, i.e. $\sigma_j^\tau = \alpha u_j^\tau$, where α is like a coefficient of variation, u_j^τ is the remaining capacity at resolve time τ. We will perform sensitivity tests on different values of α.

The safety factor (z) determines the distance between the targets ($\bar{f}_j^\tau$) and the capacities (u_j^τ). The larger the safety factor, the farther the target flows are from the capacities. We will perform sensitivity tests on different magnitudes of safety factors.

Same as the shipment instances that we generated for the base case (section 5.1.2.6), we create 10 single-day shipment-arrival data sets from the commodity forecasts. For each shipment-arrival data set, we first set the number of shipments for each commodity equal to the forecast. Next, we generate the arrival time of each shipment by drawing from a uniform distribution over the arrival time segment of its associated commodity. Finally, the feasible routes of each shipment are identified based on the arrival time by checking the three conditions listed in section 3.2.1 where we replace "commodity" by "shipment" in the description. We note that these 10 shipment-arrival data sets differ in terms of arrival sequences (arrival times) but they all set the demand for each commodity equal to the forecast.

We simulate the 10 test cases with the QP algorithm under different combinations of safety factors (z) and coefficient of variations (α), while fixing the number of shadow price resolves at 10 times per day where the time difference between each resolve is the same (namely 2.4 hours between resolves of the QP). Figure 5-2 shows the average per package cost of the 10 instances under different combinations of z and α. The

average per package cost ranges from 1.9546 to 2.1340. As a benchmark, the cost of the hindsight solution is 1.9139 and the average cost of the greedy solution is 2.0764. (We note that the cost of hindsight is the same for all test cases, since the only difference between the test cases are order arrival sequences within commodities.) The QP algorithm performs better than greedy for a wide range of z and α combinations; however, the QP algorithm could perform worse than the greedy algorithm at some z and α combinations, which shows the importance of the choice of parameter of the QP algorithm.

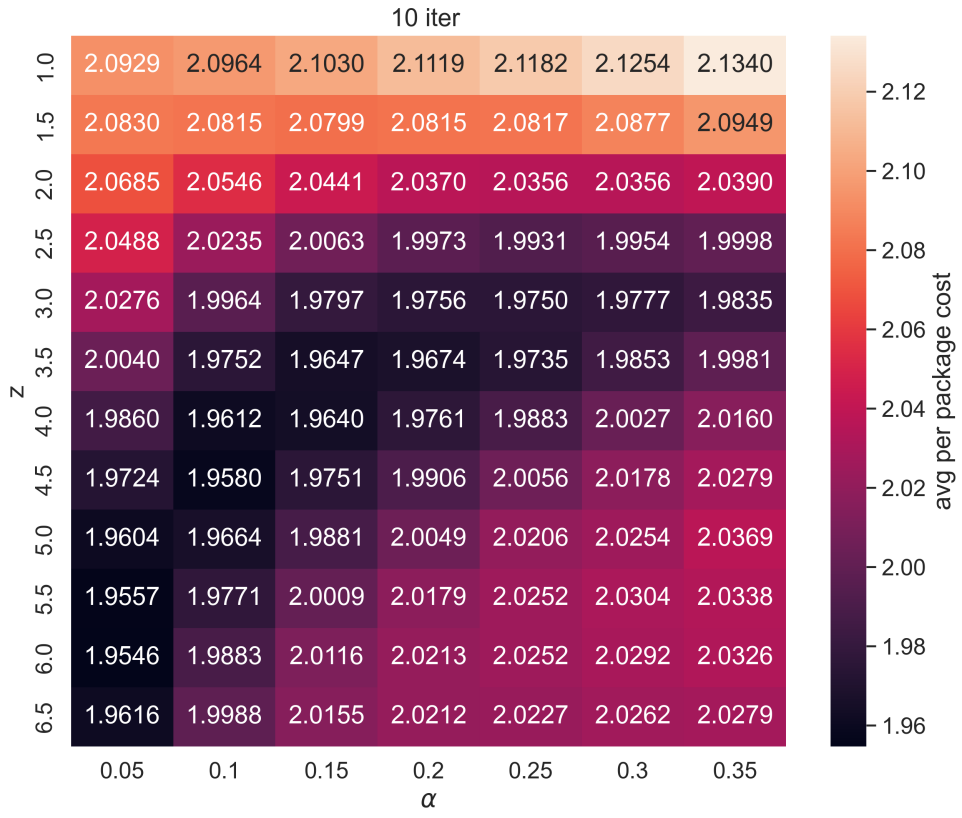


Figure 5-2: Average per package costs at different safety factors (z) and coefficient of variations (α).

Next, we test the QP algorithm with different resolve frequencies with $z = 3$ and $\alpha = 0.25$, which is one of the better-performing combinations for the parameter pair. The blue solid line in figure 5-3 shows the average per package costs of the QP

algorithm, with error bars illustrating the standard deviations. The two benchmarks are the hindsight cost (illustrated in the red dashed line) and the greedy algorithm (illustrated with the green dotted line and the green shadows represent the standard deviations). Regardless of the number of iterations, the QP algorithm performs better than the greedy algorithm, and the QP performance improves as the resolve frequency increases. However, it is surprising, and we do not have an explanation for why costs increase with more frequent resolves for some instances; but the increase is small, and the trend is consistent with our intuition (of increasing resolves improves the algorithm's performance).

In the following sensitivity tests, we set the safety factor by $z = 3$, coefficient of variation by $\alpha = 0.25$, and the resolve frequency is 10 times per day.

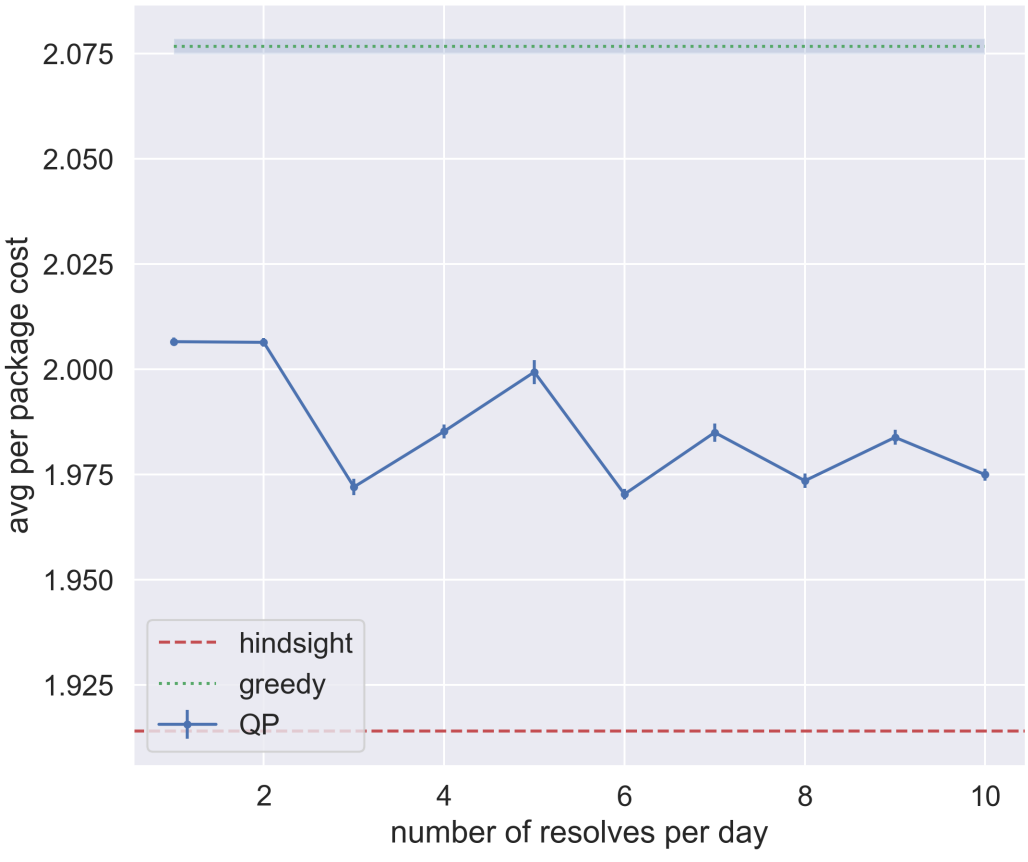


Figure 5-3: Average per package costs at different resolve frequencies

5.2.2.2 Sensitivity to Demand Variability and Forecast Accuracy

In reality, the demand forecasts are not perfect. In this section, instead of the 10 test cases created from a fixed demand forecast, we allow the actual demand to deviate from the forecast and test the order fulfillment algorithms in light of this demand uncertainty. This mimics the reality that the online retailer faces where resource capacities are planned based on demand forecasts, and inevitably the actual demand deviates from the forecasts. In addition, to assess the impact of forecast errors, we tested the QP algorithm with or without perfect forecasts.

To create demand arrivals, we first create the actual quantity of demand at a commodity level. We determine the number of shipments of each commodity k with a random draw from a log-normal distribution $\hat{d}_k \sim \text{Lognormal}(\mu_k, \sigma_k)$, where we set μ_k so that $\mathbb{E}[\hat{d}_k] = d_k$, where d_k is the original demand forecast for the commodity k. We set σ_k so that the standard deviation $\sigma[\hat{d}_k] = \rho d_k$, where ρ is an experimental parameter. The explicit expressions to do this are as follows:

$$\mu_k = \ln(\frac{d_k^2}{\sqrt{d_k^2 + (\rho d_k)^2}}) = \ln(\frac{d_k}{\sqrt{1+\rho^2}}), \tag{5.6}$$

$$\sigma_k = \ln(1 + \frac{(\rho d_k)^2}{d_k^2}) = \ln(1+\rho^2)). \tag{5.7}$$

The parameter ρ controls the magnitude of the standard deviation – the standard deviation of the distribution increases as ρ increases. For a given commodity k, we generate the $\hat{d}_k$ shipment arrival times by drawing from a uniform distribution over the arrival time segment of the commodity.

We run the experiment with 20 different choices of $\rho = 0.1, 0.2, \cdots, 2$. For each ρ, we create 10 sets of demand forecasts ($\hat{d}_k$) and shipment arrival times accordingly. We run two variants for the QP algorithm: for one case, we assume that we know the actual demand for each commodity, namely, we know $\hat{d}_k$. For the other case, we assume that we only know the forecast or expected demand, namely, we know only d_k. Figure 5-4 shows the costs of the QP algorithm with accurate demand forecast (where demand forecast matches the demand $\hat{d}_k$), the QP algorithm without accurate

demand forecast (where demand forecasts are d_k), as well as the greedy algorithm, and hindsight solution at different ρ. We note that the parameters in the QP algorithm are set as section 5.2.1.6 described; the safety factor ($z = 3$), the coefficient of variation ($\alpha = 0.25$) and the resolve frequency (10 times per day) are fixed for all sensitivity tests.

