Politecnico di Torino

Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Matematica: Statistica e Ottimizzazione su Dati e Reti

A.a. 2024/2025

Graduation Session October 2025

Approximate Dynamic Programming for Lateral Transshipments with a Concave Cost Function

Supervisor: Prof. Paolo Brandimarte

Candidate: Ismail Abouelseoud

Abstract

Inventory management in modern retail is persistently challenged by the mismatch between supply and customer demand, a problem that leads to costly overstocks and lost sales from stockouts. This issue persists despite sophisticated forecasting tools, primarily due to the inherent uncertainty and volatility of consumer behavior. Lateral transshipment—the practice of moving inventory between locations at the same supply chain echelon—has emerged as a key strategy to enhance flexibility and mitigate these imbalances by creating a pooled inventory resource. This thesis develops and analyzes a reactive transshipment policy tailored for multi-location retail networks that manage products with short selling seasons and highly uncertain demand based on the paper "Approximate Dynamic Programming for Lateral Transshipment Problems in Multi-Location Inventory Systems" by Joern Meissner and Olga V. Senicheva. In their paper Meissner and Senicheva address the computational complexity of this problem, known as the "curse of dimensionality," by implementing an approximate dynamic programming (ADP) framework to find high-quality, feasible solutions where exact optimization is intractable. The primary contribution of this work is an extension to the existing model: the incorporation of economies of scale into the transshipment cost structure. This is achieved by introducing a concave cost function, a departure from the linear cost assumptions commonly found in the literature. This enhancement creates a more realistic but non-convex optimization problem, necessitating a novel formulation to be solved effectively. The performance of the proposed ADP model is benchmarked against other established reactive transshipment policies to validate its effectiveness. For the full code of the thesis: https://github.com/ismailabouelseoud/Optimization-models-for-multiechelon-inventory-control

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In today's complex retail environment, one of the most persistent and costly challenges facing businesses is the fundamental mismatch between customer demand and retailer stock levels. This misalignment manifests itself in two primary ways: overstocks, where retailers hold excessive inventory that ties up capital and may eventually become obsolete, and stockouts, where customer demand cannot be met due to insufficient inventory levels. These inventory imbalances occur despite the widespread adoption of sophisticated modern forecasting tools and advanced analytics, highlighting the inherent complexity of demand prediction in dynamic market conditions [1].

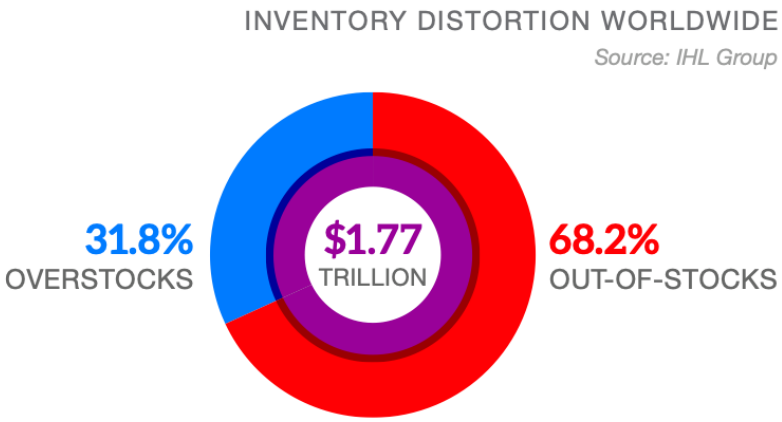


Figure 1.1: Inventory distribution. Taken from [1].

The root cause of these inventory mismatches lies in the fundamental nature of demand uncertainty. Consumer behavior is influenced by a multitude of factors that are often difficult to predict and quantify. Weather patterns can dramatically shift demand for seasonal items, economic conditions can alter purchasing power and preferences, competitive actions can redirect customer flow, and social trends can create unexpected surges or declines in product popularity. Even with the most advanced statistical models and machine learning algorithms, the stochastic nature of demand creates an environment where perfect inventory allocation remains elusive. According to industry research, extreme weather is one of the top risks to supply chains in 2025 [2], further compounding forecasting challenges.

Another weakness of conventional inventory management is that it relies heavily on historical data and fixed allocation strategies. This approach typically involves analyzing historical sales patterns, identifying seasonal trends, and establishing predetermined stock levels for every location. While this method is highly beneficial and important for establishing demand distribution patterns, it assumes that these distributions will hold over time and does not consider the high volatility of modern consumer markets [3].

1.1 Real-World Examples and Industry Impact

A compelling example of supply chain shock occurred during COVID-19 in 2020, when many retailers experienced widespread stockouts of toilet paper following a sudden spike in household purchases driven by panic buying and precautionary hoarding. Store shelves emptied even though production at paper mills had not permanently collapsed, because retail inventories and replenishment lead times were insufficient to absorb the demand surge [4].

Consequently, this demand spike propagated upstream as retailers placed larger and more frequent orders to restock, amplifying variability (the "bullwhip effect" 1.2) and causing suppliers and distribution centers to experience shortages at different points in the chain. Local stockouts occurred even when aggregate supply could have met normal demand levels if allocation and flows had been faster or more flexible.

This example demonstrates how demand shocks expose supply chain fragilities (capacity constraints, lead times, and lack of flexibility), and why measures such as additional safety stock, lateral transshipments, and near-term capacity adjustments can reduce stockouts from sudden demand fluctuations.

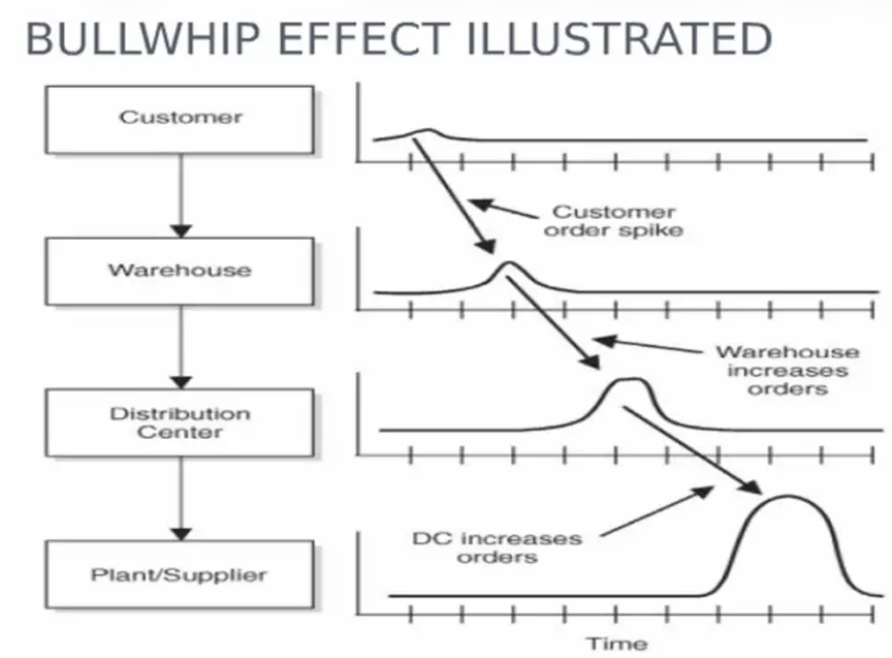


Figure 1.2: Bullwhip effect. Taken from [5].

Another compelling example of supply chain stockouts occurred during the Texas winter storm in 2021 [6], when Texas experienced a crisis where millions of homes and businesses were left without power, leading to massive stockouts of emergency supplies across the state. This crisis created extreme demand fluctuations for emergency supplies. Due to incorrect predictions of the situation's severity, retailers were completely unprepared for the surge in demand for generators, heaters, blankets, and other emergency items.

This example perfectly illustrates the research problem: while some Texas regions lost power completely, others maintained electricity. Stores in powered areas likely had excess inventory of emergency supplies (since local demand remained normal), while stores in affected areas faced complete stockouts. Weather-related disruptions like this highlight the importance of accurate demand forecasting and adaptive inventory strategies to respond to unexpected events [7].

1.2 Lateral Transshipment in Inventory Management

To enhance the efficiency and responsiveness of inventory systems, the concept of lateral transshipment has emerged as a valuable strategy. Lateral transshipment refers to the movement of stock between different stocking locations that operate at the same echelon within a supply chain network. This is distinct from the traditional hierarchical flow of goods from one echelon to the next (e.g., from a manufacturer to a wholesaler and then to a retailer). By enabling inventory sharing among entities at the same level—such as between different retail outlets or regional warehouses—lateral transshipment facilitates a more flexible and adaptive approach to managing stock [8]. This pooling of inventory can lead to significant advantages, including reduced overall inventory levels and associated holding costs while simultaneously maintaining or improving customer service [8]. The ability to fulfill demand from other locations mitigates the risk of stockouts at one facility while surplus inventory sits idle at another, thereby optimizing the utilization of the total inventory pool.

The implementation of lateral transshipment introduces additional complexity to inventory control, requiring decisions not only on when and how much to order from external suppliers but also on when, how much, and from where to transship stock. The literature identifies two primary categories based on the timing of these movements: proactive transshipment and reactive transshipment. Proactive

transshipments are executed before customer demand is realized, often to strategically position inventory based on forecasts. In contrast, reactive transshipments are triggered after demand is realized, typically in response to a stockout or a critically low inventory level at a specific location. Research shows that preventive transshipment in logistics service providers (LSP-led) systems can bring an average 40.33% increase in total profit over non-transshipment scenarios, with even higher improvements (93.31%) for products with high demand uncertainty [9]. A summary comparison of these two approaches is presented in Table 1.

Table 1.1: Comparison of Proactive and Reactive Transshipment

Feature	**Proactive Transshipment**	**Reactive Transshipment**
Timing of Transshipment	Before customer demand is realized	After customer demand is realized
Trigger for Transshipment	Anticipated demand, forecasts, predetermined schedules	Stockouts or imminent stockouts at specific locations
Primary Objective	Strategic inventory positioning, transportation efficiency	Meeting immediate demand, resolving shortages
Reliance on Demand Info	Relies heavily on demand forecasts	Acts on actual demand information
Typical Cost Trade-offs	Balancing forecast accuracy against transportation costs	Balancing transshipment costs against shortage/backorder costs

Recent research has revealed intriguing insights about these transshipment approaches. Studies indicate that proactive transshipments may not provide additional economical value compared to reactive transshipments when both cost the same, suggesting that the complexity of proactive approaches may not always be justified [10]. However, other research has found that proactive transshipment in LSP-led systems can significantly improve profits—by 23.5% compared to retailer-led systems—highlighting the importance of who makes the transshipment decisions [9]. This suggests that the value of different transshipment strategies may depend greatly on implementation details and decision-making structures within the supply chain.

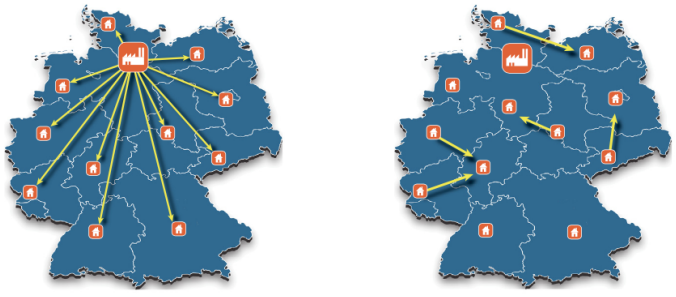
The effectiveness of lateral transshipment strategies is influenced by several factors, including demand patterns, cost structures, and network characteristics. Research shows that when the lost sale cost is higher, or the delivery cost is lower,

the marginal benefit of transshipment is more prominent, making transshipment more likely to occur with higher quantities [11]. Furthermore, studies indicate that companies employing demand planning models see a 20% cost reduction and a 10% revenue increase, highlighting the financial benefits of improved inventory management strategies that may include transshipment [12].

1.3 Thesis Scope and Methodology

This thesis focuses on reactive transshipment policies for items characterized by short selling seasons and highly uncertain demand. The research presented here is based on and serves as an extension of the work by Meissner and Senicheva in 2016 [13].

To contextualize the problem, consider a multi-location distribution system consisting of one central warehouse in Hamburg, Germany, and 15 retail outlets in different German states (see Figure 1.3) [Source of example: [13]]. At the beginning of a selling season, the central warehouse places a single replenishment order with a supplier, accounting for a long production and logistics lead time. Once the products arrive, the central warehouse allocates and ships the inventory to the 15 retailers, each of whom begins the season with a predetermined stock level. As customers purchase products each day, inventory is depleted at varying rates across the network. If a stockout occurs at one retailer while others still have stock, a reactive transshipment can be initiated between retailers to prevent a lost sale (see Figure 1.3).



(a) Distribution from central warehouse. **(b)** Lateral transshipments.

Figure 1.3: An example of a multi-location distribution network. Taken from [13].

This research addresses this specific problem by applying a dynamic programming model for a multi-location, multi-period setting with lost sales, as formulated in [13]. However, due to the "curse of dimensionality," obtaining an exact solution from the dynamic programming evaluation is computationally intractable for realistically sized problems. Therefore, this work implements approximate dynamic programming (ADP), following the framework developed in the same paper, to find high-quality, computationally feasible solutions. The performance of the proposed ADP model is compared against other reactive transshipment policies, such as the TIE and Lookahead policies.

The primary contribution of this thesis is an extension that incorporates economies of scale into the transshipment cost structure by using a concave cost function. This is a significant departure from the fixed or linear cost functions commonly used in the literature for this problem. The introduction of a concave cost function adds substantial complexity, as the objective function becomes non-convex, requiring a novel formulation to be approximated and solved effectively.

Chapter 2

Literature review

Transshipment, the practice of moving inventory between locations at the same echelon level, is a cornerstone of modern inventory management, offering a powerful mechanism to mitigate demand uncertainty and enhance service levels. A comprehensive review of the literature reveals a significant focus on reactive transshipment, where inventory movements are triggered by an immediate stockout or an observed inventory imbalance. However, an emerging and less-explored area of research is proactive transshipment, which involves preemptive stock repositioning based on anticipated future needs. This thesis contributes to this area by focusing on a proactive transshipment model, addressing notable gaps in the existing body of work concerning both strategy and cost modeling.

The majority of literature has traditionally centered on reactive transshipment strategies. The foundational work by Krishnan and Rao [14] analyzed a single-period, multi-location problem, establishing a framework that has influenced decades of research. Later work extended this scope to multiple periods and non-identical locations [15], with some studies considering the possibility of multiple reactive transshipments within a single cycle [16, 17, 18, 19]. Several works have also addressed reactive transshipments under continuous review, often motivated by spare parts applications where immediate availability is critical [20, 21, 22, 23, 24]. To handle the complexities of these systems, simulation-based approaches have been employed to analyze policy convergence and performance [25]. While these models offer robust solutions for responding to real-time stock levels, they inherently operate with a delay, as the problem they solve has already occurred.

In contrast, proactive transshipment aims to prevent stockouts before they happen. Early studies focused on simpler single-period or two-location cases [26, 27]. However, proactive transshipment problems, especially those involving multiple periods, multiple non-identical locations, and lost sales, are significantly more

complex to solve optimally. While some multi-period models considered backorders [28], the introduction of lost sales scenarios adds a substantial layer of analytical difficulty. Despite its logical appeal, proactive transshipment has received less attention, with most studies situated within periodic review systems [29]. Some recent work has begun to explore hybrid [30] or proactive [31] policies in continuous review settings, though the latter's "proactive" nature is based on responding to demand triggers rather than being scheduled before demand occurs.

The inherent complexity of proactive and large-scale reactive problems has necessitated the development of advanced solution methodologies. Dynamic programming is a common technique, but its application is often limited by the "curse of dimensionality" in large-scale problems. Consequently, many researchers have proposed heuristics to find good, practical solutions. Among these are rule-based policies such as the Transshipment-In-Time for Excess (TIE) policy [32] and more computationally intensive methods like look-ahead (PL) approaches [33].

