Inventory Management Optimization: A Case Study of Amazon Supply Chain

by

Bahareh Aghababaei

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN BUSINESS ADMINISTRATION

in

The Faculty of Graduate and Postdoctoral Studies

(Transportation and Logistics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

July 2024

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Inventory Management Optimization: A Case Study of Amazon Supply Chain

submitted by Bahareh Aghababaei in partial fulfilment of the requirements for

the degree of Master of Science in Business Administration

in Transportation and Logistics

Examining Committee:

Tamar Cohen-Hillel, Assistant Professor, Operations and Logistics, Sauder School of Business, UBC
Supervisor

Harish Krishnan, Professor, Operations and Logistics, Sauder School of Business, UBC
Supervisory Committee Member

Julia Yan, Assistant Professor, Operations and Logistics, Sauder School of Business, UBC
Supervisory Committee Member

Ning Nan, Associate Professor, Accounting and Information Systems, Sauder School of Business, UBC
Additional Examiner

Abstract

One of the unique characteristics of online retail is the ability to ship items to customers from different fulfillment centers. This causes interdependence between fulfillment centers and creates new challenges in inventory management. Hence, this thesis investigates inventory management decisions in online retail, focusing on the initial stocking of fulfillment centers, their inventory levels, and order fulfillment processes. The problem is formulated as a dynamic integer program, with the objective of minimizing the sum of receiving, holding, and shipping costs while satisfying demand and capacity constraints. The large scale of the problem arises from the size of the fulfillment centers, the number of items, and the number of demand points. We propose a large-scale solution algorithm that aggregates the three sets of items, warehouses, and customers. The algorithm solves the aggregated problem with an exact solution that is tractable in size and then disaggregates the solution through three hierarchical disaggregation stages using a Greedy algorithm to address the NP-hardness of the problem.

We develop a theoretical bound on the optimality gap caused by aggregation method. The numerical examples demonstrate that the large-scale algorithm is a computationally efficient approach which produce low-cost solutions for medium-sized clusters compared to the optimal model. Additionally, we also study the impact of key factors, including cost parameters and variations in item sizes on the optimality gap.

Lay Summary

This thesis addresses the challenge of optimizing inventory management in online retail, focusing on initial stocking, inventory replenishment, and planned order fulfillment. We aim to minimize costs associated with receiving, holding, and outbound shipping while meeting customer demand and capacity constraints. We develop a large-scale framework that integrates an aggregation approach with a Greedy algorithm to create a trade off between computation time and optimality gap. The analysis indicates that this approach provides near-optimal solutions quickly and efficiently, particularly for medium-sized problems, making it a robust alternative to traditional methods that struggle with large datasets. Additionally, our study provides theoretical insights into the algorithm's effectiveness and explores how factors like cost parameters and variations in item sizes impact performance.

Preface

This thesis is original, unpublished, independent work by the author, Bahareh Aghababaei. It has benefited greatly from the extensive feedback and insightful guidance provided by Professor Tamar Cohen-Hillel.

Table of Contents

List of Tables

List of Figures

Acknowledgements

The author acknowledges that the work for this project took place on the traditional, ancestral, and unceded territory of the xʷməθkʷəy̓əm (Musqueam) People.

I would like to thank the faculty, staff, and students at UBC who have enriched my two years of learning here.

I am profoundly grateful to my supervisor, Professor Tamar Cohen-Hillel, for the opportunity to collaborate on this challenging project. Her unwavering support, both moral and financial, and her faith in my abilities were instrumental in the completion of this work. Professor Cohen-Hillel's insightful guidance, combined with her extensive experience in the Amazon supply chain, provided invaluable perspectives and inspired me to continue pursuing research in this field.

I also appreciate the time and constructive feedback from Professor Harish Krishnan and Professor Julia Yan. My sincere thanks to Professor Joseph Paat for his exceptional teaching in combinatorial optimization and for addressing my numerous questions with clarity.

Special thanks go to my husband for his constant support throughout my educational journey at UBC.

1. Introduction

The increasingly competitive retail business demands greater customer satisfaction and operational efficiencies, including optimal inventory and logistics planning (Hübner et al., 2013). For example, lack of optimal initial stock planning could lead to stock reduction resulting in shortage costs, delays, and eventually missed customers (Seyedan et al., 2022). To avoid such issues and maintain desired service levels, retailers often tend to keep large inventories, incurring substantial storage costs, or, in some cases, paying additional logistics costs to fulfill customer demand from distant warehouses (Salam et al., 2016). Consequently, in order to meet customer demand at the optimal time and with the minimum possible cost, the retail business must optimize initial stocking and fulfillment processes at the operational level, taking into account various factors affecting the planning.