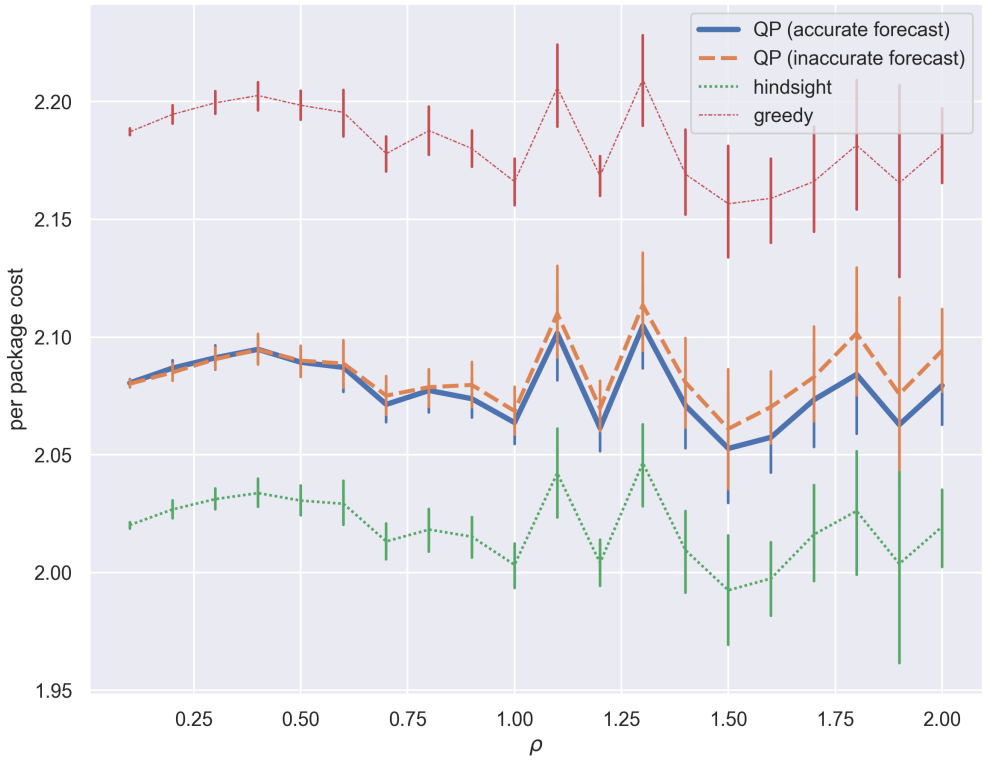


Figure 5-4: Average per package costs at different ρ

The QP algorithm's performance is always better than the greedy algorithm, regardless of demand variability. Figure 5-5 shows the average mean percentage increase from hindsight costs at different ρ. For the case when the QP algorithm knows the demand, we see that its performance is quite insensitive to increasing ρ, which increases the demand variability; however, when the QP algorithm does not know the realized demand, its performance degrades with increasing ρ, which increases the forecast error.

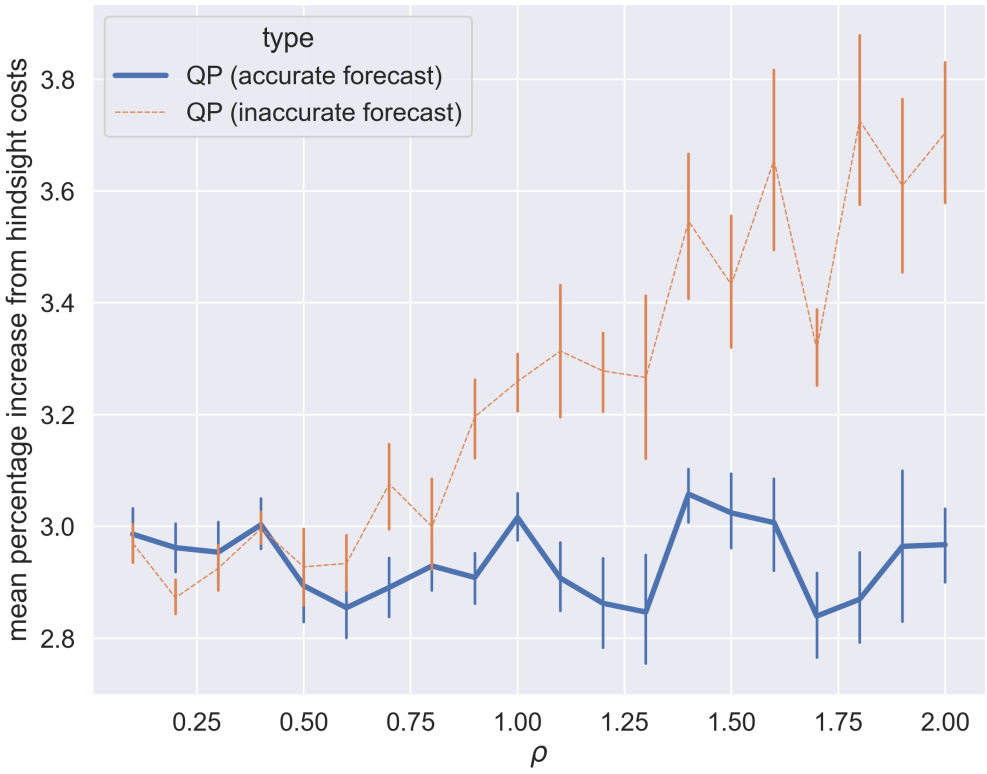


Figure 5-5: Mean percentage increase from hindsight costs at different ρ

5.2.2.3 Sensitivity to Capacity Plans

The retailer will develop a capacity plan to match its resources to its forecast demand. This capacity plan will affect how many and what shipments can be assigned to retailer-controlled resources versus to third-party carriers. For instance, if there is ample capacity then there might be limited need for third party; in contrast, if the capacity is very limited, the retailer will need extensive help from third parties.

We would like for an algorithm to perform consistently under different capacity plans. In this section, we test the algorithms on various capacity plans with a single shipment-arrival data set. We create a range of capacity plans from the resource capacity data in section 5.2.1.2 by modifying the capacity of each resource j as follows:

$$\hat{u}_j = u_j \psi (1 + w_j)^+ \tag{5.8}$$

where $\hat{u}_j$ is the modified capacity of resource j, w_j is a random noise variable for resource j drawn from a normal distribution ($w_j \sim N(0,1)$), u_j is the original capacity in the test case, and ψ is a scaling factor. For instance, by setting $\psi = 0.5$, we are effectively reducing the amount of capacity by 50% compared to the original test case. For each ψ, we make a random draw of the noise (w_j) for each resource j. We create the single shipment-arrival data set from the demand forecast described in section 5.2.1.4, where the number of shipments for each commodity equals the forecast for the commodity. We obtain the arrival time of each shipment by drawing from a uniform distribution over the arrival time segment of its associated commodity. Figure 5-6 shows the mean and standard deviation of the costs of QP, greedy and hindsight. As we decrease ψ, there is less and less retailer-controlled capacity, and hence an increased reliance on third party shipping. Thus, the per package cost increases as we decrease ψ. Furthermore, the difference between the algorithms also decreases with decreasing ψ: with less retailer-controlled resources, there is less opportunity from "smart" resource allocation. Nevertheless, over the full range of capacity plans, the QP algorithm always performs better than greedy in average. In fact, the mean percentage cost increase of the QP algorithm from the hindsight solution is between $0.1\% \sim 1.7\%$, while the mean percentage cost increase of the greedy algorithm from the hindsight solution is between $0.6\% \sim 4.4\%$ (see Figure 5-7). The QP algorithm's gap from the hindsight solution is consistently less than half of the gap for the greedy algorithm over the entire range of ϕ.

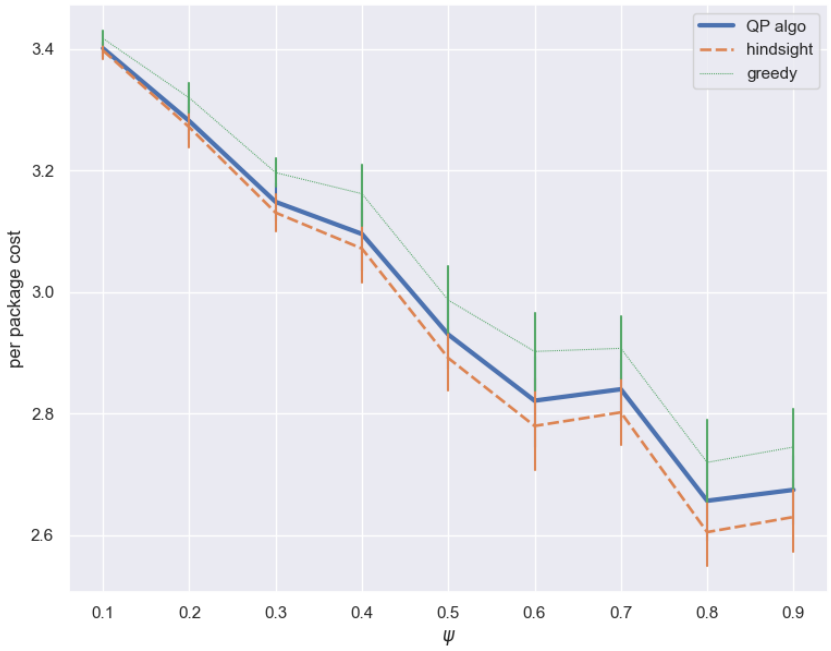


Figure 5-6: Average per package costs at different ψ

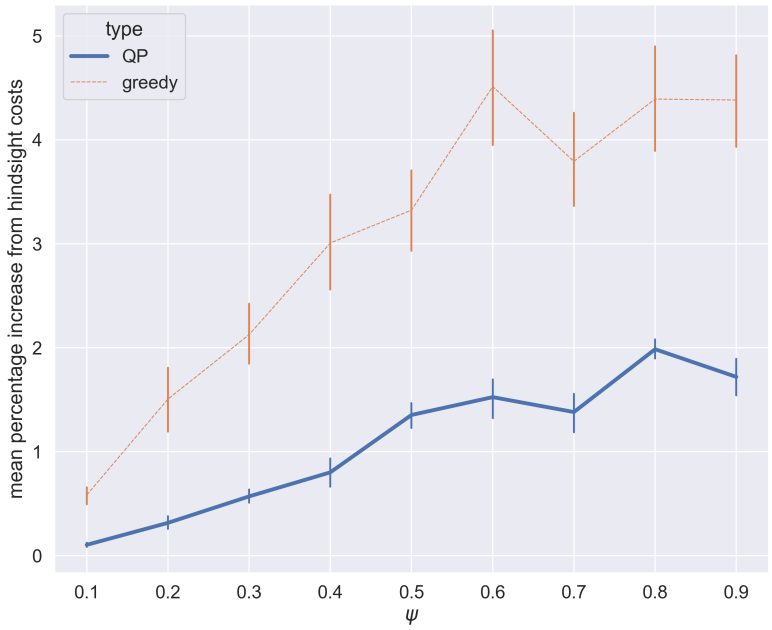


Figure 5-7: Mean percentage increase from hindsight costs at different ψ

5.2.3 QP vs LP algorithms

In this section, we introduce the LP algorithm, which mimics the algorithm currently run by the online retailer. The LP algorithm is nearly identical to the QP algorithm, except for the optimization problem that derives the shadow prices. Instead of equation (4.6), the LP algorithm solves the following linear program for shadow prices:

$$W^{LP}(\mathbf{d}, \mathbf{u}, \tau) = \min \quad \sum_{k \in K^\tau} \sum_{r \in R_k} c_{kr} x_{kr} \tag{5.9a}$$

$$\text{s.t.} \quad \sum_{r \in R_k} x_{kr} = d_k^\tau \quad \forall k \in K^\tau \tag{5.9b}$$

$$\sum_{r \in R_j} \sum_{k \in K_r} x_{kr} \leq u_j^\tau \quad \forall j \in J^\tau \tag{5.9c}$$

$$x_{kr} \geq 0 \quad \forall k, r \tag{5.9d}$$

where variable x_{kr} denotes the number of shipments of commodity k assigned to route r, and other parameters and sets are specified as follows:

Sets

- K^τ: set of commodities for time horizon $[\tau, \tau + T^h]$
- J^τ: set of relevant resources for time horizon $[\tau, \tau + T^h]$
- R_k: set of routes feasible to commodity k
- R_j: set of routes that utilize resource j

Parameters

- d_k^τ: forecast of remaining demand of commodity k at time τ
- u_j^τ: remaining capacity on resource j at time τ
- c_{kr}: per package cost of commodity k shipped through route r

The objective of the LP is to minimize the total transportation costs. Constraint (5.9b) is the demand constraint; constraint (5.9c) is the capacity constraint, from which we use the dual variables as the shadow price of each resource.