The TIE policy is a proactive heuristic designed to rebalance inventory across locations before demand occurs. At each review point, the policy identifies locations holding "excess" inventory—defined as stock exceeding a predetermined rebalancing target, typically the safety stock level. Simultaneously, it identifies locations that are in a deficit position, with inventory levels below their targets. The TIE policy then triggers transshipments from the excess locations to the deficit locations, aiming to bring all facilities as close to their respective target stock levels as possible. Its simplicity and intuitive logic make it an attractive option for systems where a straightforward rebalancing mechanism is desired without solving a complex optimization problem at each period.

Look-ahead policies (PL) represent a more sophisticated class of heuristics that make decisions by evaluating their potential future consequences. Instead of relying on a fixed rule, a look-ahead approach simulates or forecasts the system's evolution over a defined future horizon (the "look-ahead" period) for a set of possible transshipment decisions. By calculating the expected costs or performance outcomes associated with each decision at the end of this horizon, the policy can select the action that yields the most favorable future state. This method is more computationally demanding than simple heuristics but can better navigate complex trade-offs, such as balancing current transshipment costs against the risk of future stockouts and lost sales, often leading to higher-quality solutions. For finding efficient, near-optimal solutions, even more advanced methods like approximate dynamic programming (ADP) and simulation-based optimization have become valuable tools [34, 35, 30].

A crucial aspect often simplified in the literature is the modeling of transshipment costs. The vast majority of the work cited considers a fixed transshipment cost per shipment or even ignores it entirely. However, to accurately capture the economic impact of lateral transshipment, a more realistic cost structure is needed. Real-world logistics costs are often non-linear, exhibiting economies of scale. Several works in related inventory theory have examined more complex cost functions, such as piecewise linear convex costs, and highlighted the differences compared to simpler fixed-cost assumptions [36, 37]. Recent studies have begun to integrate fixed order costs more accurately into periodic-review systems with transshipment, signaling a move toward greater model fidelity [38]. This highlights a gap for models that incorporate more realistic, non-linear transshipment costs into proactive decision-making.

This divergence in the literature highlights a critical research opportunity. While reactive strategies are well-understood, the proactive paradigm remains comparatively underdeveloped, particularly in the context of realistic cost structures and solution methodologies for complex systems. This thesis, therefore, pivots away from the well-trodden path of reactive models with simplified costs to explore the nuances of a proactive transshipment policy with a more accurately modeled cost function, aiming to provide a more comprehensive understanding of its operational advantages.

Chapter 3

Problem formulation

3.1 Introduction to the Centralized Distribution Network and Operational Flow

3.1.1 The Centralized Model

In modern supply chain management, many companies operate a network of retail locations under a unified, central authority. The problem this thesis explore is set within such a centralized distribution network. This means that a single decision-making entity is responsible for managing inventory and logistics across all retail stores. The fundamental objective of this central entity is not to maximize the profit of any single store, but to maximize the total expected profit of the entire network over a given period.

This centralized approach allows for coordinated strategies that can balance inventory, reduce costs, and improve overall service levels in a way that independently operated stores cannot. The key decisions revolve around replenishment from a central warehouse and lateral transshipments between retail locations.

3.1.2 Time Structure and Replenishment

To model this complex system, time is structured in a clear hierarchy:

- Order Cycle: This is the main time frame, such as a month or a selling season. The primary inventory replenishment from the central warehouse occurs only once, at the very beginning of each order cycle.
- Period: Each order cycle is subdivided into a finite number of smaller, discrete time periods, such as a day. Decisions about moving stock between retailers (transshipment) are made at the beginning of each of these periods.

The replenishment policy is an "order-up-to" system. At the start of an order cycle, the central authority orders enough stock to bring the inventory level at each retail location up to a predetermined target level. For this analysis, I assume these order-up-to levels are already set, allowing us to focus specifically on optimizing the transshipment policy—the rules governing when and how much product to move between retailers within an order cycle.

3.1.3 The Sequence of Events in a Period

Within each period, a specific and repeating sequence of events occurs, creating a cyclical flow of operations. This process ensures that decisions are made with the most current information and that costs and revenues are accounted for systematically.

The operational flow can be broken down into the following eight steps (Fig.3.1):

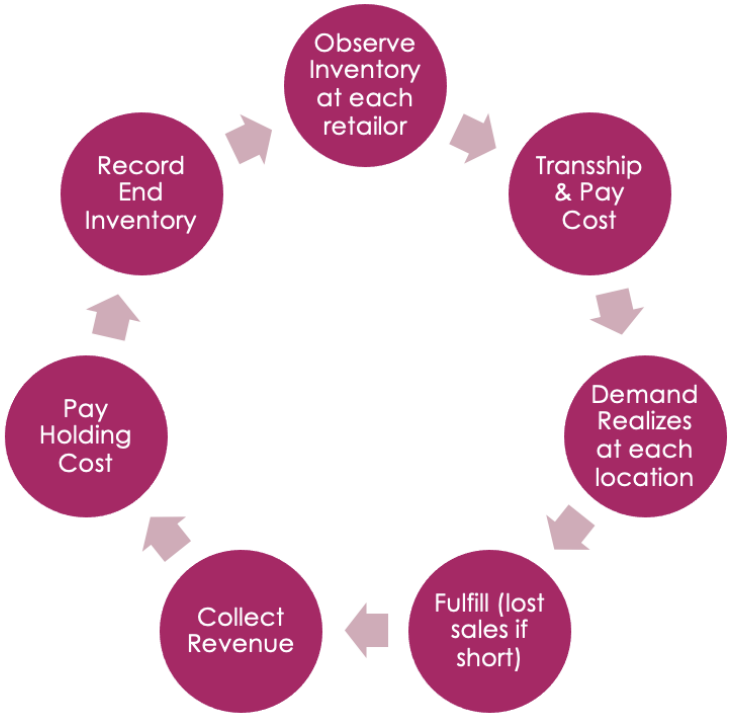


Figure 3.1: Problem formulation flow chart.

1. Observe Inventory: The period begins with the central entity observing the current stock level at every retail location.

2. Decide and Execute Transshipment: Based on the observed inventory levels and anticipated demand, the decision is made whether to move stock from an overstocked retailer to one at risk of a stockout. If a transshipment is initiated, the items are moved.

3. Pay Transshipment Cost: The costs associated with executing the transshipment are incurred immediately.

4. Demand Realizes: Customer demand for the product occurs at each retail location.

5. Fulfill Demand: Retailers use their on-hand inventory (which now includes any items received via transshipment) to satisfy customer demand.

6. Collect Revenue: Revenue is generated for each unit of demand that is successfully met.

7. Pay Holding Cost: If a retailer has excess inventory left over after demand is met, a holding cost is incurred for carrying this stock into the next period.

8. Record End Inventory: The final inventory level at each location is recorded, which becomes the starting inventory for the subsequent period. This cycle then repeats.

3.2 Core Assumptions of the Model

To make the complex reality of a distribution network mathematically tractable, a set of simplifying assumptions is required. These assumptions define the boundaries of the model and are crucial for understanding its formulation and limitations. The following core assumptions are made:

1. **Single Product:**
   The model considers the inventory management of a single, uniform product across all retail locations. The units are discrete (i.e., they can be counted in integers, like cars or books) and identical. This eliminates the complexity of managing multiple SKUs, substitutions, or product variations.

2. **Lost Sales (No Backorders):**
   If a customer's demand cannot be met immediately from a retailer's available stock, the sale is considered lost. The customer does not wait for the product to be restocked (a backorder). This is a critical assumption as it simplifies the cost structure; we only need to account for the lost revenue, not the additional administrative and fulfillment costs associated with backorders.

When a retailer stocks out, they cannot serve any more customers until their inventory is replenished.

3. **Zero Lead Times:**
   The time required for both replenishment from the central warehouse and transshipment between retailers is considered negligible or instantaneous. This means that if a decision is made to move stock, it is immediately available at its destination. This is a significant simplification, as it removes the complexity of tracking in-transit inventory. While unrealistic for geographically vast networks, it is a common assumption in the academic literature that allows for a focus on the logic of the transshipment decision itself.

4. **Distance-Dependent Transshipment Costs:**
   The cost to ship an item between any two retailers is directly proportional to the distance between them. This reflects the real-world costs of transportation, where longer distances typically incur higher fuel, labor, and time costs. This assumption ensures that the model will favor shorter, less expensive transshipments over longer, more costly ones.

5. **Known Statistical Demand:**
   While the exact demand in any future period is uncertain, its underlying probability distribution is known. This means that historical sales data or market analysis has provided a reliable statistical model of demand (e.g., a Poisson or Normal distribution) for each location. This allows the system to make decisions based on expected demand rather than perfect foresight.

6. **Independent Demand:**
   The customer demand at one retail location is statistically independent of the demand at any other location. This is a reasonable assumption when the retailers are located in different geographical regions, serving distinct customer bases. It simplifies the joint probability calculations, as the demand at one store provides no information about the demand at another.

3.3 Dynamic Programming Formulation

The problem of determining the optimal transshipment policy involves making a sequence of decisions over time under uncertainty. The outcome of each decision (i.e., the profit) depends on the current state of the system and influences its future states. This sequential, state-dependent decision-making structure makes Dynamic Programming (DP) an ideal framework for formulating and solving the problem.

A DP model breaks a complex multi-period problem into smaller, more manageable subproblems. The core idea is to find the optimal action for every possible state at each period, working backward from the end of the horizon.

3.3.1 Key Components of the DP Formulation

Fig. 3.2 explains the key components of the Dynamic Programming (DP) formulation, and the details are reported below:

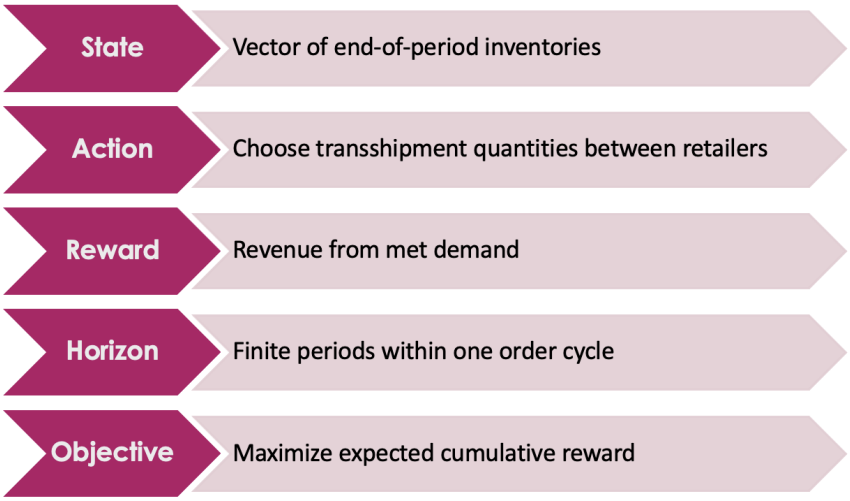


Figure 3.2: Dynamic programming key components summary.

1. **State:**
   The state is a "snapshot" of the system at a particular point in time that contains all the necessary information to make an optimal decision for the future. In this problem, the state is defined as the vector of end-of-period inventory levels at all retail locations.

   - Example: If there are three retailers, a possible state could be (10, 5, 12), representing the number of units on hand at each location.

2. **Action:**
   The action is the decision that is made in a given state. Here, the action is to choose the transshipment quantities between all pairs of retailers. This is a complex decision, as it involves specifying how many units to move from retailer i to retailer j for all possible i and j.

   - Example: An action could be "move 3 units from retailer 1 to retailer 2, and 0 units between all other pairs."

3. **Reward (or Profit Function):**
   The reward is the immediate net profit obtained in a single period as a result of taking an action in a particular state. It is calculated after demand is realized and is a combination of revenues and costs.

   - **Reward** = (Revenue from met demand) - (Transshipment costs) - (Holding costs)
     - **Revenue:** Earned from selling products to customers.
     - **Transshipment Costs:** Incurred if an action to move stock was taken.
     - **Holding Costs:** Incurred for any inventory remaining at the end of the period.

4. **Horizon:**
   The problem is defined over a finite horizon. This means we are optimizing decisions for a fixed number of periods within a single order cycle. The finite nature of the problem is what allows the DP approach to work backward from a known end point.

5. **Objective:**
   The ultimate goal of the DP formulation is to find a transshipment policy (a rule that specifies which action to take in any given state) that maximizes the expected cumulative reward over the entire finite horizon. It seeks the sequence of decisions that will yield the highest possible total profit, considering the uncertainty in demand.

3.4 Extending the Model with a Nonlinear Cost Function

3.4.1 The Limitation of a Linear Cost Assumption

The initial problem formulation assumes that the transshipment cost is directly proportional to the distance and, implicitly, to the number of units shipped. This results in a linear cost function, where shipping 10 items costs exactly 10 times as much as shipping one item. While simple and mathematically convenient, this often fails to capture the realities of modern logistics.

In practice, shipping costs frequently exhibit economies of scale. This economic principle states that as the volume of a shipment increases, the average cost per unit decreases. For example, the cost to send a single package is high, but the cost per item in a full truckload is significantly lower.

3.4.2 Introducing a Nonlinear, Concave Cost Function

To better align the model with reality, a nonlinear transshipment cost function is introduced. This function is specifically designed to model economies of scale.

- Approach: The per-item shipping cost is modeled to decrease as the shipment quantity grows, eventually settling at a fixed minimum for any additional items beyond a certain threshold.
- Consequence: This behavior creates a concave cost function. In a concave function, the marginal cost (the cost of adding one more unit) decreases as the quantity increases.

Chapter 4

Transshipment Policies

4.1 The Mathematical Model and Notation

To formally define the lateral transshipment problem, we use a mathematical model that captures the system's state, the decisions made, and the random nature of customer demand over time. This section introduces the notation used to build the model, following the framework established by Powell (2011) [39]. The dynamic programming formulation of the problem is based on the work done by Meissner and Senicheva in [13]

4.1.1 System Parameters and Variables

The model is built upon a set of parameters that define the network structure and its economic characteristics, as well as variables that describe the state of the system at any given time.

Parameters: These are the fixed constants of the model.

- **L**: The total number of retailers in the network.
- **T**: The total number of periods within one order cycle.
- S_i: The initial "order-up-to" inventory level for retailer i at the start of the order cycle (t=0).
- h_i: The cost to hold one unit of inventory at location i for one period.
- p_i: The sale price of one unit at location i.
- **c**: The base cost of transshipping one unit of inventory per unit of distance.
- ρ_{ij}: The physical distance between location i and location j.

State and Decision Variables: These variables change over time based on decisions and random events.

- **Pre-Decision State (x_{it}):** This represents the inventory on hand at location i at the beginning of period t, *before* any transshipment decisions have been made. The vector $\mathbf{x_t}$ captures the inventory levels across all locations.
- **Post-Decision State (y_{it}):** This is the inventory at location i in period t *after* a transshipment has occurred but *before* customer demand is realized. This variable reflects the immediate impact of the transshipment decision.
- **Decision Variable (z_{ijt}):** This is the core decision to be made in each period. It represents the quantity of product transshipped from location i to location j at time t. The matrix $\mathbf{Z_t}$ represents the complete set of transshipment decisions across the entire network for that period.

4.1.2 System Dynamics: The Transition Function

The model describes how the inventory levels evolve from one period to the next. This evolution is captured by a **transition function**, which breaks the process into two steps:

1. **From Pre-Decision to Post-Decision State:** The post-decision inventory y_{it} is calculated by taking the starting inventory x_{it} and adjusting it for all incoming and outgoing transshipments.

$$y_{it} = x_{it} + \sum_{j \in L} (z_{jit} - z_{ijt}) \tag{4.1}$$

   This equation states that the inventory after transshipment is the initial inventory plus what was received from all other locations, minus what was sent to all other locations.

2. **From Post-Decision to the Next Pre-Decision State:** After transshipments are complete, customer demand (d_{it}) is realized and satisfied. The inventory remaining becomes the starting inventory for the next period, x_{it+1}. Since inventory cannot be negative (due to the lost sales assumption), this is expressed as:

$$x_{it+1} = (y_{it} - d_{it})^+ \tag{4.2}$$

   The notation $(...)^+$ indicates taking the maximum of the value and zero, ensuring the inventory level does not drop below zero.

4.2 The Objective Function and Dynamic Programming Formulation

With the system's components defined, the central goal is to find a policy that maximizes the total profit over the entire time horizon. This is achieved by formulating an objective function and using dynamic programming to solve it.