Moreover, large retail industries, such as Amazon, face complex planning decisions involving millions of individual items and numerous facilities to satisfy thousands of demand points. Amazon's fulfillment network consists of three key facility types: fulfillment centers, sortation centers, and delivery stations. Fulfillment centers are extensive facilities specializing in handling large volumes of diverse orders shipped as parcels, and categorized by item types (e.g., apparel, electronics) and sizes (small sortable, large sortable, large non-sortable). Sortation centers, large-sized facilities, organize shipments based on regional or local destinations, sorting parcels into smaller batches by postal code. Delivery stations, medium-sized facilities, focus on sorting parcels for specific local delivery routes, loading them onto appropriate vehicles for efficient last-mile delivery (Rodrigue, 2020).

According to MWPVL International dataset, there are currently 376 active fulfilment centers, 111 sortation centers, and 566 delivery stations operating only in the United States in 2024 (Amazon Distribution Network Strategy | MWPVL International, n.d.). With warehouse capacity limitations, supply fluctuations, and the fact that different demand points vary in demand patterns for the same items, managing this extensive network, which aims to satisfy thousands of demand points, is a challenging task. The complexity of the problem is further increased by the fact that we are dealing with discrete optimization, and the integrality of the solution necessitates solving a very big combinatorial problem. As a result, the inventory and fulfilment planning for such an extensive network is hard.

In this work, our objective is to address the inventory management decisions for an online retailer through a proposed heuristic algorithm. The primary goal is to determine a cost-minimizing plan that simultaneously address three key decisions: initial inventory placement, inventory replenishment, and planned order fulfilment. The model is developed on a large-scale two-tier network, comprising fulfilment centers and customers, over a multi-period planning horizon. Our algorithm consists of a four-stage approach, wherein the first stage employs an auxiliary model utilizing aggregation analysis to reduce the dimension and complexity compared to the original model.

Aggregation analysis incorporates cluster analysis techniques and additionally employs the method of combining similar data. The process of identifying which data are sufficiently similar to be grouped together is called cluster analysis, and it involves a large portion of the aggregation techniques that should be considered for the optimization models. These clusters may frequently be predefined in a wide range of optimization issues (Rogers et al., 1991). In this case, for example, clustering is applied to the three sets of demand points, items, and warehouses due to their huge sizes, showing that the entities to be grouped are variables during the aggregation stage. The next sections will provide comprehensive descriptions of the aggregation step.

The next three steps include an attempt to recover the original solution through three stages of disaggregation using the outcomes of the reduced model's solution. Disaggregation analysis refers to techniques for taking an aggregated model and using it to derive components of a more refined model (usually with the same dimensions as the original model) (Rogers et al., 1991). In this context, we design a hierarchical disaggregation method in which demand zones are firstly disaggregated, followed by warehouses, and finally items. The first stage of this process is carried out optimally, while the second and third stages are completed approximately with the aid of a greedy algorithm.

Our approach is depicted in Figure 1. The main contributions of this work are threefold. First, we develop an aggregation and disaggregation algorithm to address the NP-hard nature of the inventory and fulfillment planning model for the Amazon supply chain. Second, we apply a greedy algorithm which produces near-optimal solutions to solve the warehouse and item disaggregated models given our parameters and solution obtained from the demand zones disaggregated model. The greedy algorithm is capable of solving in polynomial time. Finally, we provide computational experiments on simulated instances of the Amazon distribution network to demonstrate the algorithm's performance.