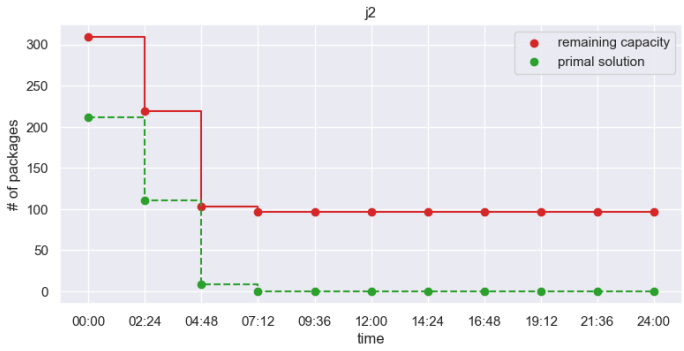
The two algorithms share the same inputs and we compare their performances on 30 test cases. These test cases are randomly generated from the extended base case by modifying the demand and the capacity. For each test case, we first create actual demand ($\hat{d}_k$) at a commodity level from the log-normal distribution based on demand forecast as described in equation (5.6) where we set $\rho = 1$. Next, we create shipment data from the actual demand ($\hat{d}_k$), where the arrival time of each shipment is drawn from a uniform distribution over the arrival time segment of its associated commodity. Finally, we set the capacity plan with equation (5.8) with $\psi = 1$ for each test case. We assume that both the LP and QP algorithm have perfect demand forecasts in this experiment. Table 5.12 shows the average per package cost of the 30 cases. The QP algorithm (where we set $z = 3$ and $\alpha = 0.25$) performs the best among all algorithms with 2.67% mean percentage error from the hindsight solution, which is half of MPE of the LP algorithm, and around 40% of MPE of the greedy algorithm. In addition, among the 30 test cases, the QP algorithm performs better than the LP algorithm in 29 cases and performs better than the greedy algorithm in all cases.

	mean	std	MPE from hindsight (%)
QP	2.31	0.16	2.67
LP	2.37	0.14	5.34
greedy	2.41	0.17	7.10
hindsight	2.25	0.16	0

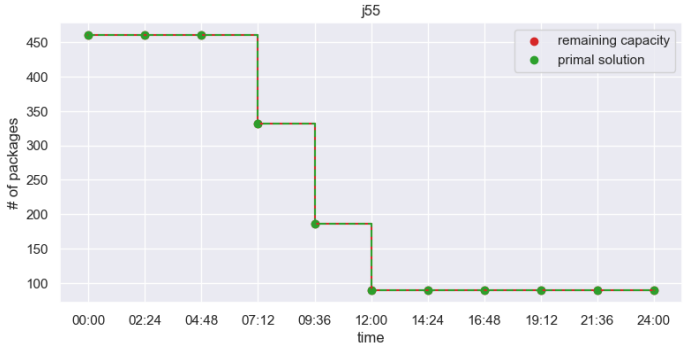
Table 5.12: Mean and standard deviation of per package costs and mean percentage error from hindsight solution of the 30 test cases.

In the 30 test cases, we observe that the primal solutions of the LP algorithm at every shadow price resolve instance are quite different from the QP algorithm's primal solution. In particular, the LP algorithm often leads to unbalance primal solutions where the resource constraints are either "tight" or "very loose" (By "tight", we mean

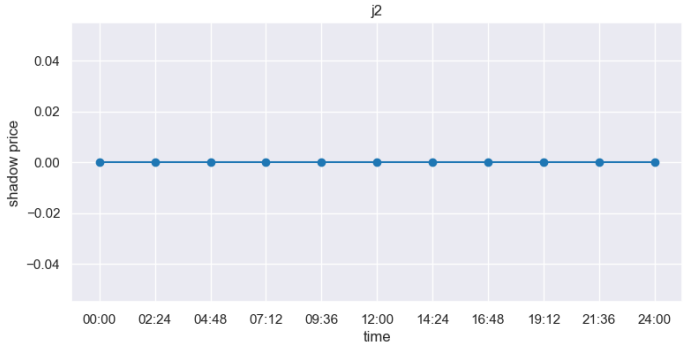
that the capacity constraints are binding, i.e., $\sum_{r \in R_j} \sum_{k \in K_r} x^*_{kr} = u^\tau_j$, where x^*_{kr} is the optimal solution of either the LP or QP problem; by "loose, we mean that the resource constraints are not binding nor close to binding, i.e., $\sum_{r \in R_j} \sum_{k \in K_r} x^*_{kr} \ll u^\tau_j$.); while the QP algorithm often leads to a more balanced primal solution where flows are more evenly distributed among resources. To illustrate this observation, we zoom in on a test case and observe the progression of remaining capacity, the primal solution, and the shadow prices of an upper arc (FC1 to SC1) over time. Figure 5-8 shows the remaining capacities, optimal flows, and shadow prices of the resources associated with the upper arc when the LP algorithm is applied to the test case, while figure 5-9 shows this information when the QP algorithm is applied. There are two relevant resources on the chosen upper arc: j_2 is the first resource (with an earlier CPT at 5:00 day 1) on the upper arc and j_{55} is the second resource (with a later CPT at day 2, 5:00) on the upper arc. With the LP algorithm, the optimal flow of resource j_{55} is always equal to the capacity, while the optimal flow of resource j_2 is always below the capacity, which leads to zero shadow price for j_2. On the other hand, with the QP algorithm, the optimal flow on resource j_2 and j_{55} are close to but not equal to the capacity, and the shadow prices are non-zero since there is a penalty when the primal solution exceeds the target flow. The shadow prices produced by the QP algorithm lead to a better utilization of the upper arc than the LP algorithm; for instance, the savings (as defined in section 5.1.3) of this upper arc resource with the QP algorithm is 7% higher than the LP algorithm.



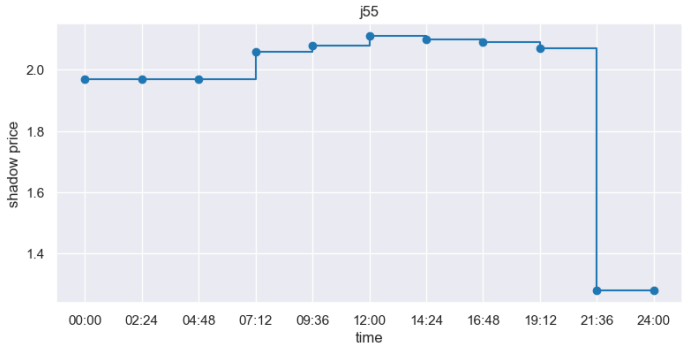
(a) Primal solutions and remaining capacities on resource j_2 over the one-day horizon.



(b) Primal solutions and remaining capacities on resource j_{55} over the one-day horizon.

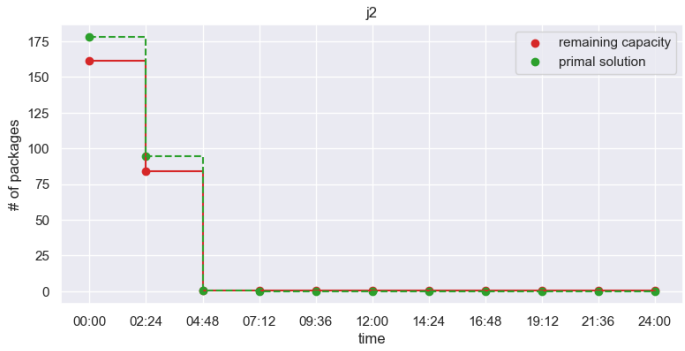


(c) Shadow price of resource j_2 over the one-day horizon.

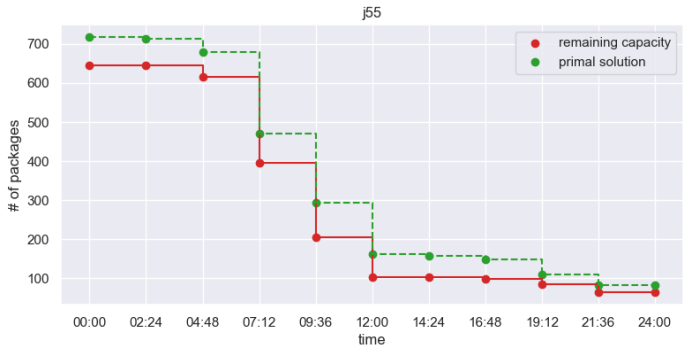


(d) Shadow price of resource j_{55} over the one-day horizon.

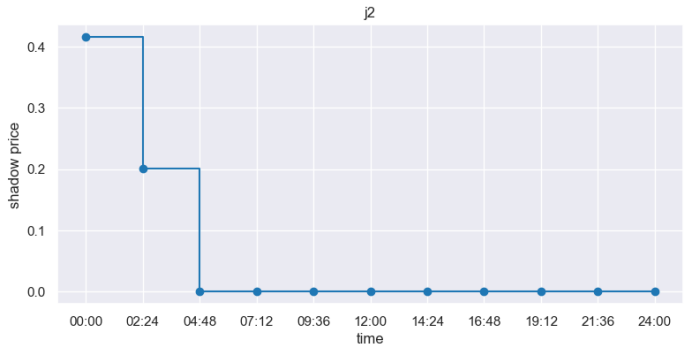
Figure 5-8: Shadow price, optimal flows, remaining capacities on the resources associated with an upper arc over the one-day horizon when applying the LP algorithm.