4.2.1 Cost and Reward Functions

The total profit in any given period is the difference between the rewards gained and the costs incurred.

- **Transshipment Cost (C_t):** This cost is incurred at the beginning of the period when the transshipment decision is executed. It is the sum of the costs for all individual movements, where each cost is a function of the quantity, distance, and base cost per unit distance.

$$C_t(x_t, Z_t) = \sum_{i \in L} \sum_{j \in L} c \cdot \rho_{ij} \cdot z_{ijt} \tag{4.3}$$

  A key constraint is that the total amount shipped out of a location cannot exceed its available inventory:

$$\sum_{j \in L} z_{ijt} \leq x_{it} \tag{4.4}$$

- **Reward Function (R_t):** The reward is calculated after demand materializes. It includes the revenue from sales minus the holding costs for any unsold inventory.

$$R_t(y_t, d_t) = \sum_{i \in L} p_i \cdot \min(y_{it}, d_{it}) - \sum_{i \in L} h_i \cdot (y_{it} - d_{it})^+ \tag{4.5}$$

  The term $\min(y_{it}, d_{it})$ captures the actual number of units sold (which is the lesser of available stock and demand), and $(y_{it} - d_{it})^+$ represents the leftover inventory.

- **Period Profit (P_t):** The net profit for period t is simply the reward minus the cost:

$$P_t = R_t - C_t \tag{4.6}$$

4.2.2 The Objective Function

The overall goal is to find a transshipment policy, denoted by π, that maximizes the sum of the expected profits over the entire T-period horizon. A policy π is a rule that specifies what decision $\mathbf{Z_t}$ to make for any given state $\mathbf{x_t}$. The objective is:

$$\max_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=0}^{T-1} P_t(x_t, d_t, Z_t^{\pi})\right] \tag{4.7}$$

4.2.3 Dynamic Programming (DP) Formulation

This stochastic optimization problem can be solved recursively using the principles of dynamic programming. The core of DP is Bellman's equation, which defines the value of being in a certain state at a certain time.

Let $V_t(x_t)$ be the **value function**, representing the maximum possible expected future profit starting from state x_t at the beginning of period t. The recursive relationship is:

$$V_t(x_t) = \max_{Z_t} \left(-C_t(x_t, Z_t) + V_t^z(y_t)\right) \tag{4.8}$$

where $V_t^z(y_t)$ is the **post-decision value function**, which is the expected value after making decision Z_t and moving to the post-decision state y_t. It is defined as:

$$V_t^z(y_t) = \mathbb{E}_{d_t}\left[R_t(y_t, d_t) + V_{t+1}(x_{t+1})\right] \tag{4.9}$$

This formulation breaks the problem down: at each step t, we choose the transshipment Z_t that maximizes the sum of the immediate (negative) cost and the expected future value. By solving this equation backward in time from period T-1 (where $V_T = 0$), we can, in principle, find the optimal policy for any state and time.

4.3 The Curse of Dimensionality and Heuristic Policies

While the dynamic programming (DP) formulation provides a path to an optimal solution, its practical application is severely limited by a problem known as the **"curse of dimensionality."**

The state space of the problem (the set of all possible inventory vectors $\mathbf{x_t}$) and the action space (the set of all possible transshipment matrices $\mathbf{Z_t}$) grow exponentially with the number of locations (L). For even a moderately sized network (e.g., 10-15 locations), the number of states and actions becomes astronomically large, making it computationally impossible to calculate and store the value function $V_t(x_t)$ for every state.

This computational barrier means that finding the true optimal solution is only feasible for very small, trivial networks. For any real-world application, we must turn to heuristic or approximation methods that can provide high-quality, though not necessarily optimal, solutions in a reasonable amount of time.

4.3.1 Benchmark Transshipment Policies

To evaluate the performance of more advanced methods, several simpler, practically-oriented heuristic policies are used as benchmarks.

1. **No-Transshipment Policy (NT):** This is the simplest baseline. No lateral transshipments are ever made. Each location operates independently, using only its initial stock to meet demand. This policy serves as the lower bound for performance, showing the value that any transshipment strategy adds.

2. **Closest Location Transshipment Policy (RC):** This is a **reactive** heuristic. A transshipment is only triggered when a location has zero inventory at the beginning of a period. In this case, a pre-defined quantity of stock is shipped from the nearest location that has available inventory. It is simple and intuitive but may not be forward-looking.

3. **Modified Lateral TIE Policy:** This is a **proactive** heuristic inspired by the "Transshipments for Inventory Equalization" (TIE) policy. It triggers a redistribution of inventory whenever any location's stock falls below its expected daily demand. The goal is to balance the "run-out time" (the number of days of supply) across all locations. This modified version improves upon the original by explicitly incorporating transshipment costs, prioritizing shipments between closer locations to fulfill the required stock equalization.

4. **Lookahead Policy (PL):** This is a more sophisticated **proactive** heuristic. At the beginning of each period, it evaluates the marginal benefit of moving a single unit between every possible pair of locations. It calculates the expected one-period profit change from this move and compares it to the transshipment cost. If the profit gain is higher than the cost, the transshipment is made. The process is repeated iteratively, increasing shipment quantities one unit at a time until no more profitable moves can be found.

Chapter 5

Approximate Dynamic Programming

Dynamic Programming (DP) has long been recognized as a fundamental framework for sequential decision-making under uncertainty. However, classical DP approaches become computationally intractable when dealing with high-dimensional state spaces, large action spaces, or complex stochastic processes. The curse of dimensionality manifests as exponential growth in computational requirements as problem dimensions increase, making exact solutions impractical for real-world applications.

Approximate Dynamic Programming (ADP) addresses these limitations by replacing exact value functions with carefully constructed approximations. Rather than computing optimal values for every possible state, ADP methods learn value function approximations through simulation and iterative improvement, making it possible to tackle problems that would otherwise be computationally prohibitive.

This chapter presents a comprehensive examination of ADP theory, focusing on forward ADP algorithms, value function approximation techniques, and the Concave Adaptive Value Estimation (CAVE) algorithm for maintaining concavity in piecewise-linear approximations. The work in this chapter is mainly taken from [13] and [39].

5.1 Theoretical Foundations

5.1.1 The Dynamic Programming Framework

In the standard DP formulation, we seek to maximize the expected cumulative reward over a finite or infinite horizon. The problem is typically characterized by a set of core components:

- **State Space ($\mathcal{S}$):** The set of all possible configurations or states the system can be in. A state $s \in \mathcal{S}$ should contain all information necessary to make an optimal decision.
- **Action Space ($\mathcal{A}$):** The set of all available decisions or actions that can be taken in a given state.
- **Transition Function ($P(s'|s,a)$):** A function that defines the system's dynamics. It gives the probability of transitioning to a future state s' given the system is currently in state s and action a is taken.
- **Reward Function ($R(s,a)$):** A function that provides the immediate reward (or cost) received for taking action a in state s.
- **Value Function ($V(s)$):** The central construct of DP. It represents the expected total cumulative reward that can be achieved starting from state s and following an optimal policy thereafter.

The relationship between the value of a state and the values of its potential successor states is elegantly captured by the **Bellman optimality equation**:

$$V^*(s) = \max_{a \in \mathcal{A}} \left[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s') \right] \tag{5.1}$$

where γ is a discount factor ($0 \leq \gamma \leq 1$) that trades off the importance of immediate versus future rewards, and V^* represents the optimal value function. Solving this equation for all states yields the optimal policy. This formulation is the same as the one derived in the previous chapter in 4.2.3 Dynamic Programming (DP) Formulation.

5.1.2 The Curse of Dimensionality

The computational complexity of solving the Bellman equation exactly grows exponentially with the number of dimensions in the problem. This "curse" makes exact DP computationally intractable for most real-world applications, particularly in cases involving:

- **High-dimensional state spaces:** If a problem has n state variables, each taking k possible values, the total state space size is k^n. Calculating and storing the value for every state becomes impossible.
- **Large or continuous action spaces:** The maximization step in the Bellman equation can become a complex optimization problem in itself.
- **Complex stochastic processes:** The expectation calculation (the summation over s') can be intractable if the number of possible outcomes is very large or continuous.

5.1.3 The ADP Solution Paradigm

ADP circumvents the curse of dimensionality by fundamentally changing the solution strategy. Instead of exact calculation, it relies on approximation and simulation, built upon three key pillars:

1. **Value Function Approximation:** The exact value function $V^*(s)$ is replaced with a parameterized approximation, $\hat{V}(s|\theta)$, which is more computationally tractable to store and evaluate.
2. **Sampling:** Instead of enumerating all possible states and outcomes, ADP uses Monte Carlo simulation to generate sample paths through the state space, focusing computational effort on states that are more likely to be encountered.
3. **Learning:** The parameters θ of the approximation are updated iteratively based on the rewards and outcomes observed during simulation, allowing the approximation to converge toward the true value function over time.

5.2 Forward ADP Algorithm

5.2.1 Algorithm Overview

Forward ADP simulates the problem's process forward in time, making decisions at each step based on the current estimate of the value function and then updating that estimate based on the observed outcome. This forward-pass, learning-based approach contrasts sharply with backward DP, which requires working backward from a known terminal state. The forward nature makes it exceptionally well-suited for problems where the process evolves over time and can be simulated.

5.2.2 Algorithm Structure

The forward ADP algorithm follows the structure presented in Algorithm 5.1:

Algorithm 1: Forward ADP algorithm around post-decision state.

Input : L, T, c, N, α, ϵ,
S_i, h_i, p_i, ρ_{ij} for $i, j \in \mathcal{L}$;

Output: $\overline{V}_t^N(Y)$ for $\forall\, t \in \mathcal{T}$;

begin

initialize $\overline{V}_t^0(Y) = 0$ for $\forall\, t \in \mathcal{T}$;

for $n = 1, \ldots, N$ **do**

initialize $x_{i0} = S_i$ for $\forall\, i \in \mathcal{L}$;

for $t = 0, \ldots, T-1$ **do**

solve the maximization problem:

$$\hat{v}_t = \max_{\mathbf{z}_t} \Big(-C_t(\mathbf{x}_t, \mathbf{z}_t) + \overline{V}_t^{n-1}(\mathbf{y}_t) \Big)$$

if $t > 0$ **then**

update approximation $\overline{V}_{t-1}^{n-1}(\mathbf{y_{t-1}})$:

$$\overline{V}_{t-1}^{n}(\mathbf{y}_{t-1}) = (1-\alpha)\overline{V}_{t-1}^{n-1}(\mathbf{y}_{t-1}) + \alpha\hat{v}_t$$

if *Uniform(0,1)* $> \epsilon$ **then**

calculate post-decision state $\mathbf{y}_t$:

$$\mathbf{y}_t = X^{M,z}(\mathbf{x}_t, \mathbf{z}_t)$$

else

take next post-decision state $\mathbf{y}_t$ randomly;

generate random demand vector $\mathbf{d}_t$;

calculate next pre-decision state $\mathbf{x}_{t+1}$:

$$\mathbf{x}_{t+1} = X^{M,d}(\mathbf{y}_t, \mathbf{d}_t)$$

Figure 5.1: Forward ADP algorithm taken from [13].

5.2.3 Key Components

ε-greedy Exploration: The algorithm must balance **exploitation** (making the best decision based on current knowledge) with **exploration** (trying new actions to discover potentially better states). It does this using an ε-greedy strategy. With probability $1 - \varepsilon$, it chooses the greedy action that maximizes the estimated value. With probability ε, it chooses a random action to explore the state-action space more broadly. Typically, ε_n decreases as the iteration count n increases, shifting the focus from exploration to exploitation as the value function approximation improves [13].

Stepsize Rule: The algorithm uses a stepsize α_n to determine how much to update the value function based on a new observation. A common choice is a generalized harmonic stepsize rule [13, 39]:

$$\alpha_n = \frac{a}{a + n - 1} \tag{5.2}$$

where n is the iteration number and $a \in \mathbb{R}^+$ is a tunable parameter. This rule ensures that stepsizes are large at the beginning (allowing for rapid learning) and decrease over time (leading to stability and convergence). When the value function is approximated, an exponential smoothing update is often applied:

$$\hat{V}_t^n(y_t) = (1 - \alpha_n)\hat{V}_t^{n-1}(y_t) + \alpha_n \hat{v}_{t+1} \tag{5.3}$$

5.3 Value Function Approximation

5.3.1 Desirable Properties of an Approximation

The choice of the value function approximation architecture is arguably the most critical design decision in ADP. A good approximation must:

1. **Capture Structural Properties:** It should preserve key properties of the true value function. For many inventory problems, this includes **concavity** (reflecting diminishing marginal returns of inventory) and, ideally, **separability**.

2. **Enable Tractable Optimization:** The structure of the approximation should ensure that the maximization step within the ADP algorithm remains computationally feasible.

3. **Support Efficient Updates:** It must be possible to efficiently update the approximation based on new sample information.

5.3.2 Piecewise-Linear and Separable Concave Approximation

For the multi-location inventory problem, a highly effective choice is a **separable, piecewise-linear, concave** approximation. This structure is motivated by the properties of the underlying problem [13].

- **Concavity:** The value of an additional unit of inventory generally decreases as the inventory level increases, which is a hallmark of concavity.
- **Separability:** The total network value is approximated as the sum of value functions for each individual location. While the true value function is not perfectly separable due to transshipments, this assumption dramatically simplifies computation.

The approximation takes the form:

$$\hat{V}_t^z(y_t) = \sum_{i=1}^{L} \hat{V}_{it}^z(y_{it}) \tag{5.4}$$

where each component $\hat{V}_{it}^z(y_{it})$ is a piecewise-linear concave function. Each of these functions is defined by a set of **breakpoints** u_{it}^k where the slope can change, and the **slopes** v_{it}^k between those breakpoints. To maintain concavity, the slopes must be arranged in non-increasing order: $v_{it}^0 \geq v_{it}^1 \geq \cdots \geq v_{it}^{k_{max}}$.

5.4 The CAVE Algorithm and Gradient Calculation

5.4.1 Motivation for CAVE

A critical challenge in ADP is that iterative updates can easily destroy the structural properties of the value function approximation. If we are using a concave approximation, a naive update might result in a non-concave function, which would invalidate the efficient solution methods that rely on this property. The **Concave Adaptive Value Estimation (CAVE)** algorithm is a specialized updating procedure designed specifically to preserve concavity in piecewise-linear approximations while incorporating new information [39].

The CAVE algorithm works by updating the *slopes* of the piecewise-linear function rather than the values directly. It uses sample gradient information to perform these updates within carefully chosen "smoothing intervals" that guarantee the resulting function remains concave.

5.4.2 CAVE Algorithm Structure

The CAVE algorithm updates the piecewise-linear approximation while preserving concavity, as shown in Fig. 5.2:

Algorithm 2: CAVE update of the approximation.

Input : $L, \mathbf{x}_t, \delta^+, \delta^-, \alpha, \pi_t^+, \pi_t^-, K_t^n, v_t^n, u_t^n$;
Output: $K_t^{n+1}, v_t^{n+1}, u_t^{n+1}$;

begin

for $i = 1, \ldots, L$ **do**

$$k^- = \min\left\{k \in K_{it}^n : v_{it}^{k,n} \leq (1-\alpha)v_{it}^{k+1,n} + \alpha\pi_{it}^-\right\},$$

$$k^+ = \max\left\{k \in K_{it}^n : (1-\alpha)v_{it}^{k-1,n} + \alpha\pi_{it}^+ \leq v_{it}^{k,n}\right\},$$

Define the smoothing interval:

$$\Delta = \left[\min\{x_{it} - \delta^-, u_{it}^{k^-,n}\}, \max\{x_{it} + \delta^+, u_{it}^{k^++1,n}\}\right]$$

Create new breakpoints at x_{it} and the end points of Δ; add them to u_t^{n+1};

Add according indexes to K_{it}^{n+1};

For each segment in the interval update the slope:

$$v_{it}^{k,n+1} = \alpha\pi_{it} + (1-\alpha)v_{it}^{k,n}, \quad i \in \{1, \ldots, L\},$$

where

$$\pi_{it} = \begin{cases} \pi_{it}^- & \text{if } u_{it}^{k,n} < x_{it}, \\ \pi_{it}^+ & \text{otherwise.} \end{cases}$$

Figure 5.2: CAVE update of the approximation. Algorithm taken from [13].