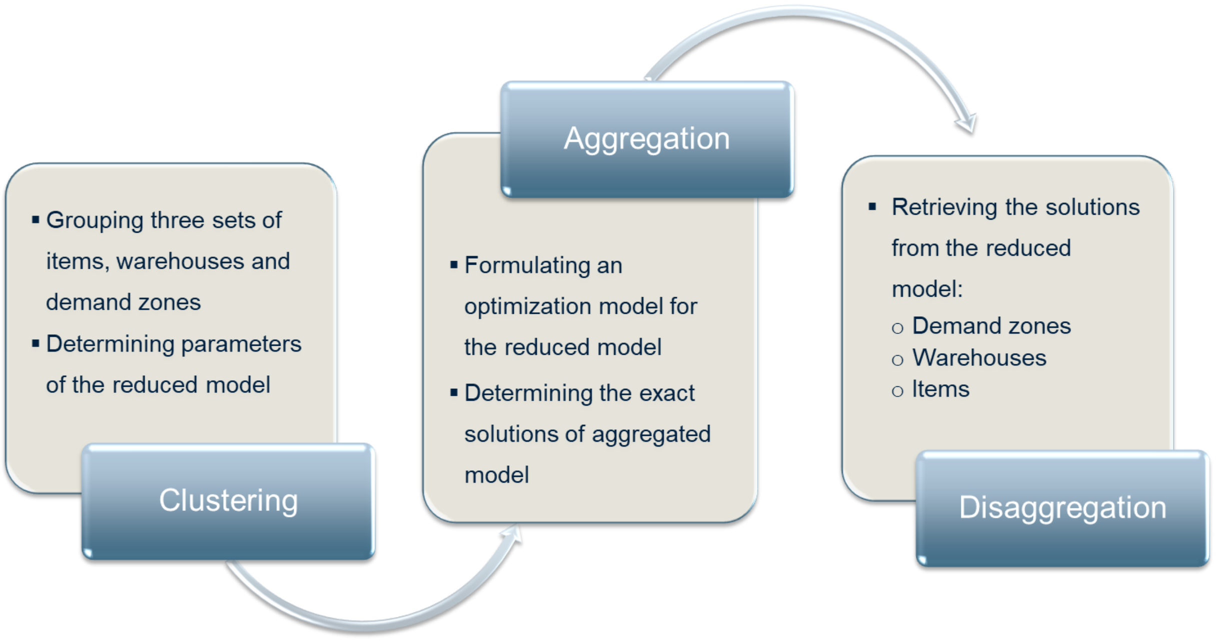

Aggregation
Grouping three sets of items, warehouses and demand zones
Determining parameters of the reduced model
Clustering
Formulating an optimization model for the reduced model
Determining the exact solutions of aggregated model
Retrieving the solutions from the reduced model:
Demand zones
Warehouses
Items
Disaggregation

Figure 1. The proposed methodology

2. Literature Review

In the retail industry, businesses use a variety of sales channels, including online platforms, physical stores, and a combination of both within their supply chains. Online sales have been experiencing rapid annual growth. For instance, while it includes a 15.4 percent share of total US sales in 2023, it experienced a considerable 7.6 percent rise over the previous year (U.S. Census Bureau, 2023).

One of the key aspects of online retail is its capability to fulfill customer orders from various fulfillment centers, regardless of their geographical proximity, each with its associated shipping costs (DeValve et al., 2021). While the shipping flexibility in online retail allows for cost savings and improved service quality, it also needs the development of inventory optimization methods to manage the connection between fulfillment centers. To achieve this, a range of decisions must be made depending on the planning horizon, such as network design, sourcing, inventory placement, replenishment, fulfillment, and transshipment (Ge et al., 2019).

Given the interconnected nature of these decisions, authors have suggested a number of approaches to optimize them, either independently in a hierarchical manner or jointly. Chen & Graves (2021), for instance, focus on optimizing initial inventory placement in online retail, developing a cost-minimizing plan for each item while considering factors such as fulfillment center capacity constraints and customer satisfaction. In another study, they also explore operational decisions to address challenges arising from sorting and picking multiple-item orders within a warehouse. Their approach involves analyzing the problem from batch scheduling and routing perspectives, leading to a proposed picking policy that optimizes efficiency in both picking and sorting tasks Chen, (2018).

Order fulfillment, a bottleneck for online retailers, has gained attention in research. In a recent study, Arlotto et al. (2023) investigate a joint inventory placement-fulfillment problem via regret analysis. At the beginning, the inventory of a single item is dispersed across different warehouses, then the decision maker determines whether to accept or reject the order at each period. They provide heuristic policies and evaluate their effectiveness in terms of minimum-inventory regret to the problem. Torabi et al. (2015) addressed the joint fulfillment and transshipment decisions by employing a mixed integer programming model with Benders decomposition to efficiently find the optimize solutions. Some studies define the order fulfillment process into three decisions: order allocation, order consolidation and synchronization, and order delivery. In this context, Li et al. (2019) integrate these decisions into an online retailer's fulfillment plan and develop an adaptive large neighbourhood search method to handle large-scale instances of this problem. Building upon this work, Jiang & Li (2020) extend it by incorporating time windows to minimize transportation costs, as well as introducing a decomposition-based approach to reduce the computational complexity of solving large instances.