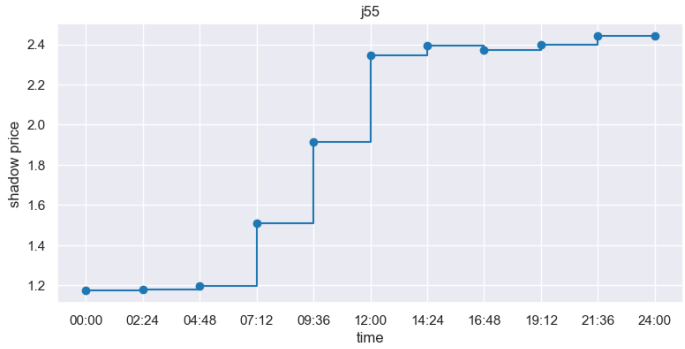
(a) Primal solutions and remaining capacities on resource j_2 over the one-day horizon.



(b) Primal solutions and remaining capacities on resource j_{55} over the one-day horizon.



(c) Shadow price of resource j_2 over the one-day horizon.



(d) Shadow price of resource j_{55} over the one-day horizon.

Figure 5-9: Shadow price, optimal flows, remaining capacities on the resources associated with an upper arc over the one-day horizon when applying the QP algorithm.

One might argue that the QP algorithm's success is due to the target flows that provide a buffer from the capacity, while the LP algorithm does not have this buffer. What if the remaining capacity (u_j^τ) in the LP formulation is replaced by a discounted capacity (ηu_j^τ), where η is a discount factor? Will the LP algorithm perform better than the QP algorithm? To answer this question, we simulate the LP algorithm on the same 30 test cases with different values for the buffer (η). Figure 5-10 shows the LP's performance with different η, where we observe that the LP algorithm does not benefit from having buffers in capacity.

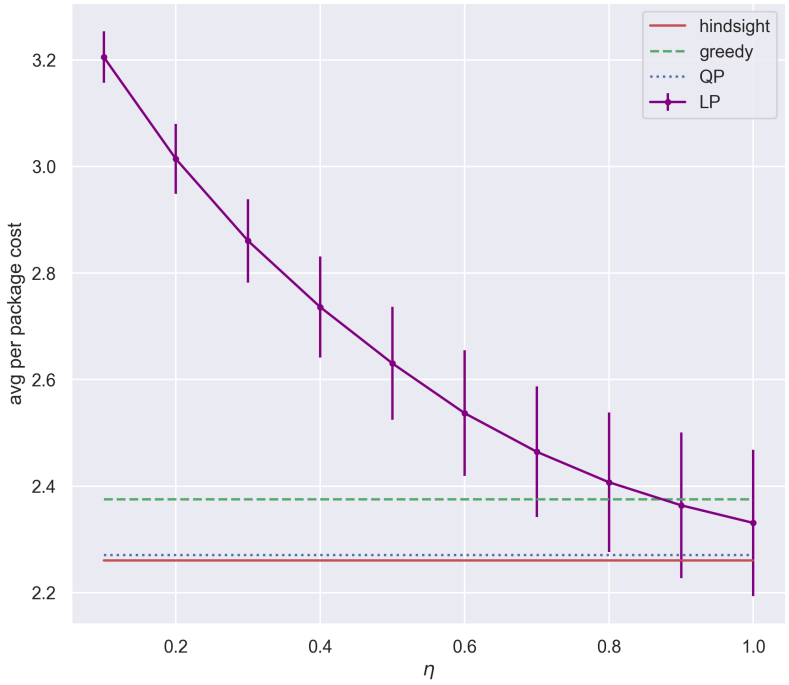


Figure 5-10: LP algorithm's performance with different buffer

5.2.4 Summary

The experiments in this chapter suggest that the QP algorithm outperforms the benchmark algorithms in this set of test problems. The QP parameters are the key to success the algorithm's success, and the algorithm is robust to different scenarios (varied demand, varied capacity, and inaccurate forecasts) when applied with the

right set of parameters. The QP algorithm provides early and gradual alerts when we expect the capacity of a resource to be tight. In addition, the algorithm seems better at modulating the shadow prices so as to allocate better scarce capacity on resources, which leads to a more flexible flow assignment.

Chapter 6

Concluding Remarks

The order fulfillment problem is highly time-sensitive in the eCommerce setting. The transportation assignment of the fulfillment problem, which our research considers, contains time elements in every corner. In particular: Orders need to be delivered by a certain deadline and can only be processed (observed) after the orders are placed in the system, therefore is time sensitive. Transportation resources are only available for a certain time period. Routes should be formed with resources in a time-harmonious way. Therefore, the formulation of the problem needs extra care, in that an ideal model should capture the time elements in all aspects. In the meantime, the model cannot be overly complex (detailed) since the fulfillment decision needs to be made very frequently given the high volume of incoming orders and short fulfillment windows. In our research, we realized that aggregation and simplification are crucial for an implementable model in practice. The formulation we proposed in chapter 3 serves this purpose, where the model remains in its simplest form, and leaves the complexity in the pre-processing of the input parameters of the model (e.g., route formation from resources, feasible route identification for orders... etc).

At the start of the project, our industry collaborator already has an algorithm up and running for the transportation assignments, and our goal is to improve upon it. As a result, the QP algorithm we proposed, by design, aims to address the undesired features of the existing algorithms. In chapter 4, we share the details of the design logic of the QP algorithm. Finally, the QP algorithm is tested in the realistic

subnetwork, and the tests are performed with semi-realistic data. The QP algorithm demonstrates its potential for cost-saving while remaining as implementable as the existing algorithm.

We conclude by noting several interesting future directions to explore. First, a natural question is whether we could provide theoretical guarantees of our algorithm in comparison with other benchmarks. Second, in our project, we formulate the transportation assignment of the fulfillment problem as a standalone problem. The upstream decisions (transportation capacity planning) and concurrent decisions (which fulfillment center to fulfill each order), if considered jointly, could potentially provide additional improvements. In addition, other problems like pricing and assortment display are also intertwined with the order fulfillment decisions and could be studied jointly. Third, the benefit of delaying fulfillment decisions is not investigated in our work. In fact, orders come with different delivery promises (ex: end of the next day or end a week from now) based on preferences of the customer. Orders with delayed delivery have more chance for reassignment, which provide a chance of improvement in costs.

Online retailing will continue to prosper in the foreseeable future, which accelerates the development of cost and time-efficient order fulfillment systems. Our industry collaborator has provided continuing improvement in the delivery experiences for their customers, thanks to their early and heavy investment in their transportation system. In an ideal world, the whole eCommerce space (including the transportation companies and online retailers) would operate jointly and harmoniously to achieve an efficient fulfillment process that has the smallest impact on the environment. Our research is only a small slice of this big picture, and we look forward to seeing more diverse systems and innovative algorithms that propel this goal.

Appendix A

The Structure of the Dual Variables from the QP

The KKT condition helps us understand the structure of the resource shadow prices derived from the QP (4.6). Here, we repeat the QP formulation (at resolve time τ) again:

$$W^{QP}(\mathbf{d}, \mathbf{u}, \tau) = \min \quad \frac{1}{2} \sum_{j \in J^\tau} v_j^\tau g_j^2 + \sum_{k \in K^\tau} \sum_{r \in R_k} c_{kr} x_{kr} \tag{A.1a}$$

$$\text{s.t.} \quad \sum_{r \in R_k} x_{kr} = d_k^\tau \quad \forall k \in K^\tau \tag{A.1b}$$

$$\sum_{r \in R_j} \sum_{k \in K_r} x_{kr} - g_j \leq \bar{f}_j^\tau \quad \forall j \in J^\tau \tag{A.1c}$$

$$x_{kr}, g_j \geq 0 \quad \forall k, r, j \tag{A.1d}$$

We denote the dual variables of the demand constraints (A.1b) by η_i^τ; denote the dual variables of the resource constraints (A.1c) by λ_j^τ; denote the shadow prices associated with the non-zero constraints of variable x_{kr} by ρ_{kr} and variable g_j by ρ_j.

By the KKT stationarity conditions, we have:

$$v_j g_j - \lambda_j + \rho_j = 0 \tag{A.2}$$

When $\rho_j = 0$, $\lambda_j = v_j g_j$, which means that the dual variables are linear to the excess flows. This cost structure is very different from the one in LP. We note that by complementary slackness, when $g_j > 0$, $\rho_j = 0$, which means that the linearity of dual variable holds when there are positive excess flows on the resource.

Appendix B

Identify Relevant Resources and Routes

The process of identifying relevant resources and routes for the capacity planning problem is non-trivial. In this section, we provide a detailed description of the process for a capacity planning problem with a fixed time horizon between $t = t^{\text{start}}$ and $t = t^{\text{end}}$.

B.0.1 Identify Relevant Resources

Relevant resources are the resources with timestamps that are available to orders arrive between $t = t^{\text{start}}$ and $t = t^{\text{end}}$. As we mentioned earlier, orders are assumed to travel in the network fluidly according to the average dwell and transit times at the nodes and arcs. With this assumption, the arrival time of the earliest order (which arrives at $t = t^{\text{start}}$) and the latest order (which arrives at $t = t^{\text{end}}$) on each resource can be estimated, and if the lifetime of the resource overlaps this time interval, the resource is considered relevant. (Note that the lifetime of the resource is the time from the resource's previous CPT to its associated CPT. For example, suppose that a facility has daily CPT at 11AM and 5PM, the lifetime of the node resource with 11AM CPT is from 5PM the day before to 11AM today; the lifetime of the node resource with 5PM CPT is from 11AM today to 5PM today.).

- For lower arc (SC v to DS w), the relevant arc resource's CPT (t) satisfies:

$$[t^{\text{start}} + \min_{u \in U}\{\gamma_u + \gamma_{uv}\} + \gamma_v, t^{\text{end}} + \max_{u \in U}\{\gamma_u + \gamma_{uv}\} + \gamma_v] \cap [f_{vw}^{-1}(t), t] \neq \emptyset$$

  where U denotes the set of FCs, γ_{uv} denotes the transit time on arc from FC u to SC v, γ_v denotes the dwell time at SC v.

- For upper arc (FC u to DS w), the relevant arc resource's CPT (t) satisfies:

$$[t^{\text{start}} + \gamma_u, t^{\text{end}} + \gamma_u] \cap [f_{uw}^{-1}(t), t] \neq \emptyset$$

- For node resources at SC (v), the relevant node resource's CPT (t) satisfies:

$$[t^{\text{start}} + \min_{u \in U}\{\gamma_u + \gamma_{uv}\}, t^{\text{end}} + \max_{u \in U}\{\gamma_u + \gamma_{uv}\}] \cap [f_{vw}^{-1}(t), t] \neq \emptyset$$

- For node resources at DS (w), the resources we consider has CPTs (t) that satisfy:

$$[t^{\text{start}} + \min_{u \in U, v \in V}\{\gamma_u + \gamma_{uv} + \gamma_v + \gamma_{vw}\}, t^{\text{end}} + \max_{u \in U, v \in V}\{\gamma_u + \gamma_{uv} + \gamma_v + \gamma_{vw}\}] \cap [f_{vw}^{-1}(t), t] \neq \emptyset$$

  where V denote the set of SCs, γ_{vw} denotes the transit time on arc from SC v to DS w, γ_w denotes the dwell time at DS w.

We denote the resources that satisfy the above criteria by $J(t^{\text{start}}, t^{\text{end}})$, which is the set of resources relevant to orders that arrive between t^{start} and t^{end}.