5.4.3 A Detailed Look at Gradient Calculation

The core of the CAVE update is the calculation of a **sample gradient**. In this context, the gradient represents the marginal value of an additional unit of inventory. Because the value function is piecewise-linear and non-differentiable at the breakpoints, we use the concepts of **left and right gradients**. For a given period t, the sample gradient is a vector composed of gradients for each location, and each location's gradient itself has two distinct components:

1. The gradient of the **immediate reward function**.

2. The gradient of the **future value function** (i.e., the value of the next state, V_{t+1}).

1. Gradient of the Immediate Reward The first component is derived from the immediate reward obtained in period t. The right gradient, π_i^+, measures the change in reward from having one *more* unit of inventory, while the left gradient, π_i^-, measures the change from having one *less*. Based on the reward function $R_t(y_t, d_t) = \sum_i (p_i \min(y_{it}, d_{it}) - h_i(y_{it} - d_{it})^+)$, these are calculated as follows [13]:

- **Right Gradient:** If the inventory stocked out ($y_{it} < d_{it}$), one additional unit will be sold, yielding the sales price p_i. If you already have sufficient stock ($y_{it} \geq d_{it}$), the extra unit will be held over, incurring a holding cost, so the marginal value is $-h_i$.

$$\pi^+_{i,\text{reward}}(y_t, d_t) = \begin{cases} p_i & \text{if } y_{it} < d_{it} \\ -h_i & \text{otherwise} \end{cases} \tag{5.5}$$

- **Left Gradient:** If the inventory stock is at or below demand ($y_{it} \leq d_{it}$), having one less unit means one less sale, for a marginal value of $-p_i$. If there is excess stock ($y_{it} > d_{it}$), one less unit means you avoid a holding cost, for a marginal value of h_i. The paper uses a slightly different derivation resulting in $-h_i$, which we will follow.

$$\pi^-_{i,\text{reward}}(y_t, d_t) = \begin{cases} p_i & \text{if } y_{it} \leq d_{it} \\ -h_i & \text{otherwise} \end{cases} \tag{5.6}$$

2. Gradient of the Future Value (Shadow Prices) The second, and more complex, component of the gradient comes from the value of being in the next state, x_{t+1}. This value is determined by solving the optimization problem for period $t+1$. The marginal value of having an extra unit of inventory at the start of period $t+1$ is precisely the **shadow price** (or dual variable) of the corresponding inventory availability constraint in the linear program.

The Challenge of Degeneracy A significant complication arises here: simply reading the dual variables from a standard LP solver can be misleading in the presence of *degeneracy*. In network flow terms, degeneracy typically appears when an arc that belongs to the optimal basis has its flow pinned at a lower or upper bound (for example, zero or the segment capacity). Under degeneracy a tiny perturbation to a right-hand side (for instance, adding one unit to a location's initial inventory) may change the optimal basis structure; consequently, the duals returned by the solver for the current basis do not necessarily equal the true

directional derivatives (the actual marginal change in the objective when the right-hand side is increased or decreased). For this reason, the CAVE gradient machinery must use augmentation/shortest-path methods (or other basis-robust techniques) to obtain the correct *right* and *left* shadow prices used in slope updates. [13]

An illustrative degeneracy example

To make this concrete, I explain the three-location degeneracy example from Meissner and Senicheva. The network has three supply nodes (locations) and a sink node that aggregates total network inventory; each supply node has an initial inventory (shown next to the node). Arcs are subject to lower and upper flow bounds and each arc-segment carries an associated cost. The sink collects the total inventory: the total outflow from the sink equals the sum of the three initial inventories.

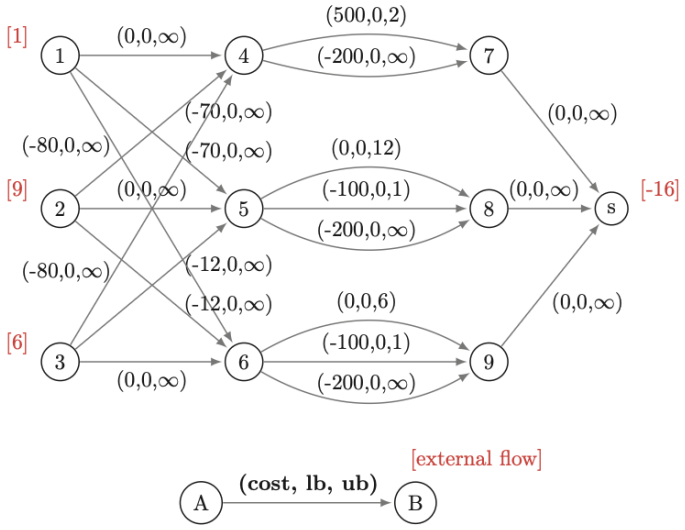


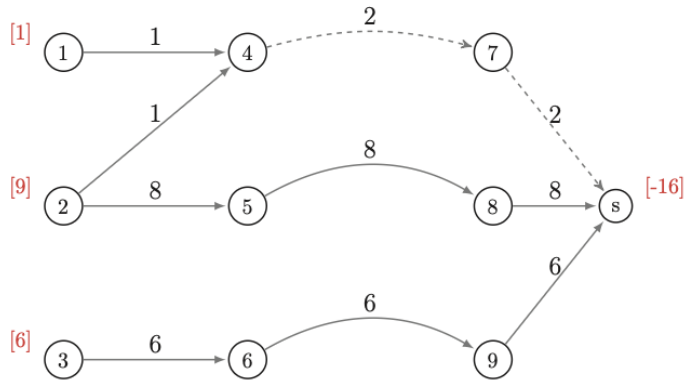
Figure 5.3: Three-location transshipment network used to demonstrate degeneracy. Red numbers adjacent to supply nodes indicate initial inventory levels. (Figure adapted from [13])

The network solver returns an optimal flow pattern as shown in Figure 5.4. For this solution the dual variables associated with the supply-node flow-balance constraints are

$$\text{dual values for nodes } 1,2,3 \;=\; 70,\; 0,\; 0$$

respectively. Focus on location 3. In the optimal solution the unique path from location 3 to the sink includes an arc that is *degenerated* because its flow equals the upper bound (the arc is saturated). Under this condition, a one-unit increase

in the initial inventory of location 3 would force a change in the optimal basis: the solver's current basis is not robust to this perturbation.



(a) For the initial problem.

Figure 5.4: Optimal flows reported by the network solver (original problem). Duals at supply nodes are (70,0,0). (Adapted from [13].)

When we form the *augmented* problem—i.e., increase the initial inventory at location 3 by one unit—and re-solve the optimization, the optimal basis changes and the new solution is shown in Figure 5.5. Comparing objective values of the original and augmented problems yields the actual (directional) positive shadow price for location 3:

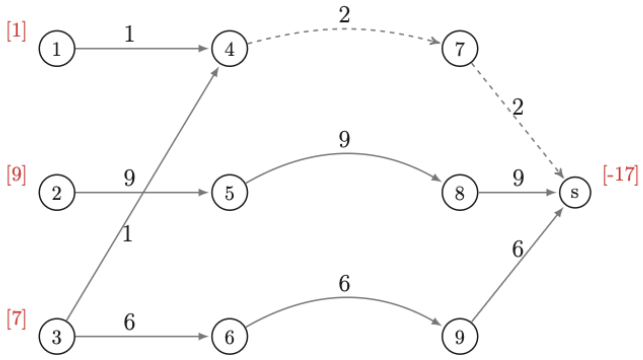
$$\pi_3^+ \;=\; V_{\text{augmented}} - V_{\text{original}} \;=\; -10,$$

which is *different* from the dual variable reported for node 3 in the original basis (which was 0). In contrast, the negative shadow price for location 3 is

$$\pi_3^- \;=\; 0,$$

and the shadow prices for locations 1 and 2 coincide with their reported dual variables. In words: the right-directional derivative (increase by +1) for location 3 equals −10, while the left-directional derivative (decrease by -1) equals 0; the solver dual for node 3 (equal to 0) therefore does not represent the correct positive directional marginal.

Solution: Shortest Path on an Augmenting Network To overcome the degeneracy issue, a more robust method is used to find the true shadow prices. As proposed by Powell (1989) and applied by Meissner and Senicheva (2018), this involves finding the **least-cost, flow-augmenting path** from each location's node to the sink node in a residual network. This calculation is equivalent to running a



(b) For the augmented problem.

Figure 5.5: Optimal flows after increasing the initial inventory of location 3 by one unit (augmented problem). The change in optimal basis produces a directional objective change corresponding to $\pi_3^+ = -10$. (Adapted from [13].)

shortest path algorithm (like Bellman-Ford or SPFA for networks with potential negative costs) on the augmenting network. This procedure correctly identifies the true marginal cost of increasing (for the positive shadow price) or decreasing (for the negative shadow price) the inventory at each location. It is far more efficient than re-solving the LP multiple times and provides the correct gradient information needed for a robust CAVE update.

5.5 Network Flow Formulation

5.5.1 Problem Reformulation

The separable, piecewise-linear approximation enables reformulation of the optimization problem as a minimum-cost flow problem. This transformation provides significant computational advantages by leveraging the structure of the value function approximation.

5.5.2 Network Structure

The network representation includes:

- **Nodes**: Representing locations at different time periods
- **Arcs**: Representing inventory flows and transshipment decisions
- **Capacity Constraints**: Reflecting physical and operational limits
- **Cost Structure**: Incorporating both immediate costs and future value estimates

Figure 5.6 illustrates the network representation where nodes are locations at different points of period t, and arrows indicate the flow of inventory.

5.5.3 Flow Formulation

Using the notation for the problem in terms of the post-decision state and introducing additional variables $g_{it}^k \in \mathcal{K}_i$, the maximization problem for each time t from the optimality equations can be stated as:

$$\max_{z_{ijt}} \left(-\sum_{i\in\mathcal{L}} \sum_{j\in\mathcal{L}} c_{ij} p_{ij} z_{ijt} + \sum_{i\in\mathcal{L}} \sum_{k\in K_{it}} v_{it}^k g_{it}^k \right) \tag{5.7}$$

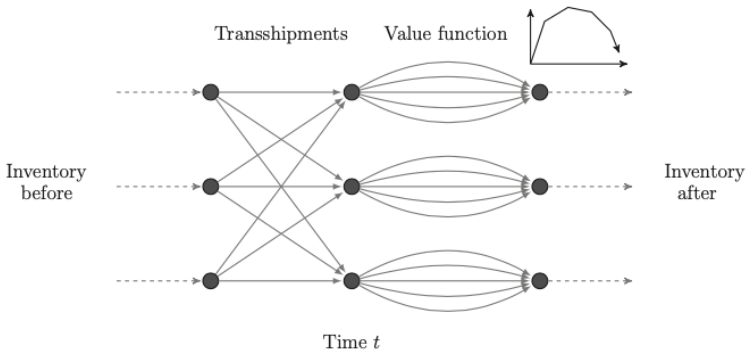


Figure 5.6: Network representation. Figure taken from [13].

Subject to:

$$\sum_{j\in\mathcal{L}} z_{ijt} = (y_{it-1} - d_{it-1})^+ \quad \forall i \in \mathcal{L} \tag{5.8}$$

$$\sum_{k\in K_{it}} g_{it}^k = \sum_{j\in\mathcal{L}} z_{ijt} \quad \forall i \in \mathcal{L} \tag{5.9}$$

$$0 \leq g_{it}^k \leq u_{it}^k \quad \forall i \in \mathcal{L} \tag{5.10}$$

$$0 \leq g_{it}^1 \leq u_{it}^2 - u_{it}^1 \quad \forall i \in \mathcal{L} \tag{5.11}$$

$$\vdots$$

$$0 \leq g_{it}^{k_{max}-1} \leq u_{it}^{k_{max}} - u_{it}^{k_{max}-1} \quad \forall i \in \mathcal{L} \tag{5.12}$$

$$g_{it}^{k_{max}} \geq 0 \quad \forall i \in \mathcal{L} \tag{5.13}$$

$$z_{ijt} \in \mathbb{Z}_{\geq 0} \quad \forall i, j \in \mathcal{L} \tag{5.14}$$

where $g_{it}^k \in [0, u_{it}^{k+1} - u_{it}^k]$.

5.5.4 Solution Methods

By applying the separable, piecewise-linear approximation, the problem can be reformulated and the network structure recognized. As the sub-problem has a linear objective function and linear constraints, it is a linear programming problem. Therefore, any linear solver can find optimal solutions in a reasonable time. Moreover, the solutions are naturally integers without applying any integer programming techniques.

The network flow formulation can be solved efficiently using Commercial LP Solvers.

5.6 Computational Considerations

5.6.1 Algorithmic Complexity

The computational complexity of ADP depends on several factors:

- **State Space Dimensionality**: Higher dimensions require more sophisticated approximations
- **Approximation Complexity**: Number of parameters in the value function approximation
- **Simulation Length**: Number of sample paths and time periods

- **Update Frequency**: How often the approximation is updated

For the network flow reformulation, the complexity is dominated by the linear programming solver, which typically runs in polynomial time for the structured problems that arise in practice.

5.6.2 Convergence Properties

Forward ADP with appropriate stepsizes and exploration parameters exhibits:

- **Almost Sure Convergence**: Under standard conditions, the algorithm converges to optimal or near-optimal policies
- **Rate of Convergence**: Depends on problem structure, approximation quality, and algorithm parameters
- **Finite Sample Performance**: Good performance can often be achieved with moderate sample sizes

The CAVE algorithm specifically ensures that the concavity property is preserved throughout the learning process, which is crucial for maintaining the structural properties that enable efficient optimization.

5.6.3 Practical Implementation

Key implementation considerations include:

- **Memory Management**: Efficiently storing and accessing breakpoints and slopes
- **Numerical Stability**: Avoiding numerical issues in slope calculations and updates
- **Parameter Tuning**: Selecting appropriate stepsizes, exploration parameters, and smoothing intervals
- **Integration with Solvers**: Interfacing effectively with linear programming and network flow solvers

The experimental parameters used in practice include setting $b = 0.7$ and $a = 5$ for the stepsize calculation, providing a balance between learning speed and stability. To evaluate each policy results, I have used monte carlo simulation to evaluate the policy using N=1000 (see Figure. 5.7), and update the algorithm presented in [13].

Algorithm 3: Evaluation of a policy.

Input : L, T, c, N, policy π,
S_i, h_i, p_i, ρ_{ij} for $i, j \in \mathcal{L}$;
Output: Average profit after N iterations;

begin
  for $n = 1, \ldots, N$ **do**
    initialize $x_{i0} = S_i$ for $\forall\, i \in \mathcal{L}$;
    for $t = 0, \ldots, T-1$ **do**
      calculate transshipments between location $\mathbf{z}_t^\pi$;
      get transshipment cost $C_t(\mathbf{x}_t, \mathbf{z}_t^\pi)$;
      generate random demand vector $\mathbf{d}_t$;
      get reward $R_t(\mathbf{x}_t, \mathbf{d}_t, \mathbf{z}_t^\pi)$;
      calculate next state $\mathbf{x}_{t+1}$;
      get profit $P_t = \big(R_t(\mathbf{x}_t, \mathbf{d}_t, \mathbf{z}_t^\pi) - C_t(\mathbf{x}_t, \mathbf{z}_t^\pi)\big)$;
    get total profit $P = \sum_{t=0}^{T-1} P_t$
  get average profit P/N

Figure 5.7: Monte carlo evaluation algorithm. Algorithm taken from [13].

5.7 Extension to a Nonlinear Transshipment Cost Function

5.7.1 Modeling Economies of Scale with a Concave Cost Function

The initial model assumes a linear transshipment cost, where the cost per unit is constant regardless of the shipment size. This is a simplification. In reality, transportation logistics exhibit significant economies of scale—the per-unit cost decreases as the shipment volume increases.