Several recent studies address inventory management decisions within an omnichannel setting (Abouelrous et al., 2022; Wang et al., 2022). Bretthauer et al. (2010) determine optimal inventory allocation across different sites to meet in-store and online demand while minimizing overall costs, including holding, backorder, operating, transportation, and handling costs. Govindarajan et al. (2021) develop network-based strategies considering online demand spillover, where inventory decisions are made at the beginning of the selling horizon and fulfillment decisions are made iteratively. They derive inventory heuristics based on a two-stage approximation. In addition, their fulfillment heuristic provides location-specific inventory thresholds that determine how much inventory is rationed between in-store and online demands.

The goal of this research is similar to that of Chen & Graves (2021), who study the problem of inventory placement across fulfilment centers considering capacity restrictions. However, we specifically focus on variable costs related to receiving, storage, and shipping rather than the fixed charge related to assigning each item to a fulfillment center. Additionally, we aim to simultaneously optimize the following decisions to address their interdependent nature: 1) inventory placement or initial stocking (determining which fulfillment centers store each item), 2) inventory replenishment (deciding how much inventory to hold), and 3) planned order fulfillment (selecting which fulfillment center to use for an order).

If we ignore the fact that items differ in dimensions and each occupies a specific space in a warehouse, the problem reduces to the integral flow problem, which has been proven to be an easy one given the fact that the matrices forming the set of constraints has a total unimodularity property (Conforti et al., 2014a; Hoffman et al., 1956). In other words, it would be a special case of a transportation problem on a bipartite graph, where two sets of nodes correspond to fulfillment centers and customer regions, respectively (Hitchcock, 1941). The problem, however, is a generalized assignment problem (GAP) which falls into the category of multiple knapsack problems (MKP), in which the number of identical copies available for each object type is restricted, each item j that is assigned to knapsack i yields profit P_{ij} (rather than profit Pj). This problem has been shown to be np-complete (Kellerer et al., 2004) and will be further discussed in the subsequent sections.

According to Martello & Toth (1990), there are three types of approaches to solving generalized assignment problems: 1) exact algorithms, such as Variants of branch-and-bound (Sarin et al., 2014; Woodcock & Wilson, 2010), 2) heuristic algorithms (Fathollahi-Fard et al., 2023; Saeedi et al., 2024), and 3) approximation algorithms (Cohen et al., 2006). When GAP is combined with a network flow problem, its complexity increases, particularly with a substantial number of nodes. Hence, in online retail networks, which involve numerous demand points, items, and warehouses, it is critical to employ appropriate techniques to handle the inherent intractability caused by both NP-hardness and the scale of the problem. Various approaches, such as decomposition-based algorithms like those discussed by Dell'Amico et al. (2019), are employed to handle large-scale linear integer programming models. However, the effectiveness of these algorithms depends on the specific characteristics of the problems, including variables and constraint structures. Despite their practical efficiency, they may not necessarily guarantee a solution within polynomial time. In this situation, it could be necessary to use some approximation techniques. The heuristic approach we provide here is the integration of aggregation technique and a greedy algorithm to handle disaggregation stage.

Aggregation is an efficient technique to facilitate the analysis of large optimization models and provide a set of tools to transform the model with a high degree of complexity to a reduced manageable form. The simplified model can then be solved, and its optimal solution disaggregated into a feasible solution for the original problem (Å. Hallefjord et al., 1993). Rogers et al. (1991) establish a framework for aggregation and disaggregation methodology and provide a comprehensive review of previous research and applications in various areas of operations research. Later, Litvinchev & Tsurkov (2003) delve into aggregation analysis by reviewing the findings from previous decades, different aggregation techniques, and discussing aggregation error and bounds. Aggregation, as a technique for reducing computational complexity, has been widely applied in many contexts, particularly in those involving a large number of variables and constraints. For example, applications of aggregation theory in optimization include addressing the classical transportation problem by aggregating demand points (Geoffrion, 1976), supply points (Evans, 1979), and both (Zipkin, 1980a). Aggregation has been also employed to reduce the number of items in a multi-commodities network flow problem (Kazemi, 2021), production planning (Leisten, 1998), and online retail inventory management (Chen & Graves, 2021).