B.0.2 Identify Relevant Routes

In this section, we compose routes from relevant resources ($J(t^{\text{start}}, t^{\text{end}})$). As we mentioned in section 3.1.2, there are four route types, and every route type has its own pattern. To form different types of routes, we start with the first resource that was identified as "relevant" by the previous step, and identify consecutive resources

according to the following formula:

$$\tilde{t}^{down} \leq t^{up} + \gamma^{up} < t^{down} \tag{B.1}$$

where t^{up} (t^{down}) is the CPT of the upstream (downstream) resource, $\tilde{t}^{down}$ is the previous CPT of t^{down} of the same arc or node resource, γ^{up} is the time it takes to travel from the upstream resource to the downstream resource. Specifically, if the upstream resource is a node, then γ^{up} is the dwell time at the node facility; if the downstream resource is an arc, then γ^{up} is the transit time on the arc. In addition to time coherence, the formation of routes follow fixed patterns. The four different route patterns are:

- Direct route (FC u to DS w) is composed of a FC-DS resource and a DS resource, i.e. $r(j(u, w, t_{uw}), j(w, t_w))$. Based on formula (B.1), t_w can be uniquely identified, where t_w satisfies:

$$f_w(t_w)^{-1} \leq t_{uw} + \gamma_{uw} < t_w$$

- Indirect route (FC u to SC v to DS w) is composed of a FC-SC resource, a SC resource, a SC-DS resource and a DS resource, i.e. $r(j(u, v, t_{uv}), j(v, t_v), j(v, w, t_{vw}), j(w, t_w))$. Based on formula (B.1), t_v, t_{vw}, t_w can be identified uniquely sequentially by satisfying the following equations:

$$f_v^{-1}(t_v) \leq t_{uv} + \gamma_{uv} < t_v$$
$$f_{vw}^{-1}(t_{vw}) \leq t_v + \gamma_{vw} < t_{vw}$$
$$f_w^{-1}(t_w) \leq t_{vw} + \gamma_{vw} < t_w$$

- Third party route (with pickup point at FC u) is composed of one FC third party resource, i.e. $r(j(u, t))$. We create routes for every relevant FC third party resources.

- Mixed route (starts from FC u, with pickup point at SC v) is composed of one

FC-SC resource and one SC third party resource, i.e. $r(j(u, v, t_{uv}), j(v, t_v))$. Based on formula (B.1), t_v can be uniquely identified, where t_v satisfies:

$$f_v(t_v)^{-1} \leq t_{uv} + \gamma_{uw} < t_v$$

We denote the routes composed in this section by $R(t^{\text{start}}, t^{\text{end}})$, which is the set of relevant routes to orders that arrive between t^{start} and t^{end}.

Appendix C

Estimation of Incremental Costs

Incremental costs (c_j) are estimated per package cost for flows that exceed the capacities. It is used in the derivation of penalty costs (v_j) in the QP formulation (section 4.3.2). It also represent an estimated "alternative cost", which is the cost for the network to process an extra unit on the resource. In reality, such cost is not readily available for every resource. For some third-party resources that charge extra packages by package, the incremental costs are readily available, but for other resources, especially the retailer-controlled resources, there is no per package costs for flows that exceed capacities.

In this section, we assume that the incremental costs of a subset of resources ($\tilde{J}$) are unknown, while the remaining resources ($J/\tilde{J}$) have known incremental costs. Our goal is to develop an approximation or reasonable estimate for the unknown incremental costs. We provide two different methods that achieve this goal.

C.1 Method 1 - The Iterative Method

The iterative method solves a transportation LP (C.1) with gradually inflated demand. The intent of this heuristic is to capture the cost for each resource right after capacities are exceeded. The inputs of the LP include initial demand forecasts (d_k), capacities (u_j) and route base costs (c_{kr}). The LP has demand constraints (C.1b), resource constraints for the resources with unknown incremental costs (C.1c). The

first term of the objective function is the sum of total route base costs, and the second term of the objective function penalize the resources with known costs. As demand is gradually inflated by the parameter α, a subset of the resource constraints of the LP becomes tight, and by complementary slackness, these resource constraints have non-zero shadow prices, which we can use as their incremental costs. The heuristic goes through the following steps until termination:

1. Set $\alpha = 1$,
2. Solve the LP (C.1) with updated α ($W(\alpha)$) for dual variables associated with the resource constraints.
3. For any resource $j \in \tilde{J}$ that has non-zero duals in the solution of $W(\alpha)$, set c_j equal to the duals from the solution of $W(\alpha)$.
4. Remove j from $\tilde{J}$.
5. If $\tilde{J}$ is not empty, set $\alpha = \alpha(1 + \min_j \delta_j)(1 + \epsilon)$, where δ_j are the relative slacks (slack divided by capacity) in resource constraints and ϵ is some small constant. Return to step 2. If $\tilde{J}$ is an empty set, the heuristic terminates.

$$W(\alpha) = \min \quad \sum_{k \in K} \sum_{r \in R_k} c_{kr} x_{kr} + \sum_{j \in J/\tilde{J}} c_j (\sum_{r \in R_j} \sum_{k \in K_r} x_{kr} - u_j) \tag{C.1a}$$

$$\text{s.t.} \quad \sum_{r \in R_k} x_{kr} = \alpha d_k \quad \forall k \in K \tag{C.1b}$$

$$\sum_{r \in R_j} \sum_{k \in K_r} x_{kr} \leq u_j \quad \forall j \in \tilde{J} \tag{C.1c}$$

$$x_{kr} \geq 0 \quad \forall k, r \tag{C.1d}$$

We note that there are several details of this heuristic that can be done differently. First, the parameter α can be updated differently. The intent of the formula in step 5 is to infer a reasonable step size that guarantees the inflation would lead to at least one binding resource constraint in the next iteration. Second, instead of having

a universal α, the parameter can be commodity-dependent (α_k). The commodity-dependent α_k allows the demand to be inflated in different directions. We note that the convergence of this iterative process is not guaranteed and need to be tested numerically for practical purposes.

C.2 Method 2 - The Linear Approximation

The linear approximation is a straightforward approach that approximates the incremental costs for the retailer-controlled arc resources from third party costs. The method assumes that each arc is associated with a "weight" and approximates the incremental costs with these weights. In addition, it assumes that we have the third party cost of every FC (u) and DS (w) pair, i.e., c^{3p}_{uw} is given. There are three types of retailer-controlled arcs: upper arcs, lower arcs and direct arcs. For an upper arc between FC u to SC v, the incremental cost is approximated by the following formula:

$$c_{j(u,v)} = \sum_{w \in W(v)} \frac{l_{uv}}{l_{uv} + l_{vw}} c^{3\mathrm{p}}_{uw},$$

where $W(v)$ is the set of DSs connected to SC v, $c^{3\mathrm{p}}_{uw}$ is the third party cost from FC u to DS w, l_{uv} and l_{vw} are weights associated with upper arc (u, v) and lower arc (v, w), respectively. For lower arc between SC v to DS w, the incremental cost is approximated by the following formula:

$$c_{j(v,w)} = \sum_{u \in U(v)} \frac{l_{vw}}{l_{uv} + l_{vw}} c^{3\mathrm{p}}_{uw},$$

where $U(v)$ is the set of FCs connected to SC v, l_u is the weight associated with FC u. For direct arc between FC u to DS w, the incremental cost is set by the third party cost:

$$c_{j(u,w)} = c^{3\mathrm{p}}_{uw}.$$

The weighing of arcs (l_{uv}, l_{vw}) could be set based on the distance between the origin and destination, or the adhoc truck cost, or other sensible weights. By setting the incremental cost with this linear approximation, the incremental costs are compatible to the third party costs in magnitude, in that the cost of shipping a package from a given origin and destination with third party would be similar to the cost of all the feasible routes where the costs are the sum of incremental costs of the resources along the routes.

We note that the linear approximation can be extended to the case where there are not only arc resources, but node resources that have unknown incremental costs. By the same token, the linear approximation would require estimated “weights" for node resources (l_u, l_v, l_w), which could be estimated from the size of the facility or the cost of processing a package. And for each resource type, a sensible linear approximation need to be created similarly as the above equations such that the incremental costs are compatible to the third party costs in magnitude.

Appendix D

Commodity Forecast of the Extended Base Case

service area\FC	FC1	FC2	FC3	FC4	FC5	FC6	FC7	FC8	FC9	FC10	FC11	FC12	FC13	Total
SA1	6.0	4.0	7.0	14.0	1.0	9.0	5.0	2.0	3.0	3.0	13.0	4.0	4.0	75.0
SA2	4.0	4.0	3.0	8.0	2.0	6.0	3.0	1.0	2.0	1.0	7.0	2.0	2.0	45.0
SA3	3.0	2.0	2.0	6.0	1.0	4.0	2.0	1.0	1.0	1.0	7.0	2.0	2.0	34.0
SA4	5.0	6.0	4.0	9.0	1.0	9.0	4.0	2.0	2.0	2.0	11.0	2.0	3.0	60.0
SA5	5.0	5.0	5.0	13.0	1.0	13.0	4.0	4.0	4.0	2.0	13.0	4.0	4.0	77.0
SA6	12.0	12.0	10.0	26.0	3.0	20.0	9.0	6.0	6.0	5.0	33.0	8.0	8.0	158.0
SA7	11.0	12.0	10.0	20.0	2.0	20.0	7.0	5.0	5.0	6.0	26.0	7.0	7.0	138.0
SA8	6.0	9.0	8.0	20.0	2.0	12.0	5.0	5.0	5.0	4.0	21.0	5.0	5.0	107.0
SA9	4.0	3.0	5.0	11.0	1.0	7.0	3.0	4.0	3.0	3.0	10.0	4.0	3.0	61.0
SA10	5.0	5.0	4.0	10.0	1.0	8.0	3.0	2.0	2.0	2.0	11.0	3.0	3.0	59.0
SA11	7.0	7.0	6.0	12.0	0.0	9.0	5.0	3.0	4.0	3.0	14.0	4.0	4.0	78.0
SA12	3.0	5.0	4.0	8.0	1.0	7.0	3.0	3.0	3.0	2.0	8.0	4.0	4.0	55.0
SA13	6.0	7.0	7.0	13.0	2.0	11.0	5.0	3.0	3.0	2.0	17.0	4.0	4.0	84.0
SA14	6.0	5.0	5.0	12.0	1.0	8.0	4.0	4.0	4.0	3.0	14.0	4.0	3.0	73.0
SA15	9.0	9.0	11.0	20.0	3.0	16.0	8.0	5.0	6.0	4.0	23.0	6.0	6.0	126.0
SA16	4.0	4.0	4.0	9.0	0.0	6.0	4.0	2.0	2.0	1.0	8.0	2.0	3.0	49.0
SA17	5.0	5.0	5.0	11.0	1.0	6.0	3.0	3.0	4.0	2.0	9.0	4.0	3.0	61.0
SA18	8.0	8.0	7.0	16.0	2.0	13.0	6.0	7.0	6.0	4.0	20.0	6.0	6.0	109.0