To create a more realistic model, I introduce a piecewise-linear concave transshipment cost function. This function better reflects real-world cost structures. (example: Fig. 5.8) This cost function is defined by two key features:

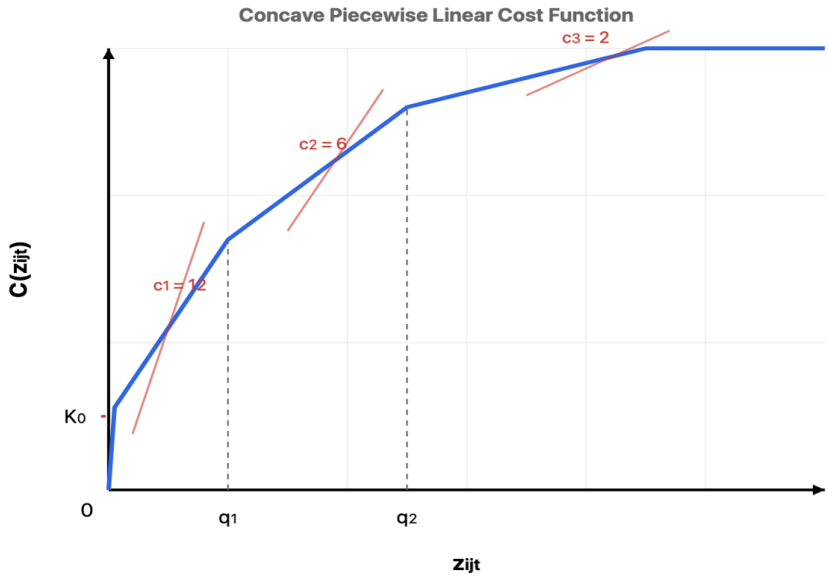


Figure 5.8: Concave transhipment cost structure.

- **Fixed Cost (K_0)**: A fixed charge is incurred if any quantity greater than zero is shipped. This represents the cost of dispatching a vehicle, independent of the volume.
- **Decreasing Marginal Costs**: The cost is broken into segments, where the marginal (per-unit) cost is highest for the initial units and decreases for larger quantities. This models volume discounts, where filling a truck is progressively

cheaper on a per-unit basis. The marginal costs for successive segments are ordered such that $c_1 > c_2 > \cdots > c_m$.

The introduction of this non-convex cost function is a critical extension, but it fundamentally changes the nature of the optimization problem. The problem can no longer be solved as a simple minimum-cost network flow problem, as the convexity assumption required by linear programming is violated.

5.7.2 Reformulation as a Mixed-Integer Linear Program (MILP)

To accommodate the concave cost function, the optimization subproblem solved at each step of the ADP algorithm must be reformulated as a Mixed-Integer Linear Program (MILP). MILPs are a class of optimization problems that include both continuous variables and integer variables (in this case, binary variables), making them substantially more complex to solve than standard LPs.

The objective function is now to minimize the total piecewise-linear transshipment cost. This is achieved by modeling the flow on each transshipment arc (i, j) as a sum of flows across different cost segments, s_{ijk}.

$$\text{Minimize Cost} = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} \sum_{k=0}^{K-1} (m_k \cdot \rho_{ij}) \cdot s_{ijk} \tag{5.15}$$

This minimization is subject to a new set of constraints for each transshipment arc:

1. **Total Flow Constraint**: The sum of the flows across all segments must equal the total quantity transshipped, z_{ij}.

$$z_{ij} = \sum_{k=0}^{K-1} s_{ijk} \tag{5.16}$$

2. **Segment Capacity Constraint**: To model the piecewise structure, I introduce a binary variable y_{ijk} that equals 1 if cost segment k is used for the flow on arc (i, j), and 0 otherwise. The flow within a segment cannot exceed its width, W_k.

$$s_{ijk} \leq W_k \cdot y_{ijk} \tag{5.17}$$

3. **Segment Ordering Constraint**: To enforce the decreasing marginal costs, we must ensure that cheaper, higher-volume segments are only used after the more expensive, lower-volume segments have been filled to capacity.

$$y_{ij,k+1} \leq y_{ijk} \tag{5.18}$$

This constraint ensures that if segment $k+1$ is used (i.e., $y_{ij,k+1} = 1$), then segment k must also be used ($y_{ijk} = 1$).

This MILP formulation accurately captures the economies of scale in transshipment but significantly increases the computational effort required to find the optimal decision at each step of the ADP simulation.

5.7.3 Updating Shadow Price Calculations for the MILP

The transition from a standard Linear Program (LP) to a **Mixed-Integer Linear Program (MILP)** introduces a critical challenge: calculating the shadow prices for the CAVE value function update. For LPs, strong duality guarantees a dual variable for each constraint, representing the marginal value of inventory. However, for MILPs, the presence of integer variables breaks this property, making the direct extraction of shadow prices impossible.

To overcome this, several heuristic methods can be implemented to approximate these values. While they vary in complexity and accuracy, they all aim to estimate the marginal value of an additional unit of inventory. The implementation includes three such methods, with **Local Approximation** chosen as the default to minimize computational load during the intensive learning phase.

Method 1: Local Approximation (Default)

The **Local Approximation** method provides a computationally efficient estimate by linearizing the problem around the optimal MILP solution, z^*_{ij}.

1. **Solve the MILP**: The optimal integer transshipment quantities, z^*_{ij}, are found.
2. **Identify Active Cost Segment**: For each arc (i, j), the algorithm identifies the marginal cost, m_k, of the specific segment that the optimal flow z^*_{ij} falls into.
3. **Construct Linearized Residual Graph**: A simple residual graph is built where each arc's weight is fixed to the identified local marginal cost $m_k \cdot c \cdot \rho_{ij}$.
4. **Calculate Shadow Prices**: A standard shortest path algorithm (e.g., Bellman-Ford) is run on this simplified graph to find the shadow prices π_i^+ and π_i^-.

This method is fast because it reduces the complex, piecewise problem to a standard shortest path problem on a simple graph.

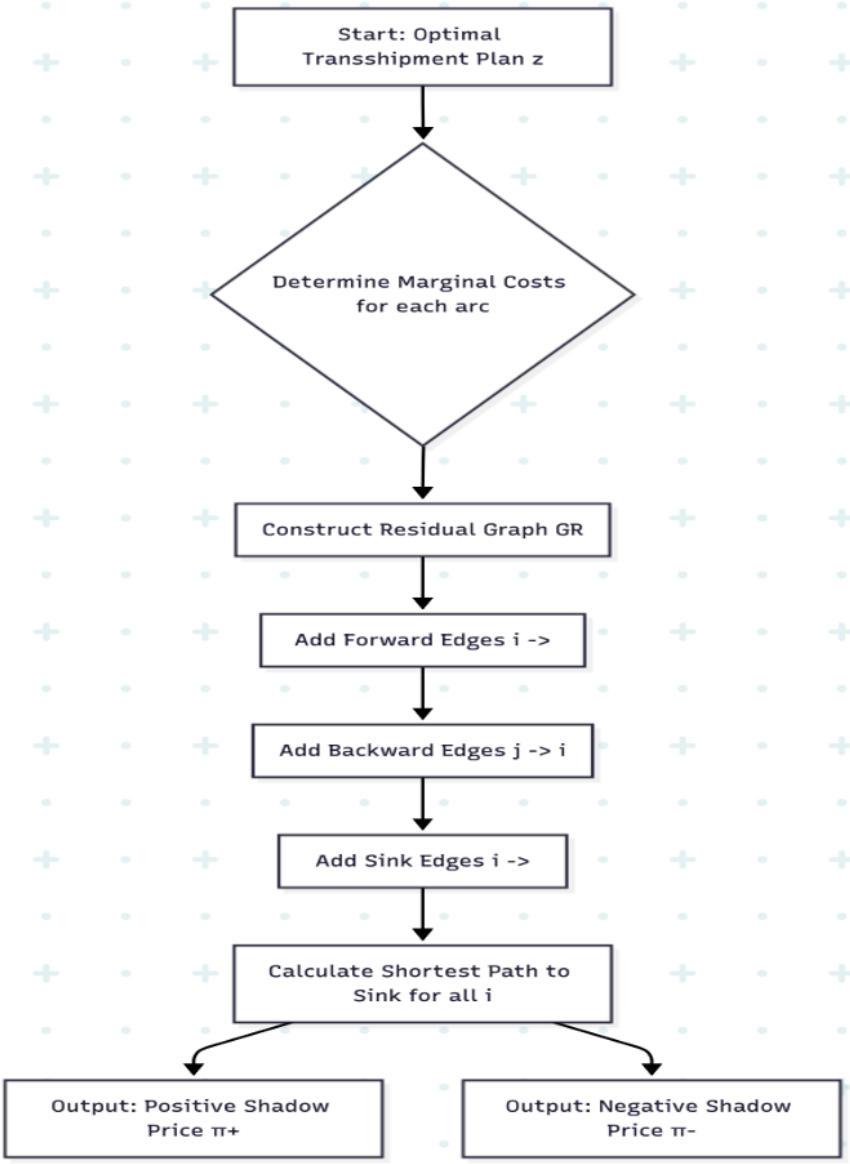


Figure 5.9: Shadow price calculation flow for ADP with a concave transshipment cost function. The default path uses the Local Approximation heuristic for computational efficiency.

Method 2: Expanded Network Linearization

As implemented in the `compute_shadow_prices_piecewise_linearized` function, this method offers a more detailed and potentially more accurate approximation at a higher computational cost.

Instead of using a single marginal cost per arc, it constructs an **expanded residual network**. For each transshipment arc (i, j), it creates a chain of nodes and edges that explicitly models *every segment* of the piecewise cost function. An arc from origin i to destination j is replaced by a path like:

$$i \to (i, j, seg_0^{in}) \to (i, j, seg_0^{out}) \to (i, j, seg_1^{in}) \to \cdots \to j$$

The edge from (i, j, seg_k^{in}) to (i, j, seg_k^{out}) is assigned a weight equal to the marginal cost m_k of that segment. Reverse arcs with weight $-m_k$ are added for segments that currently have flow, representing the savings from rerouting. By running a shortest path algorithm on this expanded graph, an additional unit of flow can intelligently choose the cheapest available segment, providing a more globally aware shadow price. However, the size of this graph grows significantly with the number of locations and cost segments, making it computationally much slower than the default local approximation.

Method 3: Discrete Enumeration

The `compute_shadow_prices_discrete_enumeration` function implements a heuristic based on local search. This approach attempts to find a better integer solution in the immediate neighborhood of the one found by the MILP solver.

1. **Generate Nearby Integer Solutions**: It creates a set of candidate solutions by rounding, flooring, and ceiling the values of the current solution z_{ij}^*.
2. **Evaluate Candidates**: Each candidate solution is checked for feasibility (i.e., it doesn't violate inventory constraints). The total transshipment cost is calculated for each feasible candidate using the full piecewise cost function.
3. **Select Best Candidate**: The feasible candidate solution with the absolute lowest total cost is identified.
4. **Approximate from Best Candidate**: Crucially, this method then uses the **Local Approximation** technique (Method 1) on this new, best-found integer solution to calculate the final shadow prices.

The intuition is that the shadow prices derived from a locally-optimal integer point may be more stable and representative than those from the initial MILP solution.

This method is computationally very expensive as it requires evaluating multiple potential solutions and is therefore impractical for the main learning loop.

While the Expanded Network and Discrete Enumeration methods offer alternative, potentially more accurate estimations, they come with a significant computational burden. The ADP framework requires thousands of iterations to learn the value function effectively. Therefore, the **Local Approximation** method is chosen as the default strategy. It strikes the most effective balance between providing a reasonable estimate of marginal inventory value and the computational speed required to make the learning process tractable.

Chapter 6

Numerical Experiments

In this chapter, I analyze the performance of the transshipment policies presented in this thesis. The first part presents results for small networks with limited time horizons, allowing comparison of heuristic methods against the optimal transshipment policy derived from dynamic programming. The second part focuses on larger networks representing real-world problems. All algorithms were implemented in Python, and experiments were conducted on a MacBook Pro with a 2.5 GHz Intel Core i7 processor and 16 GB RAM. Linear programs were solved using the open-source PuLP linear programming solver.

The term **ADP policy** refers to the proactive transshipment policy produced by the ADP algorithm described in Chapter 5, while **DP policy** denotes the optimal transshipment policy found by solving the DP equations formulated in Chapter 4.

6.1 Two-Location Networks

First, I start the experiment by considering a linear transhipment cost function and get similar results to what presented in the reference paper [13].

6.1.1 Linear transhipment cost function

Considering two non-identical locations that receive inventory from a supplier at the beginning of the first period and use this inventory to satisfy customer demands over four subsequent periods without additional shipments from the supplier. The initial order-up-to level is fixed exogenously for each location. Following [35], I set the order-up-to level for each location approximately one standard deviation above the mean demand scaled for the entire time horizon:

$$S_i = \left\lfloor \mu_i T + \sigma_i \sqrt{T} \right\rfloor \tag{6.1}$$

where $\lfloor y \rfloor$ denotes the largest integer not exceeding y.

Table 6.1: Parameters for two-location experiments

Parameter	**Values**
Number of locations, L	2
Number of periods, T	4
Holding cost per unit, h_i [\$]	{8, 12, 20}
Sales price per unit, p_i [\$]	{40, 80, 100}
Transshipment cost per unit, c_i [\$]	1
Distance between locations, ρ_{ij}	{29, 61}
Demand distribution	{Unif(a_i, b_i), Pois(λ_i), NegBin(r_i, p_i)}
Mean demand, μ_i	{0.5, 1, 1.5}

Table 6.2: Parameters of distributions for two-location experiments

	Discrete Uniform (Low Variance)	**Poisson (Medium Variance)**	**Negative Binomial (High Variance)**
Notation	Unif(a_i, b_i)	Pois(λ_i)	NegBin(r_i, p_i)
Parameters	$a_i = 0$ $b_i \in \{1, 2, 3\}$	$\lambda_i \in \{0.5, 1, 1.5\}$	$p_i = 0.8$ $r_i \in \{2, 4, 6\}$

I conducted a factorial study encompassing most of the parameter combinations, totalling 54 scenarios. For each scenario, I have performed six simulation runs (one per policy) with 1000 independent replications per simulation. All simulation runs within a scenario used common random numbers.

I compared six transshipment policies:

- Optimal transshipment policy (DP)
- No-transshipment policy (NT)
- Reactive transshipment from closest location in case of stockout (RC)
- Proactive transshipment based on inventory equalization (TIE)
- Lookahead policy (PL)

- ADP policy with 1000 iterations for value function approximation

Comparison between ADP and DP policies in 99% of scenarios giving very similar results, substantially outperforming other policies. Other measures, including average transshipment size and frequency, further demonstrated the similarity between ADP and DP policies.

Table 6.3: Table presents the difference between optimal policy and the policy in examination for identical demand distributions at both locations ($p_1 = 40$, $p_2 = 80$, $h_1 = 8$, $h_2 = 12$)

Distribution	**NT**	**RC**	**TIE**	**PL**	**ADP**
Uniform(0,1)	0.45	5.75	6.72	0	0.45
Uniform(0,2)	1.745	6.62	8.86	1.375	0.755
Uniform(0,3)	2.105	6.525	21.98	16.12	1.3
Pois(0.5)	0.65	9.59	5.38	-0.11	0.65
Pois(1)	1.375	4.01	11.59	2.085	0.86
Pois(1.5)	2.58	3.08	23.745	4.73	2.58
NegBin(2,0.8)	0.55	13.66	7.01	1.905	0.55
NegBin(4,0.8)	1.86	3.945	7.595	2.875	0.87
NegBin(6,0.8)	1.895	2.255	12.29	40.92	0.985

Overall policy performance compared to the optimal DP solution is presented in Table 6.3. I evaluate results using the absolute difference from optimal solution rather than percentage optimality gap, as network profit contains both positive and negative components that preclude percentage calculation.

Using a box plot, we can understand the variation from optimality for the three demand distributions. Results are reported in Fig. 6.1, Fig. 6.2, and Fig. 6.3. From these results, we can conclude that overall, across all cases, ADP behaves the closest to optimality despite the variation in demand structure.

The results support the intuition that transshipment policies outperform no-transshipment policies significantly for items with higher demand variability. Increased demand variability also correlates with decreased performance of other policies, likely due to unnecessary stock movements in anticipation of future stock-outs that reduce profitability. Nevertheless, the ADP policy demonstrates consistent performance across all conditions.