In this paper, we aim to perform aggregation on nodes and items simultaneously within an integer programming framework. To guarantee feasibility, we develop a problem-specific hierarchal disaggregation procedure which consists of three stages; the first stage is executed optimally, while the second and third ones employ a greedy algorithm to handle their NP-hardness. One main part of the aggregation approach involves quantifying the degree of accuracy lost under specific aggregation. Using duality theory, Zipkin provided bounds on the loss in objective function value that results from column (variables) aggregation (Zipkin, 1980c), and row (constraints) aggregation (Zipkin, 1980b) in linear programming models. A. Hallefjord & Storoey (1986) remark that Zipkin's results can be generalized to apply to discrete optimization problems as well. Here, our theoretical bound is based on input parameters and is specifically obtained from the final disaggregation stage.

3. Problem description

In this study, we address the joint optimization of initial stocking, inventory replenishment and planned order fulfilment problem for an online retailer across a multi-period planning horizon. The goal is to determine a cost-minimizing plan which meets demand and supply requirements while taking fulfillment center capacity limitations into account. As previously stated, the retailer operates and manages a two-echelon order fulfilment system. The first tier of this system consists of a set of fulfilment centres (FCs), where retailer's products are stored, while the second tier comprises individual customer nodes. Although the real-world system also involves a full network of sortation centers and delivery stations between these two layers, our study does not consider these intermediate facilities.

Given the projected customer demand for each product, the retailer needs to decide on how to best place items in the fulfilment centers and fulfill customer orders, aiming to minimize overall expenses. Our focus lies on the variable costs associated with both inbound and outbound logistics. Inbound costs encompass expenses related to assigning items to fulfillment centers and storage, while outbound costs involve shipping expenses from fulfillment centers to customer demand points. The item assignment costs to fulfillment centers include any associated receiving expenses such as labor and inbound delivery costs. In addition, shipping costs from fulfillment centers to each customer point are computed based on the distance between nodes and a unit rate per distance, along with any additional handling charges incurred by the retailer. If the retailer employs a third-party service like UPS for item delivery, this rate can reflect the fee paid by the retailer to the carrier, or it can be considered a fuel rate if the retailer utilizes its own vehicles and resources for transportation.

To ensure a balance between practicality and tractability, we make the following assumptions:

- Transshipment between fulfillment centers (FCs) is prohibited to avoid additional costs in order fulfillment.
- We exclude location strategic planning considerations; network structure and sourcing plans are predetermined so that the number and the locations of fulfilment centers and demand points, fulfillment center capacities, and supply amounts are given as input.

- Every customer demand must be satisfied within each period; there are no shortages in terms of backorders or lost sales.
- Demand and supply values are deterministic but dynamically changes over time.
- Edge capacities are unrestricted; there are enough vehicles available for transferring products from FCs to demand zones.

Given these assumptions, we formulate the problem of initial stocking and order fulfillment for an online retailer as an integer linear program. In the following, we outline the sets, parameters, and variables defining the proposed model.

Table 1. The indices and notations used in the optimal model

Sets	
T	The set of time periods
I	The set of demand points
F	The set of Fulfilment centers
P	The set of items
Parameters	
d_{pit}	Demand of demand zone i for item p in period t
α_{pt}	Total number of available items p in period t
β_{fipt}	The unit of shipment cost from the fulfilment centre f to demand zone i in period t
h_{pft}	The unit of storage cost of item p in fulfilment center f in period t
ϖ_{ft}	The capacity of fulfilment center f in period t
r_{pft}	The processing cost of item p in fulfilment center f in period t
v_p	The volume of item p
Decision Variables	
x_{pft}	Quantity of item p shipped to fulfilment center f in period t
y_{fipt}	Quantity of item p shipped from fulfilment center f to demand zone i in period t
q_{pft}	The inventory level of fulfilment center f for item p in period t

We can formulate the online-retail inventory management problem (1) as follows:

$$Z_{OPT} = \underset{x,q,y}{Min} \left(\sum_{p,f,t} r_{pft} * x_{pft} + \sum_{p,f,t} h_{pft} * q_{pft} + \sum_{f,i,p,t} \beta_{fipt} * y_{fipt} \right) \quad (1a)$$

subject to:

$$\sum_{f} y_{fipt} \geq d_{pit} \qquad \forall\, p, i, t \quad (1b)$$

$$q_{pft} = q_{pf,t-1} + x_{pft} - \sum_{i} y_{fipt} \qquad \forall\, p, f, t \quad (1c)$$