SA19	10.0	5.0	5.0	12.0	0.0	9.0	4.0	1.0	3.0	3.0	8.0	5.0	0.0	65.0
SA20	4.0	3.0	3.0	6.0	0.0	4.0	2.0	1.0	2.0	1.0	6.0	2.0	0.0	34.0
SA21	6.0	6.0	6.0	11.0	1.0	10.0	4.0	2.0	3.0	2.0	13.0	4.0	5.0	73.0
SA22	9.0	7.0	7.0	16.0	2.0	12.0	4.0	4.0	4.0	3.0	14.0	5.0	5.0	92.0
SA23	7.0	8.0	8.0	18.0	0.0	11.0	6.0	4.0	5.0	4.0	18.0	6.0	4.0	99.0
SA24	8.0	6.0	6.0	37.0	2.0	8.0	4.0	3.0	4.0	2.0	13.0	5.0	4.0	102.0
SA25	8.0	4.0	5.0	34.0	1.0	8.0	3.0	3.0	3.0	2.0	14.0	4.0	4.0	93.0
SA26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
SA27	4.0	3.0	4.0	20.0	0.0	4.0	3.0	3.0	2.0	1.0	8.0	2.0	2.0	56.0
SA28	5.0	5.0	8.0	12.0	1.0	8.0	3.0	2.0	3.0	3.0	13.0	4.0	4.0	71.0
SA29	11.0	6.0	9.0	44.0	3.0	14.0	6.0	4.0	4.0	4.0	19.0	6.0	6.0	136.0
SA30	2.0	2.0	2.0	4.0	0.0	4.0	1.0	1.0	1.0	0.0	4.0	2.0	1.0	24.0
SA31	4.0	2.0	3.0	17.0	1.0	4.0	2.0	1.0	1.0	1.0	8.0	2.0	2.0	48.0
SA32	6.0	6.0	8.0	14.0	1.0	9.0	5.0	4.0	4.0	3.0	16.0	4.0	6.0	86.0
SA33	5.0	7.0	6.0	12.0	1.0	6.0	3.0	2.0	3.0	2.0	12.0	3.0	5.0	67.0
SA34	4.0	4.0	6.0	10.0	1.0	7.0	3.0	3.0	4.0	2.0	9.0	3.0	4.0	60.0
SA35	6.0	6.0	6.0	39.0	2.0	13.0	5.0	3.0	3.0	4.0	14.0	4.0	4.0	109.0
SA36	5.0	5.0	4.0	10.0	1.0	7.0	3.0	2.0	2.0	1.0	10.0	3.0	3.0	56.0
SA37	6.0	5.0	6.0	32.0	2.0	7.0	3.0	4.0	4.0	2.0	14.0	4.0	4.0	93.0
SA38	8.0	7.0	9.0	42.0	2.0	12.0	6.0	4.0	4.0	3.0	20.0	6.0	5.0	128.0

SA39	4.0	2.0	3.0	18.0	1.0	4.0	2.0	2.0	2.0	1.0	6.0	2.0	2.0	49.0
SA40	5.0	5.0	6.0	10.0	1.0	4.0	4.0	3.0	4.0	2.0	13.0	4.0	3.0	64.0
SA41	4.0	4.0	3.0	8.0	1.0	6.0	3.0	2.0	2.0	1.0	9.0	2.0	2.0	47.0
SA42	3.0	3.0	2.0	6.0	1.0	4.0	2.0	2.0	2.0	2.0	6.0	3.0	2.0	38.0
SA43	4.0	5.0	5.0	12.0	1.0	9.0	3.0	3.0	4.0	2.0	14.0	4.0	4.0	70.0
SA44	8.0	6.0	8.0	13.0	2.0	11.0	5.0	3.0	4.0	3.0	15.0	5.0	6.0	89.0
SA45	2.0	2.0	3.0	4.0	1.0	4.0	2.0	2.0	2.0	1.0	5.0	2.0	2.0	32.0
SA46	4.0	5.0	4.0	10.0	1.0	7.0	2.0	2.0	2.0	2.0	9.0	3.0	2.0	53.0
SA47	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	2.0
SA48	17.0	7.0	5.0	15.0	0.0	10.0	4.0	5.0	4.0	4.0	12.0	6.0	0.0	89.0
SA49	18.0	8.0	9.0	18.0	0.0	12.0	5.0	4.0	6.0	4.0	11.0	4.0	0.0	99.0
SA50	12.0	5.0	6.0	14.0	0.0	7.0	5.0	3.0	4.0	2.0	7.0	5.0	0.0	70.0
SA51	20.0	10.0	11.0	24.0	0.0	14.0	8.0	5.0	8.0	5.0	17.0	8.0	0.0	130.0
SA52	16.0	8.0	6.0	17.0	0.0	12.0	6.0	4.0	5.0	4.0	9.0	6.0	0.0	93.0
SA53	12.0	4.0	4.0	10.0	0.0	7.0	3.0	2.0	3.0	2.0	8.0	4.0	0.0	59.0
SA54	20.0	10.0	10.0	22.0	0.0	15.0	6.0	4.0	7.0	4.0	12.0	9.0	0.0	119.0
SA55	16.0	6.0	6.0	16.0	0.0	8.0	5.0	2.0	4.0	3.0	9.0	6.0	0.0	81.0
SA56	20.0	11.0	12.0	24.0	0.0	17.0	9.0	4.0	7.0	5.0	15.0	8.0	0.0	132.0
SA57	8.0	2.0	4.0	9.0	0.0	5.0	2.0	2.0	2.0	2.0	4.0	3.0	0.0	43.0
SA58	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0

SA59	0.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0	0.0	1.0	0.0	0.0	5.0
SA60	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
SA61	12.0	9.0	9.0	40.0	2.0	14.0	6.0	4.0	6.0	4.0	17.0	5.0	4.0	132.0
SA62	11.0	8.0	6.0	41.0	1.0	10.0	5.0	4.0	6.0	4.0	19.0	6.0	6.0	127.0
SA63	6.0	4.0	4.0	28.0	0.0	8.0	3.0	3.0	3.0	2.0	11.0	3.0	3.0	78.0
SA64	5.0	1.0	3.0	17.0	0.0	4.0	2.0	2.0	2.0	1.0	7.0	2.0	2.0	48.0
SA65	3.0	3.0	3.0	16.0	0.0	6.0	2.0	2.0	2.0	1.0	7.0	1.0	2.0	48.0
SA66	7.0	6.0	6.0	28.0	1.0	10.0	3.0	2.0	2.0	2.0	12.0	4.0	5.0	88.0
SA67	6.0	4.0	3.0	22.0	1.0	6.0	3.0	2.0	2.0	1.0	8.0	2.0	3.0	63.0
SA68	6.0	5.0	5.0	34.0	2.0	7.0	4.0	4.0	3.0	2.0	14.0	5.0	4.0	95.0
SA69	6.0	4.0	5.0	11.0	1.0	8.0	3.0	3.0	3.0	2.0	13.0	5.0	5.0	69.0
SA70	3.0	3.0	5.0	8.0	1.0	7.0	2.0	2.0	2.0	2.0	9.0	2.0	3.0	49.0
SA71	8.0	7.0	6.0	41.0	1.0	10.0	5.0	4.0	4.0	4.0	15.0	4.0	5.0	114.0
SA72	6.0	4.0	4.0	24.0	0.0	6.0	3.0	2.0	3.0	2.0	9.0	3.0	3.0	69.0
SA73	7.0	4.0	6.0	28.0	1.0	7.0	4.0	3.0	4.0	2.0	10.0	3.0	5.0	84.0
SA74	7.0	5.0	7.0	28.0	1.0	8.0	3.0	3.0	3.0	2.0	12.0	3.0	4.0	86.0
SA75	8.0	6.0	6.0	38.0	0.0	9.0	5.0	5.0	4.0	3.0	14.0	4.0	4.0	106.0
SA76	10.0	10.0	9.0	44.0	1.0	17.0	7.0	4.0	6.0	3.0	12.0	6.0	6.0	135.0
SA77	5.0	5.0	4.0	24.0	0.0	7.0	3.0	3.0	3.0	1.0	8.0	3.0	4.0	70.0
SA78	4.0	4.0	5.0	10.0	0.0	8.0	2.0	2.0	2.0	1.0	8.0	2.0	1.0	49.0

SA79	10.0	6.0	7.0	16.0	0.0	12.0	5.0	4.0	5.0	3.0	19.0	4.0	1.0	92.0
SA80	1.0	2.0	1.0	4.0	0.0	2.0	1.0	0.0	0.0	1.0	2.0	1.0	1.0	16.0
SA81	6.0	3.0	4.0	9.0	0.0	8.0	2.0	3.0	3.0	2.0	12.0	2.0	1.0	55.0
SA82	8.0	6.0	6.0	12.0	1.0	11.0	3.0	4.0	4.0	3.0	18.0	4.0	1.0	81.0
SA83	5.0	6.0	6.0	15.0	0.0	7.0	3.0	3.0	2.0	1.0	13.0	4.0	0.0	65.0
SA84	2.0	2.0	2.0	8.0	0.0	4.0	3.0	1.0	0.0	0.0	3.0	1.0	3.0	29.0
SA85	1.0	1.0	1.0	4.0	0.0	2.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	11.0
SA86	1.0	2.0	2.0	4.0	1.0	4.0	0.0	1.0	0.0	1.0	2.0	0.0	2.0	20.0
SA87	1.0	1.0	1.0	4.0	1.0	2.0	1.0	0.0	0.0	0.0	2.0	1.0	2.0	16.0
SA88	2.0	2.0	1.0	5.0	0.0	4.0	0.0	1.0	0.0	0.0	1.0	1.0	2.0	19.0
SA89	1.0	1.0	1.0	2.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	9.0
SA90	18.0	29.0	17.0	37.0	1.0	28.0	8.0	9.0	10.0	5.0	29.0	9.0	3.0	203.0
SA91	12.0	8.0	8.0	15.0	0.0	11.0	4.0	4.0	5.0	3.0	20.0	6.0	2.0	98.0
SA92	13.0	10.0	10.0	19.0	0.0	18.0	8.0	6.0	8.0	4.0	26.0	7.0	2.0	131.0
SA93	2.0	3.0	2.0	6.0	0.0	2.0	0.0	0.0	0.0	1.0	2.0	1.0	3.0	22.0
SA94	14.0	10.0	9.0	12.0	0.0	13.0	5.0	4.0	5.0	3.0	13.0	5.0	2.0	95.0
SA95	3.0	5.0	4.0	6.0	1.0	6.0	1.0	2.0	2.0	7.0	5.0	2.0	0.0	44.0
SA96	1.0	2.0	2.0	4.0	0.0	2.0	1.0	1.0	1.0	1.0	2.0	0.0	2.0	19.0
SA97	4.0	5.0	5.0	11.0	0.0	6.0	2.0	2.0	2.0	2.0	10.0	3.0	0.0	52.0
SA98	2.0	2.0	2.0	8.0	1.0	4.0	1.0	0.0	0.0	0.0	3.0	1.0	2.0	26.0