The results indicate that PL and TIE perform better when demand distributions are less spread out. Notably, TIE does not directly incorporate cost into decision-making, considering cost only when determining redistribution locations rather

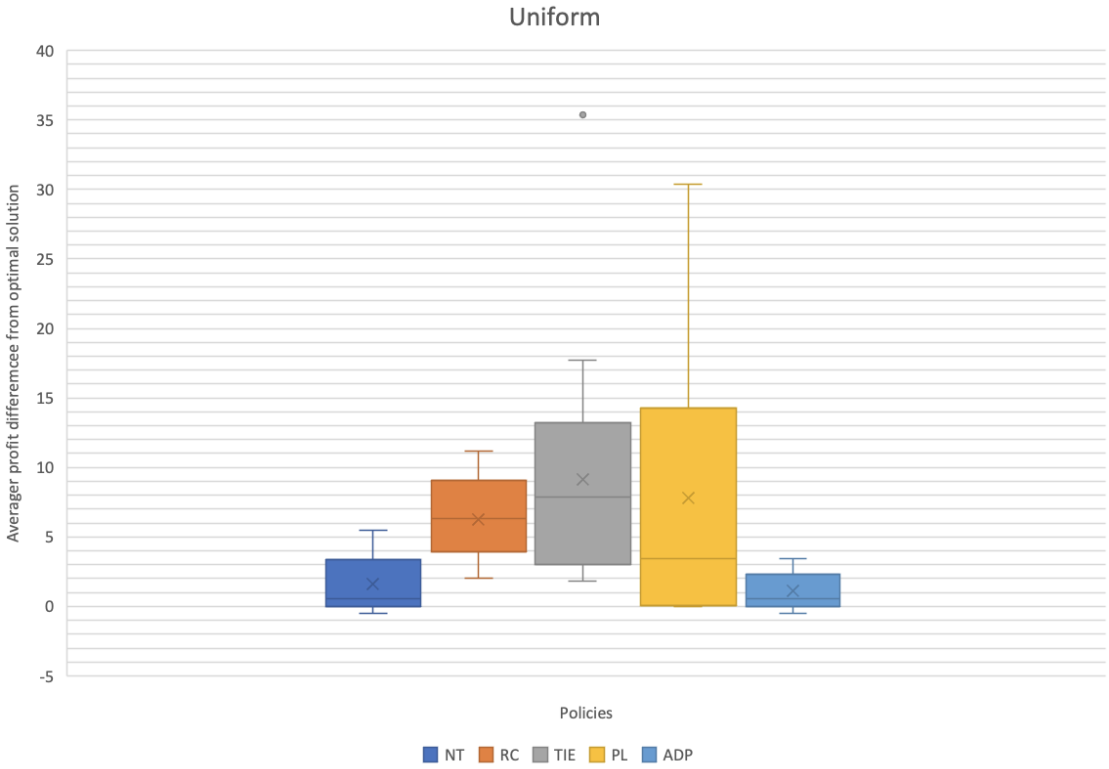


Figure 6.1: Results for two locations with uniform demand

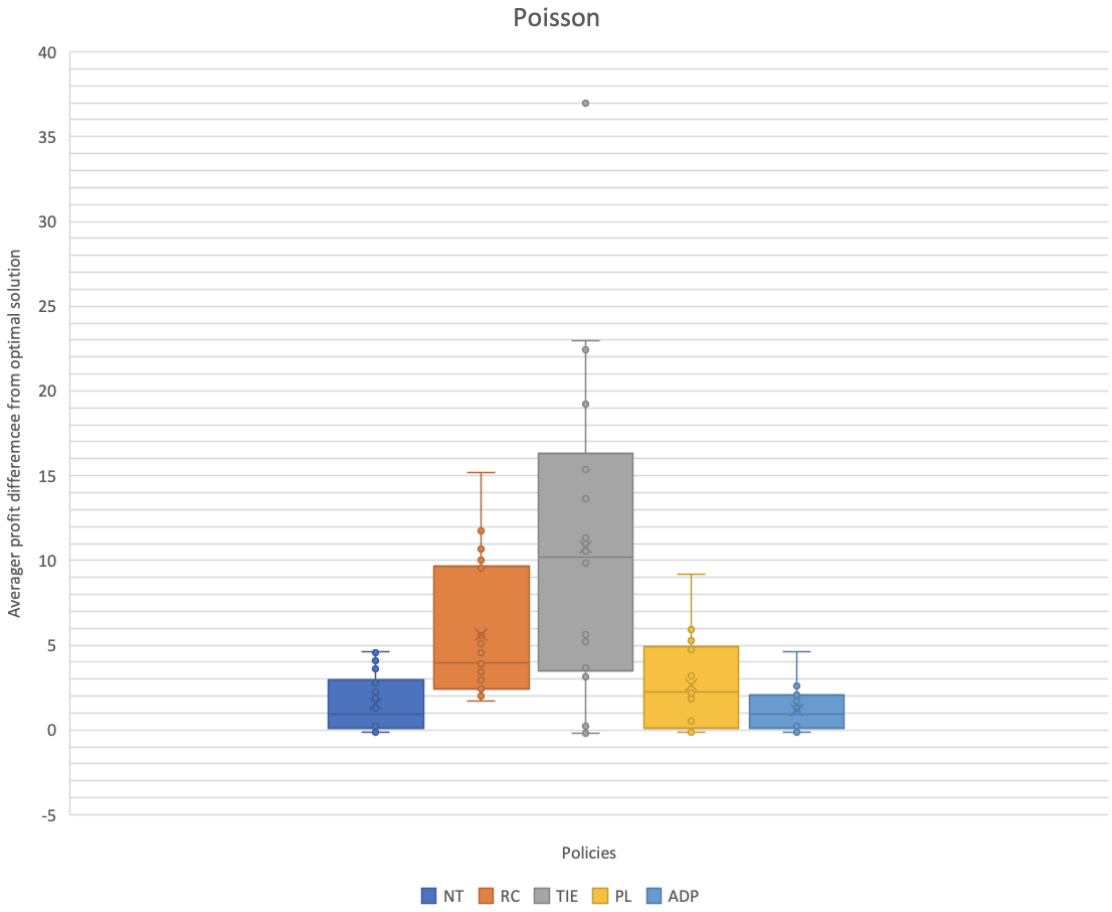


Figure 6.2: Results for two locations with Poisson demand

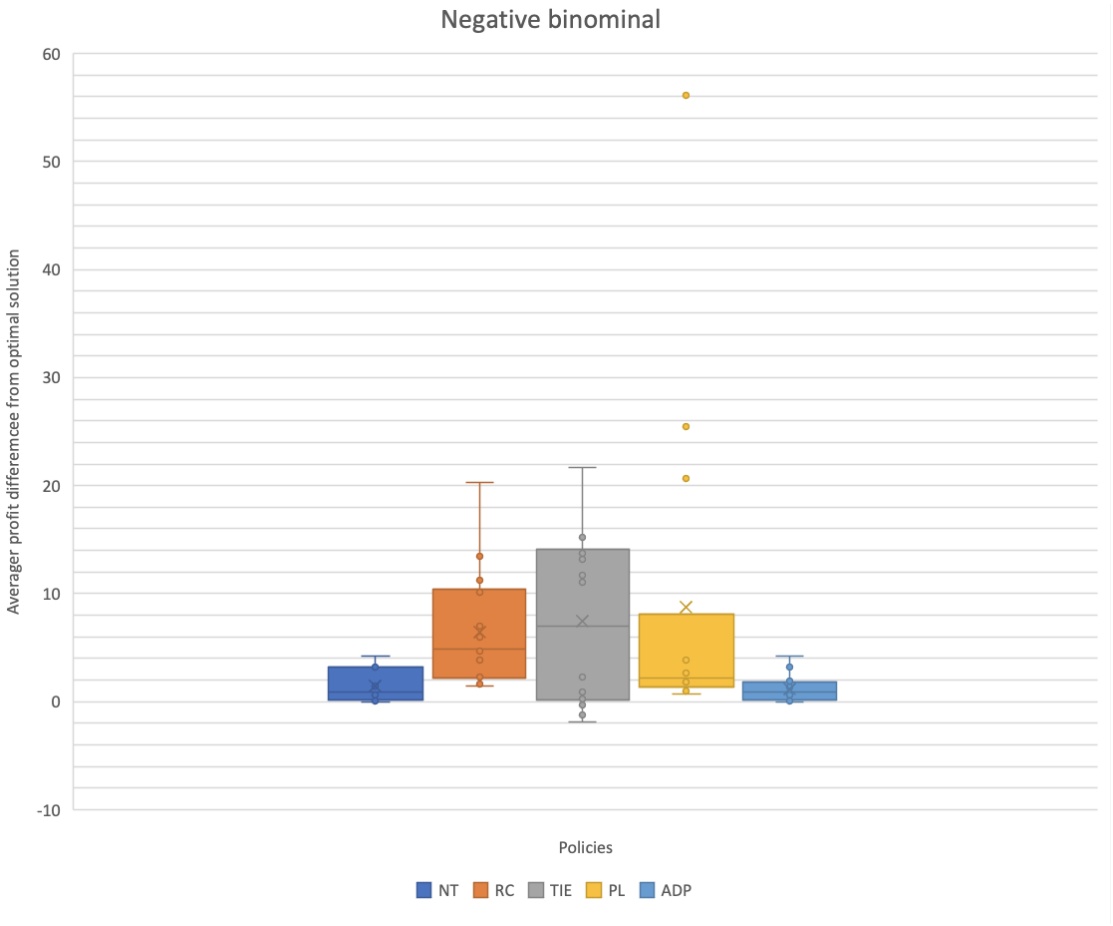


Figure 6.3: Results for two locations with negative binomial demand

than whether to redistribute at all.

6.1.2 Experiments with Concave Transshipment Cost Function

To evaluate the impact of economies of scale in transportation costs, we extended our experiments to incorporate a piecewise-linear concave transshipment cost function. This cost structure better reflects real-world logistics where per-unit costs decrease as shipment volume increases due to factors such as vehicle capacity utilization and volume discounts.

For these experiments, we used the following concave cost structure:

- Segment breakpoints: $U_k = [0, 2, 4, 8]$ units
- Marginal costs: $m_k = [0.5, 0.3, 0.2, 0.1]$ dollars per unit per distance

This structure represents significant economies of scale, with the marginal cost decreasing from \$0.5 for the first two units to \$0.1 for units beyond the eighth.

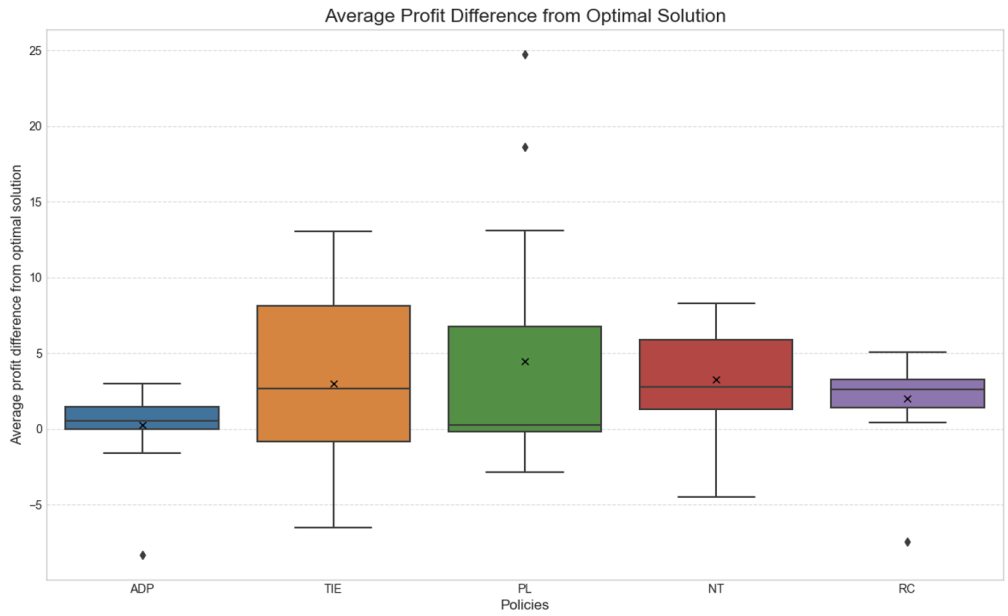


Figure 6.4: Results for two locations with uniform demand under concave transshipment cost

Figures 6.4, 6.5, and 6.6 present the results for two-location networks under the concave cost structure across different demand distributions. The findings reinforce

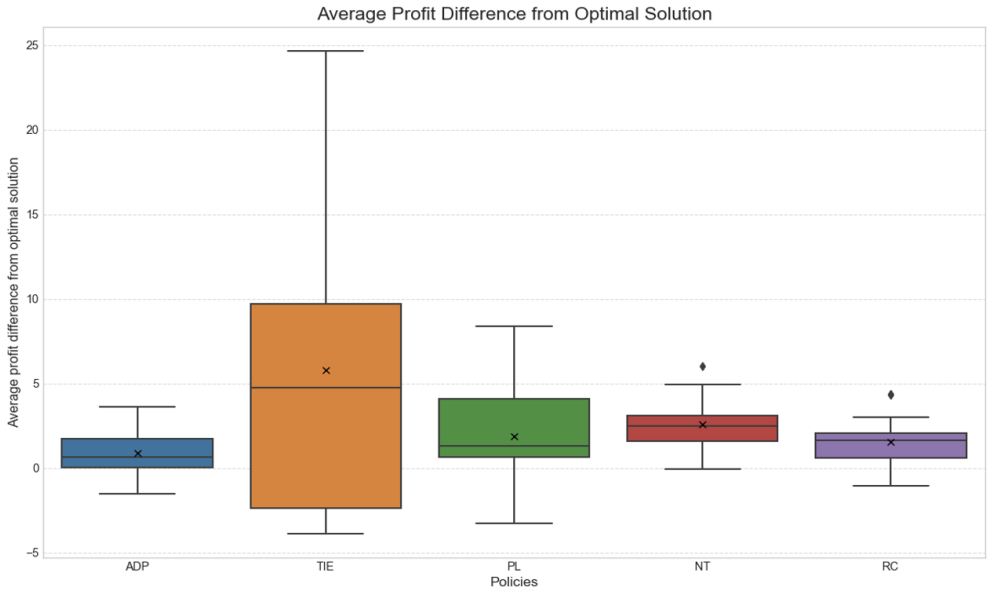


Figure 6.5: Results for two locations with Poisson demand under concave transshipment cost

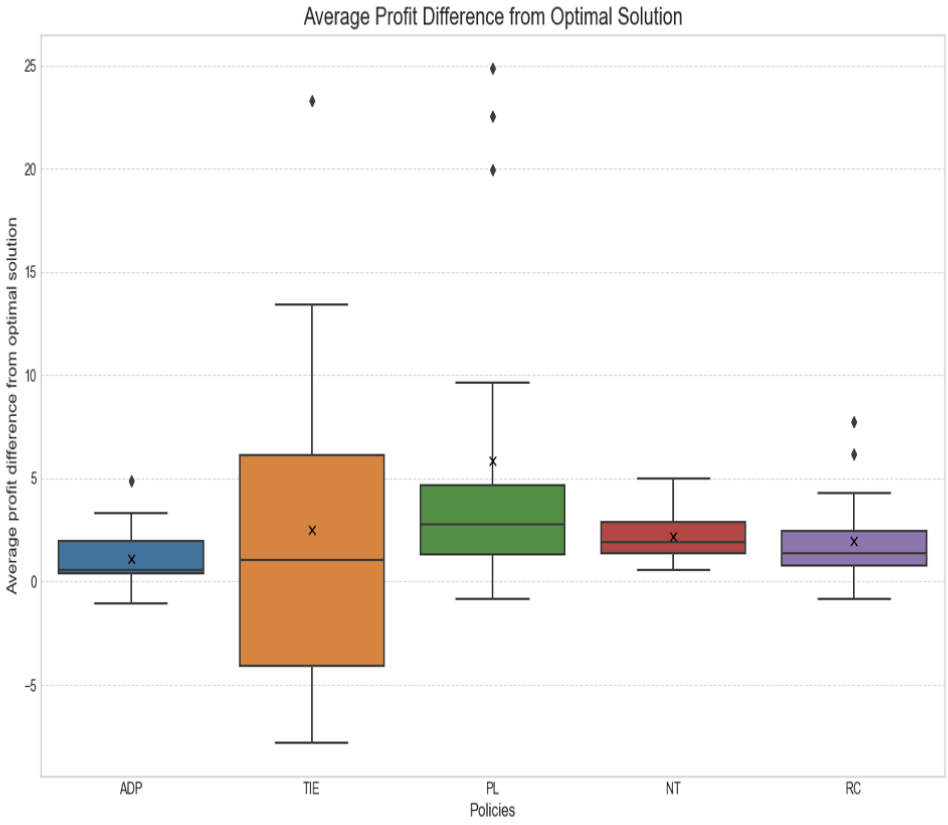


Figure 6.6: Results for two locations with negative binomial demand under concave transshipment cost

our previous conclusions: the ADP policy continues to deliver the best performance, while the value of the PL policy increases with higher demand variance. Notably, the no-transshipment (NT) policy becomes significantly less valuable under this cost structure, as the potential savings from economies of scale cannot be realized without transshipment activities.