SA99	1.0	2.0	1.0	4.0	0.0	2.0	1.0	1.0	1.0	0.0	1.0	1.0	2.0	17.0
SA100	1.0	1.0	1.0	3.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	2.0	10.0
SA101	1.0	2.0	2.0	5.0	0.0	0.0	1.0	0.0	1.0	1.0	2.0	1.0	2.0	18.0
SA102	1.0	2.0	1.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	8.0
SA103	1.0	1.0	1.0	4.0	0.0	4.0	1.0	1.0	0.0	0.0	3.0	1.0	0.0	17.0
SA104	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	2.0
SA105	12.0	16.0	15.0	20.0	0.0	20.0	6.0	5.0	6.0	3.0	15.0	6.0	2.0	126.0
SA106	14.0	18.0	14.0	28.0	0.0	20.0	5.0	5.0	6.0	4.0	22.0	7.0	3.0	146.0
SA107	18.0	19.0	20.0	39.0	1.0	20.0	6.0	7.0	7.0	3.0	31.0	9.0	5.0	185.0
SA108	1.0	2.0	1.0	4.0	0.0	2.0	1.0	0.0	0.0	0.0	1.0	1.0	2.0	15.0
SA109	4.0	4.0	3.0	11.0	0.0	6.0	2.0	1.0	2.0	1.0	8.0	2.0	1.0	45.0
SA110	17.0	24.0	18.0	38.0	1.0	28.0	8.0	7.0	7.0	5.0	25.0	9.0	4.0	191.0
SA111	12.0	17.0	9.0	20.0	0.0	14.0	5.0	4.0	4.0	2.0	16.0	6.0	2.0	111.0
SA112	9.0	11.0	10.0	22.0	0.0	15.0	5.0	5.0	6.0	2.0	18.0	4.0	1.0	108.0
SA113	21.0	32.0	20.0	48.0	2.0	32.0	8.0	8.0	10.0	4.0	30.0	10.0	6.0	231.0
SA114	18.0	27.0	21.0	32.0	0.0	26.0	7.0	9.0	10.0	5.0	23.0	9.0	5.0	192.0
SA115	1.0	1.0	2.0	4.0	1.0	2.0	1.0	0.0	0.0	1.0	2.0	1.0	2.0	18.0
SA116	18.0	18.0	18.0	40.0	1.0	26.0	10.0	9.0	7.0	3.0	32.0	8.0	2.0	192.0
SA117	20.0	27.0	29.0	56.0	1.0	31.0	9.0	8.0	10.0	6.0	48.0	12.0	4.0	261.0
SA118	1.0	1.0	2.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	2.0	11.0

SA119	15.0	17.0	20.0	42.0	2.0	30.0	7.0	6.0	8.0	6.0	21.0	9.0	42.0	225.0
SA120	3.0	5.0	4.0	7.0	1.0	7.0	1.0	2.0	1.0	2.0	5.0	3.0	6.0	47.0
SA121	3.0	3.0	3.0	4.0	0.0	5.0	1.0	1.0	2.0	1.0	5.0	3.0	10.0	41.0
SA122	13.0	11.0	10.0	20.0	2.0	19.0	7.0	7.0	4.0	4.0	9.0	3.0	14.0	123.0
SA123	11.0	12.0	13.0	22.0	2.0	23.0	8.0	7.0	7.0	7.0	19.0	6.0	22.0	159.0
SA124	7.0	6.0	7.0	12.0	1.0	11.0	2.0	4.0	2.0	2.0	7.0	3.0	11.0	75.0
SA125	2.0	3.0	3.0	6.0	0.0	4.0	1.0	1.0	1.0	1.0	3.0	1.0	6.0	32.0
SA126	11.0	13.0	13.0	24.0	2.0	20.0	7.0	4.0	5.0	6.0	14.0	7.0	22.0	148.0
SA127	11.0	14.0	13.0	24.0	2.0	18.0	5.0	4.0	5.0	5.0	17.0	6.0	22.0	146.0
SA128	4.0	4.0	4.0	32.0	1.0	9.0	2.0	3.0	2.0	1.0	7.0	3.0	7.0	79.0
SA129	13.0	17.0	18.0	36.0	3.0	28.0	5.0	6.0	8.0	7.0	22.0	8.0	42.0	213.0
SA130	9.0	16.0	11.0	24.0	1.0	14.0	5.0	3.0	4.0	4.0	11.0	5.0	23.0	130.0
SA131	4.0	6.0	5.0	8.0	0.0	7.0	3.0	2.0	2.0	1.0	6.0	2.0	9.0	55.0
SA132	14.0	16.0	17.0	33.0	1.0	27.0	6.0	5.0	8.0	5.0	21.0	7.0	33.0	193.0
SA133	2.0	2.0	3.0	6.0	0.0	3.0	1.0	0.0	1.0	1.0	4.0	2.0	5.0	30.0
SA134	4.0	6.0	6.0	12.0	1.0	10.0	3.0	2.0	3.0	3.0	9.0	3.0	11.0	73.0
SA135	3.0	2.0	3.0	6.0	0.0	5.0	2.0	1.0	1.0	1.0	3.0	2.0	5.0	34.0
SA136	9.0	11.0	13.0	22.0	2.0	26.0	6.0	5.0	7.0	5.0	18.0	5.0	27.0	156.0
SA137	11.0	13.0	10.0	20.0	2.0	20.0	6.0	4.0	3.0	4.0	14.0	6.0	22.0	135.0
SA138	8.0	9.0	11.0	20.0	1.0	20.0	6.0	5.0	5.0	5.0	12.0	5.0	22.0	129.0

SA139	10.0	13.0	12.0	18.0	1.0	18.0	6.0	5.0	5.0	4.0	15.0	6.0	21.0	134.0
SA140	2.0	3.0	3.0	4.0	0.0	5.0	2.0	1.0	0.0	0.0	3.0	1.0	6.0	30.0
SA141	8.0	8.0	10.0	24.0	1.0	15.0	4.0	3.0	5.0	3.0	12.0	5.0	23.0	121.0
SA142	4.0	5.0	5.0	11.0	0.0	9.0	2.0	2.0	2.0	2.0	6.0	2.0	10.0	60.0
SA143	6.0	6.0	6.0	14.0	0.0	10.0	2.0	4.0	4.0	3.0	6.0	4.0	14.0	79.0
Total	1006	957	925	2396	107	1406	522	426	476	342	1621	542	747	

Table D.1: Demand forecast of each (FC, service area) pair. (In this table, we omit service areas that has 0 forecast for all FCs.)

Appendix E

Tools for Solving Large Scale QP

The quadratic program in the QP algorithm can be hard to solve when the number of variables (resources, routes and commodities) are large. We propose two iterative methods that aim to make the process of obtaining near optimal (dual) solutions more computationally efficient. We remind readers that the quadratic program in the QP algorithm is equation (4.6). The time element τ in quadratic program denotes the time when the quadratic program is solved, which is irrelevant to this section. Therefore, we drop τ in this section for simplicity:

$$\begin{aligned} W^{QP}(\mathbf{d}, \bar{\mathbf{f}}) = \min \quad & \frac{1}{2}\sum_{j\in J} v_j g_j^2 + \sum_{k\in K}\sum_{r\in R_k} c_{kr}x_{kr} \\ \text{s.t.} \quad & \sum_{r\in R_j}\sum_{k\in K_r} x_{kr} - g_j \le \bar{f}_j \quad \forall j \in J \\ & \sum_{r\in R_k} x_{kr} = d_k \quad \forall k \in K \\ & x_{kr}, g_j \ge 0 \quad \forall k, r, j \end{aligned} \tag{E.1}$$

E.1 Method 1: The Subgradient Method

The quadratic program can be decomposed into multiple subproblems when resource constraints are relaxed:

$$\tilde{W} = \min \quad \frac{1}{2}\sum_{j\in J} v_j g_j^2 + \sum_{k\in K}\sum_{r\in R_k} c_{kr}x_{kr} + \lambda_j(\sum_{r\in R_j}\sum_{k\in K_r} x_{kr} - g_j - \bar{f}_j)$$
$$\text{s.t.} \quad \sum_{r\in R_k} x_{kr} = d_k \quad \forall k \in K \tag{E.2}$$
$$x_{kr}, g_j \geq 0 \quad \forall k, r, j,$$

where λ_j is a non-negative multiplier of resource constraint j. It is not hard to show that solving this relaxed problem is equivalent to solving the following $K+1$ subproblems in parallel:

Subproblem 0:

$$\min \quad \frac{1}{2}\sum_{j\in J} v_j g_j^2 - \lambda_j \sum_{r\in R_j}\sum_{k\in K_r} g_j \tag{E.3}$$
$$\text{s.t.} \quad g_j \geq 0 \quad \forall j,$$

Subproblem 1 to K (one for each commodity $k \in K$):

$$\min \quad \sum_{r\in R_k} c_{kr}x_{kr} + \lambda_j \sum_{r\in R_j} x_{kr}$$
$$\text{s.t.} \quad \sum_{r\in R_k} x_{kr} = d_k \tag{E.4}$$
$$x_{kr} \geq 0 \quad \forall r$$

We note that subproblem 0 has closed form solution: $g_j = \lambda_j / v_j$. The subgradient method starts with an initial vector of multiplier $\{\lambda_j^0 \forall j \in J\}$, then update the multipliers with the optimal solutions of the subproblems until the optimality gap is smaller than some desired threshold ϵ In summary, the subgradient method goes through the following steps until termination:

1. Set an initial multipler λ_j^0 for every resource j. One choice of initial multipler is the variable cost for the corresponding resource, i.e., $\lambda_j^0 = c_j$ for all j.

2. Solve the subproblems for optimal solutions (x_{kr}^* and g_j^*). Note that we have an upper and lower bound of the optimal cost and a feasible solution at every

iteration:

- upper bound $W^{\text{upper}} = W^{QP}(x^*_{kr},\, \hat{g}_j) lowerbound \mathrm{W}^{\text{lower}} = \tilde{W}(x^*_{kr},\, g^*_j)$
- feasible solution x^*_{kr}, $\hat{g}_j$, where $\hat{g}_j = (\sum_{r \in R_j} \sum_{k \in K_r} x^*_{kr} - \bar{f}_j)^+$

3. Update the mutlipliers with optimal solutions (x^*_{kr} and g^*_j). We propose two different ways of doing it:

   (a) Method 1: Updated the multipliers with x^*_{kr} and g^*_j

   $$\lambda^{m+1}_j := \lambda^m_j + \alpha(\sum_{r \in R_j} \sum_{k \in K_r} x^*_{kr} - \bar{f}_j - g^*_j) \quad \forall j,$$

   where step size α is a function of the upper and lower bound of the optimal solution

   $$\alpha = \frac{\theta(W^{\text{upper}} - W^{\text{lower}})}{(\sum_{r \in R_j} \sum_k \in K_r x^*_{kr} - \bar{f}_j - g^*_j)^2}$$

   where θ is a scalar satisfying $0 < \theta \leq 2$. See [11] for further discussion about the choice of step sizes.