6.1.3 Impact of Economies of Scale on Profitability

To quantify the benefits of the concave cost structure, we compared the profitability under the decreasing marginal cost model against a fixed cost model. Figures 6.7 and 6.8 illustrate the absolute and relative profit improvements, respectively, achieved by incorporating economies of scale into the transshipment cost structure.

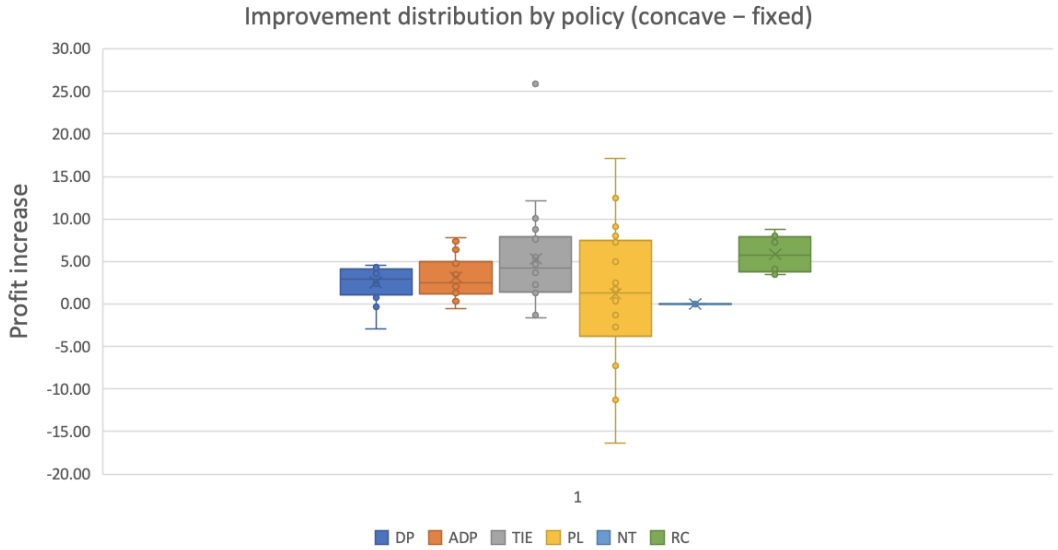


Figure 6.7: Profit increase when considering economies of scale (concave cost minus fixed cost)

The results demonstrate substantial improvements in network profitability when accounting for economies of scale in transportation. The DP and ADP policies show the greatest absolute profit increases, leveraging their sophisticated decision-making capabilities to optimize shipment quantities and fully exploit the concave cost structure. The reactive policy (RC) also benefits significantly, though to a lesser extent than the proactive policies.

The no-transshipment policy (NT) shows no profit improvement, as expected,

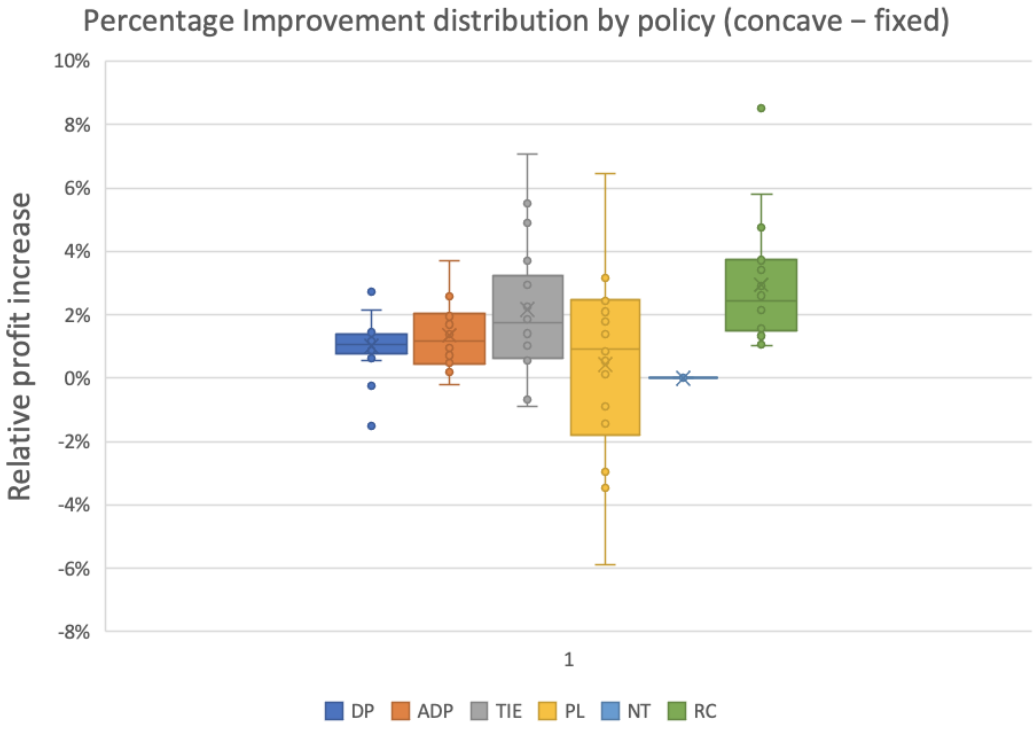


Figure 6.8: Relative profit improvement when considering economies of scale

since it cannot take advantage of the economies of scale without engaging in transshipment activities. This highlights the critical importance of transshipment decisions in environments where transportation costs exhibit significant scale economies.

The relative profit improvement analysis reveals that all transshipment policies achieve percentage gains, with the proactive policies (ADP, TIE, PL) showing the most consistent improvements across different scenarios. These results emphasise the value of incorporating realistic cost structures into transshipment optimization models, as they can significantly impact both the absolute profitability and the relative performance of different policies.

The concave cost structure particularly benefits scenarios with:

- Higher demand variance, where larger but less frequent shipments can be optimized
- Locations with complementary demand patterns, allowing consolidation of shipments
- Longer distances between locations, where transportation cost savings are more substantial

These findings have important practical implications for logistics and inventory management, suggesting that companies can achieve significant cost savings by properly modeling and exploiting economies of scale in their transportation operations.

6.2 Multiple-Location Networks

For larger networks, conducting full dynamic programming becomes computationally infeasible due to the curse of dimensionality. To address this limitation, I employ two experimental approaches: perfect foresight policy with known demand used as an upper bound, and stochastic demand environments with Poisson distributions to examine ADP , TIE, and PL policy.

6.2.1 Perfect Foresight as an Upper Bound

We utilize a perfect foresight approach that assumes knowledge of future demand in advance, formulated as a deterministic network flow problem:

$$\max_{z^d_{ijt}, z^e_{ijt}} \left(\sum_{t=0}^{T-1} \left(- \sum_{i \in \mathcal{L}} \sum_{j \in \mathcal{L}} c\rho_{ij}(z^d_{ijt} + z^e_{ijt}) + \sum_{j \in \mathcal{L}} p_j \sum_{i \in \mathcal{L}} z^d_{ijt} - \sum_{j \in \mathcal{L}} h_j \sum_{i \in \mathcal{L}} z^e_{ijt} \right) \right) \tag{6.2}$$

Subject to:

$$\sum_{j\in\mathcal{L}}(z^d_{ijt}+z^e_{ijt})=\sum_{j\in\mathcal{L}}z^e_{jit-1}\quad\forall i\in\mathcal{L},\forall t\in\mathcal{T}:t\geq 1 \tag{6.3}$$

$$\sum_{j\in\mathcal{L}}(z^d_{ijt}+z^e_{ijt})=S_i\quad\forall i\in\mathcal{L},t=0 \tag{6.4}$$

$$\sum_{i\in\mathcal{L}}z^d_{jit}\leq d_{it}\quad\forall i\in\mathcal{L},\forall t\in\mathcal{T} \tag{6.5}$$

$$z^d_{ijt},z^e_{ijt}\in\mathbb{Z}_{\geq 0}\quad\forall i,j\in\mathcal{L},\forall t\in\mathcal{T} \tag{6.6}$$

where z^d_{ijt} and z^e_{ijt} represent inventories transshipped from location i to j for sale or storage, respectively. This integer program has a min-cost flow structure, allowing us to obtain the optimal solution via LP relaxation. The solution provides an upper bound on network profit, against which we compare heuristic policies.

We conducted two experiment sets with the following fixed parameters:

- Time horizon: 28 days
- Holding cost: \$5 per item
- Sales price: \$80 per item
- Locations uniformly distributed on grid $[0, 100] \times [0, 100]$
- Euclidean distances between locations

Set 1: Varied transshipment costs $c \in \{0.1, 0.5, 1\}$ with initial inventory calculated using Equation (6.1).

Set 2: Fixed transshipment cost (\$0.5) with varied initial inventory levels $S \in \{697, 680, 697\}$, where 697 represents balanced inventory and 680 introduces imbalance.

6.2.2 Linear Transhipment Cost

For stochastic environments with known Poisson-distributed demand (arrival rate: 24 units/day), we compare policies against the no-transshipment baseline:

The results demonstrate that even simple reactive heuristics (RC) can be more profitable than no-transshipment policies (Table 6.6). However, the TIE policy, which doesn't consider costs, can lead to unprofitable decisions when transshipment costs are high, as shown in Table 6.6 for 10 locations. Also, even though it is the fastest and simplest transshipment heuristic, it performs worse than ADP and PL

Table 6.4: Percentage of upper bound for Set 1 with variable cost and multiple locations

	TIE			ADP			PL		
L	0.1$	0.5$	1$	0.1$	0.5$	1$	0.1$	0.5$	1$
5	98.9%	94.6%	89.1%	95.3%	94.1%	94.7%	97.3%	97.5%	97.8%
10	95.0%	75.0%	49.6%	95.3%	94.4%	95.2%	99.8%	99.0%	98.0%
15	99.2%	95.9%	91.8%	98.1%	97.6%	98.4%	99.0%	97.7%	96.2%
20	99.6%	97.9%	95.8%	100%	99.7%	99.7%	99.9%	99.7%	99.3%

Table 6.5: Percentage of upper bound for Set 2 (cost = 0.5$)

	TIE			ADP			PL		
L	697	680	$\overline{697}$	697	680	$\overline{697}$	697	680	$\overline{697}$
5	94.57%	98.68%	98.85%	94.12%	94.66%	98.46%	97.52%	97.02%	97.01%
10	75.01%	89.25%	74.05%	94.45%	93.65%	83.76%	98.98%	97.45%	96.90%
15	95.93%	98.22%	–	97.63%	94.51%	–	97.75%	96.00%	–

Table 6.6: Profit gain compared to no-transshipment policy (NT)

L	RC vs. NT	TIE vs. NT	PL vs. NT	ADP vs. NT
5	0.00	1162.19	1054.80	611.76
10	45.58	-554.61	3190.40	1698.11
15	33.97	1573.79	666.60	199.04

Table 6.7: Average runtime for different policies (seconds)

Locations (L)	RC	TIE	PL	ADP
5	4	126	663	4512
10	12	408	12509	8404
15	24	956	44182	14525

when there are high transshipment costs (Table 6.4) and high variability (Table 6.5). The PL policy shows competitive performance, sometimes outperforming ADP, particularly in larger networks.

The runtime analysis reveals significant computational differences between policies. While ADP provides excellent performance, its computational requirements grow substantially with network size. However, in the case of the PL policy, the runtime becomes a very significant issue in large networks, with 3× more runtime for 15 locations compared to ADP (Table 6.7).

These findings highlight the trade-off between solution quality and computational effort, suggesting that policy selection should consider both network characteristics and available computational resources.

6.2.3 Concave Cost Function in Multiple-Location Networks

To evaluate the performance of transshipment policies under more realistic cost structures that incorporate economies of scale, I extend the analysis to include a piecewise-linear concave transshipment cost function in multi-location networks.

6.2.4 Perfect Foresight MILP with Concave Costs

For the multi-location problem with concave transshipment costs, I reformulate the perfect foresight policy as a Mixed-Integer Linear Program (MILP) to serve as an upper bound:

$$\max \sum_{t} \left(\sum_{j \in \mathcal{L}} \sum_{i \in \mathcal{L}} p_j z^d_{ijt} - \sum_{j \in \mathcal{L}} \sum_{i \in \mathcal{L}} h_j z^s_{ijt} - \sum_{i \in \mathcal{L}} \sum_{j \in \mathcal{L}} c_{ij} \rho_{ij} \sum_{k=1}^{K} m_k s_{ijkt} \right) \tag{6.7}$$

Subject to:

$$\sum_{k=1}^{K} s_{ijkt} = z^d_{ijt} + z^s_{ijt} \quad \forall i, j, t \tag{6.8}$$

$$s_{ijkt} \leq W_k y_{ijkt} \quad \forall i, j, t, k \tag{6.9}$$

$$y_{ijkt} \geq y_{ijk+1,t} \quad \forall i, j, t, k = 1, \ldots, K-1 \tag{6.10}$$

$$\sum_j (z^d_{ij0} + z^s_{ij0}) = x_i \quad \forall i \tag{6.11}$$

$$\sum_j (z^d_{ijt} + z^s_{ijt}) = \sum_j z^s_{jit-1} \quad \forall i, t \geq 1 \tag{6.12}$$

$$\sum_i z^d_{ijt} \leq d_t(j) \quad \forall j, t \tag{6.13}$$

$$z^d_{ijt}, z^s_{ijt} \in \mathbb{Z}_{\geq 0}, \quad s_{ijkt} \geq 0, \quad y_{ijkt} \in \{0,1\} \tag{6.14}$$

This formulation extends the previous linear cost model by introducing:

- Segment variables s_{ijkt} representing flow in cost segment k on arc (i, j) at time t
- Binary variables y_{ijkt} indicating whether segment k is used
- Segment capacities W_k and marginal costs m_k that decrease with volume
- Constraints ensuring proper segment ordering and utilization

6.2.5 Experimental Setup and Results

We implemented a concave cost structure with the following parameters:

- Segment breakpoints: $U_k = [0, 100, 200, 400, 500]$ units
- Marginal costs: $m_k = [0.5, 0.25, 0.125, 0.075, 0.05]$ dollars per unit per distance

This structure represents substantial economies of scale, with marginal costs decreasing from \$0.5 for the first 100 units to \$0.05 for units beyond 500.

The results in Table 6.8 demonstrate that the ADP policy maintains strong performance under the concave cost structure across different network sizes and inventory configurations. However, it's important to note that these experiments have a significant limitation: the daily demand of approximately 28 units falls well below the first cost breakpoint of 100 units in our concave cost structure.

This limitation means that the experiments primarily test the policies' behavior in the highest cost segment rather than their ability to exploit economies of scale

Table 6.8: Percentage of upper bound with concave cost structure and multiple initial inventory levels set 2

	TIE			ADP		
Locations	697	680	$\overline{697}$	697	680	$\overline{697}$
5	94%	90%	98%	95%	95%	98%
10	94%	91%	59%	96%	99%	83%
15	97%	94%	87%	83%	96%	96%

through volume consolidation. Despite this constraint, the ADP policy shows remarkable robustness, consistently achieving near-optimal performance.