   (b) Method 2: Update the multipliers with x^*_{kr}

   First, we create the "intermediate multipliers" $\hat{\lambda^m_j}$ with feasible excess flow solution $\hat{g}_j$ inspired by the closed form solution of subproblem 0:

   $$\hat{\lambda^m_j} := \frac{\hat{g}_j}{v_j} \quad \forall j$$

   Then we update the multiplier with the "intermediate multipliers" with some step size $\alpha \in (0, 1)$:

   $$\lambda^{m+1}_j = \alpha \lambda^m_j + (1 - \alpha)\hat{\lambda^m_j} \quad \forall j$$

4. If the optimality gap is smaller some desired threshold ϵ, i.e.,

$$W^{\text{upper}} - W^{\text{lower}} < \epsilon,$$

then the heuristic terminates, otherwise, return to step 2.

E.2 Method 2: The ADMM Method (Multi-blocks)

A dilemma in the subgradient method is that we do not have a clear guidance on the choice of step size that guarantees convergence for multipliers. On the other hand, Alternating Direction Method of Multipliers (ADMM) has strong convergence properties, including the one we care the most - multipliers will converge to optimal regardless of the choice of step size. (An introduction to ADMM can be found in [4].) The standard ADMM framework described in [4] can not be directly applied to our problem since it is limited to two blocks of variables, while our problem was decomposed into $K+1$ blocks after relaxing the resource constraints. Therefore, we introduce an ADMM-like method that contains $K+1$ blocks instead of two blocks. We note that method doesn't enjoy the same convergence guarantees as the standard ADMM method due to the differences in problem structure, and therefore, would require more research on convergence guarantee like [6].

Before we dive in to the algorithm, let us re-write the full problem (equation E.1) into a simpler form, which captures the main structure of our problem:

$$\begin{aligned} \min \quad & f(\mathbf{g}) + \sum_{k \in K} h(\mathbf{x_k}) \\ \text{s.t.} \quad & \sum_{k \in K} A_k \mathbf{x_k} - \mathbf{g} \leq \bar{\mathbf{f}} \\ & \mathbf{x_k} \in X_k \quad \forall k \\ & \mathbf{x_k}, \mathbf{g} \geq 0 \quad \forall k \in K, \end{aligned} \tag{E.5}$$

where $\mathbf{g}$ is a vector (with size $|J|$) that contains all the excess flow variables (g_j), $\mathbf{x_k}$ is a vector (with size $|R|$) that concatenates all the assignment variables (x_{kr}) of

commodity k, $\bar{\mathbf{f}}$ is a vector (with size $|J|$) of target flows. The resource constraint in (E.1) is replaced by the first constraint in (E.5), the demand constraint in (E.1) is replaced by the second constraint in (E.5). Unlike the regular Lagrange relaxation applied in the subgradient method, we apply the augmented Lagrangian:

$$L(\mathbf{x}, \mathbf{g}\lambda) = f(\mathbf{g}) + \sum_{k \in K} h(\mathbf{x_k}) + \lambda^T (\sum_{k \in K} A_k \mathbf{x_k} - \mathbf{g} \quad \leq \bar{\mathbf{f}}) + \frac{\rho}{2} ||(\sum_{k \in K} A_k \mathbf{x_k} - \mathbf{g} \leq \bar{\mathbf{f}})||^2 \tag{E.6}$$

where λ is a vector of multipliers (with size $|J|$) and $\rho > 0$. We note that the penalty parameter ρ need not be constant over iterations. We refer readers to [4] for further discussion on choice of penalty parameters. The method goes through the following steps until termination:

1. Start with an initial feasible solution of (E.5): x^0_{kr} and g^0_j and λ^0_j, where 0 denote the 0^{th} iteration

2. Minimize equation (E.6) sequentially with respect to $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_K}$ and $\mathbf{g}$ under the last two constraints in (E.5), i.e,

   $$\mathbf{x}_i^{m+1} := \underset{\mathbf{x}_i \in X_i, \mathbf{x}_i \geq 0}{\arg\min} L(\mathbf{x}_1^{m+1}, \mathbf{x}_2^{m+1}, \cdots, \mathbf{x}_{i-1}^{m+1}, \mathbf{x}_i, \mathbf{x}_{i+1}^{m}, \cdots, \mathbf{x}_K^{m}, \mathbf{g}^m, \lambda^m) \quad \forall i := 1, 2, \cdots K$$

   $$\mathbf{g}^{m+1} = \underset{\mathbf{g} \geq 0}{\arg\min} L(\mathbf{x}_1^{m+1}, \mathbf{x}_2^{m+1}, \cdots, \mathbf{x}_N^{m+1}, \mathbf{g}^m, \lambda^m)$$

   where m is the number of iteration.

3. Update multipliers by the following equation:

   $$\lambda^{m+1} := \lambda^m + (\sum_{k \in K} A_k {\mathbf{x_k}}^m - \mathbf{g}^m \leq \bar{\mathbf{f}})$$

4. If the optimality gap is smaller than some threshold ϵ, then the heuristic terminates, otherwise, return to step 2. We note that upper and lower bound is readily available in each iteration from $\mathbf{x}_k^m$ and $\mathbf{g}^m$ (the same procedure as the upper and lower bound derivation in the subgradient method).

E.3 Method 3: The ADMM Method (Two-blocks)

In this section, we show that by introducing additional variables $\mathbf{z}$, the quadratic program can be decomposed into two-blocks of variables, which fits the two-block structure described in [4], therefore, preserves the convergence guarantee of the multipliers. We replace the first constraint in E.5 by the following constraints:

$$A_k \mathbf{x_k} = \mathbf{z_k} \quad \forall k \in K \tag{E.7}$$

$$\mathbf{g} = \mathbf{z_0} \tag{E.8}$$

$$\sum_{k \in K} \mathbf{z_k} + \mathbf{z_0} \leq \bar{\mathbf{f}} \tag{E.9}$$

With this new formulation, the augmented Lagrangian (after relaxing constraint E.7 and E.8) is:

$$\begin{aligned} L(\mathbf{x}, \mathbf{g}, \mathbf{z}, \theta) = f(\mathbf{g}) + \sum_{k \in K} h(\mathbf{x_k}) + \sum_{k \in K} {\theta_\mathbf{k}}^T (A_k \mathbf{x_k} - \mathbf{z_k}) + {\theta_\mathbf{0}}^T (\mathbf{g} - \mathbf{z_0}) \\ + \frac{\rho}{2} (\sum_{k \in K} ||(A_k \mathbf{x_k} - \mathbf{z_k})||^2 + ||(\mathbf{g} - \mathbf{z_0})||^2) \\ \text{where } \mathbf{x_k} \in X_k, \quad \mathbf{g} \geq 0, \quad \mathbf{z} \in Z = \{\mathbf{z} | \sum_{k \in K} \mathbf{z_k} + \mathbf{z_0} \leq \bar{\mathbf{f}}\} \end{aligned}$$

It's easy to see that variable $\mathbf{x}, \mathbf{g}$ are now decomposed. By treating variable $\mathbf{x}$ and $\mathbf{g}$ as one block, and $\mathbf{z}$ as the other block, we obtain the same two-block structure as described in [4]. The rest is just standard ADMM procedure, which we do not repeat here.

Bibliography

[1] J. Acimovic and S. C. Graves. Making better fulfillment decisions on the fly in an online retail environment. Manufacturing & Service Operations Management, 17(1):34–51, 2015.

[2] A. Amil, A. Makhdoumi, and Y. Wei. Multi-item order fulfillment revisited: Lp formulation and prophet inequality. Available at SSRN 4176274, 2022.

[3] J. M. Andrews, V. F. Farias, A. I. Khojandi, and C. M. Yan. Primal–dual algorithms for order fulfillment at urban outfitters, inc. INFORMS Journal on Applied Analytics, 49(5):355–370, 2019.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning, 3(1):1–122, 2011.

[5] U. C. Bureau. The 2nd quarter 2022 retail e-commerce sales report. 2022. `https://www.census.gov/retail/index.html`.

[6] C. Chen, B. He, Y. Ye, and X. Yuan. The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. Mathematical Programming, 155(1):57–79, 2016.

[7] S. R. Department. Projected retail e-commerce gmv share of amazon in the united states from 2016 to 2021. 2022. `https://www.statista.com/statistics/788109/amazon-retail-market-share-usa/`.

[8] S. R. Department. Number of packages delivered by amazon logistics in the united states from 2018 to 2020. 2022. https://www.statista.com/statistics/1178979/amazon-logistics-package-volume-united-states/.

[9] S. R. Department. Amazon's shipping costs from 2011 to 2021. 2022. https://www.statista.com/statistics/806498/amazon-shipping-costs/.

[10] L. DeValve, Y. Wei, D. Wu, and R. Yuan. Understanding the value of fulfillment flexibility in an online retailing environment. Manufacturing & Service Operations Management, 2021.

[11] M. L. Fisher. The lagrangian relaxation method for solving integer programming problems. Management science, 27(1):1–18, 1981.

[12] W. Gautschi. A computational procedure for incomplete gamma functions. ACM Transactions on Mathematical Software (TOMS), 5(4):466–481, 1979.

[13] A. Govindarajan, A. Sinha, and J. Uichanco. Joint inventory and fulfillment decisions for omnichannel retail networks. Naval Research Logistics (NRL), 68 (6):779–794, 2021.

[14] P. Harsha, S. Subramanian, and J. Uichanco. Dynamic pricing of omnichannel inventories: honorable mention—2017 m&som practice-based research competition. Manufacturing & Service Operations Management, 21(1):47–65, 2019.

[15] A. Hartmans. Amazon says it will ship more packages than ups and fedex by 2022 at the latest. 2021. https://www.businessinsider.com/amazon-surpassing-ups-fedex-by-2022-dave-clark-2021-11.

[16] S. Jasin and A. Sinha. An lp-based correlated rounding scheme for multi-item ecommerce order fulfillment. Operations Research, 63(6):1336–1351, 2015.

[17] Y. Lei, S. Jasin, and A. Sinha. Joint dynamic pricing and order fulfillment for e-commerce retailers. Manufacturing & Service Operations Management, 20(2): 269–284, 2018.

[18] Y. Lei, S. Jasin, J. Uichanco, and A. Vakhutinsky. Joint product framing (display, ranking, pricing) and order fulfillment under the multinomial logit model for e-commerce retailers. Manufacturing & Service Operations Management, 24(3): 1529–1546, 2022.

[19] Y. F. Lim, S. Jiu, and M. Ang. Integrating anticipative replenishment allocation with reactive fulfillment for online retailing using robust optimization. Manufacturing & Service Operations Management, 23(6):1616–1633, 2021.

[20] W. Ma. Simple and order-optimal correlated rounding schemes for multi-item e-commerce order fulfillment. arXiv preprint arXiv:2207.04774, 2022.

[21] L. Wei, R. Kapuscinski, and S. Jasin. Shipping consolidation across two warehouses with delivery deadline and expedited options for e-commerce and omnichannel retailers. Manufacturing & Service Operations Management, 23(6):1634–1650, 2021.

[22] P. J. Xu, R. Allgor, and S. C. Graves. Benefits of reevaluating real-time order fulfillment decisions. Manufacturing & Service Operations Management, 11(2): 340–355, 2009.

[23] Y. Zhao, X. Wang, and L. Xin. Multi-item online order fulfillment in a two-layer network. Chicago Booth Research Paper, (20-41), 2020.