The TIE policy shows more variable performance, particularly in scenarios with imbalanced initial inventories (680 units) and larger networks. This variability highlights the importance of cost-aware decision-making in transshipment policies, especially when facing complex, non-linear cost structures.

To test the policy under different marginal costs and force the use of economies of scale, I have built a new set of segment breakpoints: $U_k = [0, 5, 20, 30, 40]$ units and the corresponding marginal costs: $m_k = [1, 0.5, 0.25, 0.2, 0.1]$ dollars per unit per distance. Considering that the mean demand in this case is $\lambda = 24$ units/day, the new breakpoints are more accurate for this case. The initial inventory level is set to 697 across all locations, the holding cost is \$5, and the price per unit is \$80. The results of this experiment, reported in Table 6.9 as percentages of the optimal transhipment policy, show that ADP and Lookahead policies perform the best in this case. Again, the issue with the Lookahead policy is the huge runtime when increase the number of location.

Table 6.9: Percentage of upper bound with concave cost structure and initial inventory levels fixed to 697 across multiple locations, with new breakpoints and marginal costs

Locations (L)	ADP	TIE	PL	NT	RC
5	94%	79%	97%	94%	94%
10	96%	45%	99%	94%	94%
15	99%	85%	98%	97%	97%

To check the full behavior of ADP using the new marginal costs: $m_k = [1, 0.5, 0.25, 0.2, 0.1]$ dollars per unit per distance and segment breakpoints: $U_k = [0, 5, 20, 30, 40]$, I have repeated the experiment with different initial inventories and numbers of locations. We can see from the results in Table 6.10 that ADP is the only policy that has used the economies of scale and improved the performance with respect to linear transhipment cost (Table 6.5).

6.2.6 Implications and Limitations

The experimental results, while limited by the demand-volume mismatch, suggest several important insights:

1. **Policy Robustness**: ADP demonstrates strong performance even when operating primarily in the highest cost segment, indicating good generalization

Table 6.10: Percentage of upper bound with concave cost structure and different initial inventories and number of location, using concave transhipment cost

	TIE			ADP			PL		
L	697	680	$\overline{697}$	697	680	$\overline{697}$	697	680	$\overline{697}$
5	57.96%	75.95%	52.24%	94.28%	94.40%	98.63%	97.38%	95.23%	96.81%
15	67.31%	79.24%	65.85%	97.96%	96.60%	93.08%	95.86%	91.30%	94.28%

to different cost structures.

2. **Economies of Scale Potential**: The results hint at the significant savings possible when demand volumes allow exploitation of lower marginal cost segments, though our current experiments don't fully capture this potential.

3. **Computational Considerations**: The MILP formulation for perfect foresight with concave costs is substantially more complex than the linear equivalent, requiring careful implementation and potentially limiting the scale of solvable problems.

4. **Real-world Relevance**: The concave cost structure more accurately reflects actual transportation economics, making these results more applicable to practical logistics operations.

Future work should focus on experimental designs with higher demand volumes that better match the cost structure breakpoints, allowing for more comprehensive evaluation of policies' abilities to exploit economies of scale through strategic shipment consolidation.

Chapter 7

Summary and Conclusion

7.1 Summary of Findings

This study has implemented and extended the approximate dynamic programming framework for lateral transshipment problems in multi-location inventory systems, replicating and validating the original authors' methodologies ("Approximate Dynamic Programming for Lateral Transshipment Problems in Multi-Location Inventory Systems" by Joern Meissner and Olga V. Senicheva) while introducing significant enhancements to the cost structure and policy implementations.

The experimental results across both two-location and multi-location networks demonstrate several key findings:

Two-Location Networks: The ADP policy consistently demonstrated performance nearly identical to the optimal dynamic programming solution (99% of scenarios showed no significant difference), substantially outperforming other heuristic approaches. The proactive lookahead (PL) policy also showed strong results, particularly in scenarios with higher demand variance. The value of transshipment policies increased significantly with demand variability, confirming the intuition that inventory redistribution provides greater benefits in uncertain environments.

Multi-Location Networks: In larger networks where exact DP solutions are computationally infeasible, the ADP policy maintained robust performance, achieving 94-100% of the perfect foresight upper bound across various configurations. The PL policy demonstrated particularly strong results in some cases, even outperforming ADP in certain scenarios, though with substantially higher computational requirements.

Cost Structure Analysis: The extension from linear to piecewise-linear concave transshipment costs represented a significant advancement in modeling realism. my results showed that economies of scale in transportation can substantially improve network profitability when properly exploited. The ADP policy effectively adapted to this non-linear cost structure, demonstrating its flexibility and robustness.

Policy Performance: Across all experiments, several patterns emerged:

- The ADP policy provided the most consistent performance across different network sizes, cost structures, and demand patterns
- The PL policy showed excellent results but with higher computational costs
- The TIE policy performed well in balanced scenarios but struggled with cost sensitivity and inventory imbalances
- Simple reactive policies (RC) provided modest improvements over NT in some cases
- The no-transshipment policy (NT) became increasingly disadvantageous as demand variability increased and economies of scale were introduced

7.2 Methodological Contributions

This work makes several methodological contributions to the field of inventory transshipment optimization:

First, I successfully implemented and validated the complete ADP framework described in the original paper, achieving similar results despite limited implementation details provided by the authors. This independent replication adds credibility to the original findings.

Second, I extended the modeling framework to incorporate a more realistic piecewise-linear concave cost function that captures economies of scale in transportation—a critical feature of real-world logistics operations that was absent from the original formulation.

Third, I developed MILP formulations for both the perfect foresight upper bound and the lookahead policy under the new cost structure, addressing the significant computational challenges introduced by the non-convex cost function.

Finally, I implemented a practical shadow price estimation heuristic that maintains computational tractability while providing reasonable approximations for the value function updates in the MILP context.

7.3 Limitations and Future Research Directions

While this study provides comprehensive insights into transshipment optimization, several limitations suggest directions for future work:

Parameter Sensitivity: Further investigation is needed regarding policy sensitivity to holding costs, product prices, and initial inventory levels. Systematic sensitivity analysis could provide valuable insights for practical implementation.

Transshipment Time Considerations: An important extension would incorporate transshipment lead times, where products require time to move between locations. This would more accurately reflect real-world logistics and might affect the relative performance of proactive versus reactive policies.

Backordering: Allowing unmet demand to be backordered rather than lost represents another valuable extension. This would particularly affect the evaluation of reactive policies, which might become more viable when stockouts don't necessarily result in lost sales.

Storage Locations: Introducing dedicated storage locations with low holding costs and no demand could create interesting opportunities for strategic inventory positioning and further optimization.

Additional Extensions: Other promising directions include:

- Non-stationary demand patterns to reflect seasonal variations
- Capacity constraints on transportation vehicles
- Multiple product types with different characteristics
- Integration with upstream supply chain decisions
- Real-time implementation considerations and computational efficiency improvements

7.4 Concluding Remarks

This thesis has demonstrated that approximate dynamic programming provides a powerful framework for solving complex transshipment problems in multi-location inventory systems. The ADP approach consistently delivered near-optimal performance across various network configurations and cost structures while maintaining computational feasibility for practical-sized problems.

The extension to concave cost functions significantly enhances the practical relevance of the models, acknowledging the economies of scale that characterize real-world transportation logistics. While introducing additional computational complexity, my results show that the ADP framework can successfully adapt to this more realistic cost structure.

The comprehensive implementation and testing of multiple policies provides practitioners with valuable insights for designing inventory redistribution strategies. The robust performance of the ADP policy suggests it as a preferred approach for systems where transshipment decisions have significant financial implications.

As inventory systems continue to grow in complexity and scale, the methods and insights developed in this research will contribute to more effective inventory management practices, reducing costs and improving service levels across supply chain networks.

Bibliography

[1] *Global Retail Inventory Management Trends*. IHS Markit Report. 2023 (cit. on p. 1).

[2] *Weathering the Storm: Building Supply Chain Resilience to Climate Risks*. Boston Consulting Group White Paper. 2024 (cit. on p. 2).

[3] *The Future of Demand Forecasting in Retail*. Forrester Research Report. 2023 (cit. on p. 2).

[4] D. Ivanov and A. Dolgui. «Viability of intertwined supply networks: extending the supply chain resilience angles through survivability». In: *International Journal of Production Research* 58.10 (2020), pp. 2904–2915 (cit. on p. 2).

[5] bsaikrishna. *The Bullwhip Effect in Supply Chain*. Updated Sep 19, 2023. Accessed: 21 Sep 2025. Mar. 2016. URL: `https://brandalyzer.blog/2016/03/23/the-bullwhip-effect-in-supply-chain/` (cit. on p. 3).

[6] *Final Report on the February 2021 Winter Storm and Power Outages*. Tech. rep. Austin, TX: Texas Power Outage Task Force, 2021 (cit. on p. 4).

[7] *Building Resilient Supply Chains: Lessons from Natural Disasters*. Tech. rep. Washington, DC: Federal Emergency Management Agency, 2022 (cit. on p. 4).

[8] Colin Paterson, Gudrun P. Kiesmüller, Ruud H. Teunter, and Kevin D. Glazebrook. «Inventory models with lateral transshipments: A review». In: *European Journal of Operational Research* 210.2 (2011). [review article], pp. 125–136. DOI: `10.1016/j.ejor.2010.05.048` (cit. on p. 4).

[9] L. Zhao, X. Wang, and C. G. Lee. «Preventive transshipment in LSP-led distribution systems». In: *Proceedings of the 2022 Industrial Engineering Research Conference*. 2022, pp. 1123–1128 (cit. on p. 5).

[10] H. Wong, G. J. van Houtum, D. Cattrysse, and D. van Oudheusden. «Multi-item inventory models with lateral transshipments». In: *International Journal of Production Economics* 108.1-2 (2007), pp. 300–304 (cit. on p. 5).

[11] X. Yan, X. Zhao, and Y. Cheng. «Analysis of factors influencing lateral transshipment decisions in supply chain networks». In: *Transportation Research Part E: Logistics and Transportation Review* 147 (2021), pp. 102–245 (cit. on p. 6).

[12] *Demand-Driven Transformation: The Key to Supply Chain Resilience*. AMR Research Report. 2022 (cit. on p. 6).

[13] J. Meissner and O. V. Senicheva. «Approximate dynamic programming for lateral transshipment problems in multi-location inventory systems». In: *European Journal of Operational Research* 249.1 (2016), pp. 47–57. DOI: 10.1016/j.ejor.2015.08.041 (cit. on pp. 6, 7, 18, 23, 26–33, 35, 37, 38, 45).

[14] K. S. Krishnan and V. R. K. Rao. «Inventory control in N warehouses». In: *Journal of Industrial Engineering* 16.4 (1965), pp. 212–215 (cit. on p. 8).

[15] L. W. Robinson. «Optimal and approximate policies in multiperiod, multilocation inventory models with transshipments». In: *Operations Research* 38.2 (1990), pp. 278–295 (cit. on p. 8).

[16] T. W. Archibald, S. E. Sassen, and L. C. Thomas. «An optimal policy for a two-depot inventory problem with stock transfer». In: *Management Science* 43.2 (1997), pp. 173–183 (cit. on p. 8).

[17] T. W. Archibald, B. C. Black, and K. D. Glazebrook. «The discrete-time spare parts inventory problem with a service-level constraint». In: *Journal of the Operational Research Society* 58.7 (2007), pp. 958–967 (cit. on p. 8).

[18] T. W. Archibald, C. Paterson, and K. D. Glazebrook. «A heuristic for the multi-item spare parts inventory problem with a service level agreement». In: *European Journal of Operational Research* 192.2 (2009), pp. 475–487 (cit. on p. 8).

[19] J. Grahovac and A. K. Chakravarty. «Sharing and lateral transshipment of inventory in a supply chain». In: *Management Science* 47.11 (2001), pp. 1579–1594 (cit. on p. 8).

[20] H. L. Lee. «A multi-echelon inventory model for repairable items with emergency lateral transshipments». In: *Management Science* 33.10 (1987), pp. 1302–1316 (cit. on p. 8).

[21] S. Axsäter. «A new decision rule for lateral transshipments in inventory systems». In: *International Journal of Production Economics* 81.1 (2003), pp. 101–107 (cit. on p. 8).

[22] S. Minner, E. H. S., and G. J. V. H. «A continuous-review inventory model with lateral transshipments». In: *OR Spectrum* 25.4 (2003), pp. 485–504 (cit. on p. 8).

[23] H. Wong, G. J. V. H., and J. C. F. «Order-level inventory systems with lateral transshipments». In: *IIE Transactions* 37.6 (2005), pp. 539–551 (cit. on p. 8).

[24] H. Wong, G. J. V. H., and J. C. F. «A periodic-review inventory model with lateral transshipments and a service-level constraint». In: *European Journal of Operational Research* 174.3 (2006), pp. 1806–1825 (cit. on p. 8).

[25] Y. T. Herer, E. D. R., and R. D. H. «A simulation study of an inventory system with lateral transshipments». In: *IIE Transactions* 38.4 (2006), pp. 287–298 (cit. on p. 8).

[26] S. G. Allen. «A redistribution model with set-up charge». In: *Management Science* 8.1 (1958), pp. 99–108 (cit. on p. 8).

[27] D. Gross. «Centralized inventory control in a multi-location supply system». In: *Multistage Inventory Models and Techniques*. Stanford, CA: Stanford University Press, 1963, pp. 47–83 (cit. on p. 8).

[28] U. S. Karmarkar. «A multi-period, multi-location inventory problem». In: *Operations Research* 28.2 (1980), pp. 241–254 (cit. on p. 9).

[29] J. Lee and M. Park. «A study on the proactive transshipment policy in a one-warehouse multi-retailer system». In: *Computers & Industrial Engineering* 48.2 (2005), pp. 247–260 (cit. on p. 9).

[30] C. Paterson, G. K., and R. T. «A continuous review inventory model with a hybrid push/pull transshipment policy». In: *International Journal of Production Economics* 140.2 (2012), pp. 856–863 (cit. on p. 9).

[31] F. Seidscher and S. Minner. «A continuous review inventory model with proactive transshipments». In: *OR Spectrum* 35.4 (2013), pp. 957–982 (cit. on p. 9).

[32] A. Banerjee, P. K. Banerjee, and B. Burton. «A simulation study of a two-echelon inventory system with periodic review and transshipment». In: *International Journal of Production Economics* 81.1 (2003), pp. 125–132 (cit. on p. 9).

[33] G. Van der Heide and K. J. Roodbergen. «A clustering and look-ahead heuristic for the multi-item inventory problem with lateral transshipments». In: *European Journal of Operational Research* 229.2 (2013), pp. 374–384 (cit. on p. 9).

[34] S. Agrawal, A. V. Iyer, and A. P. M. «An approximation algorithm for the capacitated dynamic lot-sizing problem with polyhedral capacities». In: *Operations Research* 52.2 (2004), pp. 269–281 (cit. on p. 9).

[35] T. W. Archibald, C. Paterson, K. D. Glazebrook, and A. J. R. Thomas. «A computationally efficient method for solving the multi-item newsvendor problem with a service level constraint». In: *Journal of the Operational Research Society* 61.2 (2009), pp. 263–273 (cit. on pp. 9, 45).

[36] Y. Lu and M. Song. «Inventory Control with a Fixed Cost and a Piecewise Linear Convex Cost». In: *Operations Research* 62.5 (2014), pp. 1163–1174 (cit. on p. 10).

[37] D. P. Bertsekas. «Dynamic programming and dimensionality in convex stochastic optimization and control». In: *arXiv preprint* (2022). arXiv:2203.01258 (cit. on p. 10).

[38] C. Glaas and S. Minner. «Replenishment and transshipment in periodic-review systems with a fixed order cost». In: *Manufacturing & Service Operations Management* 24.2 (2022), pp. 1070–1087 (cit. on p. 10).

[39] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. 2nd. Hoboken, NJ: Wiley, 2011 (cit. on pp. 18, 23, 27, 28).