$$\sum_{f} x_{pft} \leq \alpha_{pt} \qquad \forall\, p, t \quad (1d)$$

$$\sum_{p} v_p q_{pf,t-1} + \sum_{p} v_p x_{pft} \leq \varpi_{ft} \qquad \forall\, f, t \quad (1e)$$

$$x_{pft},\, q_{pft}, y_{fipt} \in Z^{+} \quad (1f)$$

The objective function ($1a$) aims to minimize the overall costs related to processing, storage, and transportation. The first constraint set (1b) guarantees that the total quantity shipped from all fulfilment centres must meet the demand of item p for demand zone i in each time period. The second equation (1c) captures the inventory balance for each fulfilment centre within each item and time period. It ensures that the inventory level at each centre is equal to the inventory level from the previous period plus the quantity assigned to the centre, subtracting the total quantity shipped from this centre to all demand zones. The third constraint set (1d) guarantees that the total quantity assigned to all fulfilment centres does not exceed the total supply. The constraint set (1e) indicates that the sum of the quantity assigned to fulfillment center f and the inventory carried over from the previous period must be less than or equal to the storage capacity of fulfillment center f. The last constraint set (1f) defines the domain and type of decision variables. Among variables defined above, x_{pft} and y_{fipt} are key variables, while q_{pft} is a secondary variable, as its value can be obtained from x_{pft} and y_{fipt} through flow balance equation.

3.1. Complexity of problem

Here, we interpret the original program as a network flow problem, incorporating additional 'non-network' type side constraints (1e). First, the constraint sets (1b) and (1d) represent capacity constraints, which can be expressed as $l \leq x \leq u$. To demonstrate that equation set (1c) is a network conservation equation, shift all variables in (1c) to the left-hand side of the equations. Consequently, each variable in (1c) appears at most twice, once with a coefficient of + 1 and once with a coefficient of -1, which can be written as $Bx = 0$, where B is a total unimodular matrix. In addition, the constraint set (1e) is a multiple knapsack problem with weights λ_j and a set of knapsacks $M = \{1, \dots, m\}$ with positive capacities ψ_i which can be written in the form of $\sum_{j=1}^{N} \lambda_j x_{ij} \leq \psi_i, \ i = 1, \dots, m$ or equivalently $\lambda^T x \leq \psi$. Therefore, the original problem can be presented as:

$$\min\{c^T x\text{: } Bx = 0 , \quad \lambda^T x \leq \Psi, \quad l \leq x \leq u \text{ and } x \in z^n\}$$

Where $M = F, N = P, \lambda^T = v, \psi = \varpi, l = \alpha, u = d, c^T = (r\ h\ \beta)$, and $B = \{-1,0,1\}^{|A|*|E|}$. Here, $|A|$ and $|E|$ denote the number of nodes and arcs in the corresponding network, respectively.

As a result, the problem combines aspects of network flow and bounded multiple knapsack problems. Our objective is to demonstrate the NP-completeness of the aforementioned problem. The complexity of the problem depends on the complexity of the knapsack constraint. If we apply the Lagrangian relaxation method to this constraint, the reaming problem is transformed to a linear programming (LP) model which can be solved in polynomial time. The following paragraphs provide clarification on why we believe the problem is NP-complete.

To demonstrate the dynamic online-retail inventory management Problem is NP-hard in the strong sense, we consider its decision version and prove that the decision version of the Dynamic Promotion Planning Optimization Problem is NP-complete.

DEFINITION 1.3.1 (decision dynamic online-retail inventory management problem). Given a set of items $p \in P$, warehouses $f \in F$, demand zones $i \in I$, planning period $t \in T$, and parameters demand d_{pit} , supply α_{pt}, transportation cost β_{fipt}, holding cost h_{pft}, receiving cost r_{pft}, warehouse capacity ϖ_f, and item volume v_p, does there exist a feasible solution to the formulation (1) with an objective value less than or equal Ω?

We propose a polynomial time reduction from the bounded multiple knapsack problem to the decision dynamic online-retail inventory management problem. We demonstrate that for every instance of the bounded multiple knapsack problem (denoted by I_{MKP}) and its reduced decision dynamic online-retail inventory management problem instance (denoted by I_{DIM}), the answer to I_{MKP} is "yes" if, and only if, the answer to I_{DIM} is "yes".

DEFINITION 2.3.1 (Bounded Multiple Knapsack Problem). Given a finite set M of knapsacks, each with a capacity of ψ_l for $l \in M$, and a finite set N of items, each with a corresponding weight λ_j and cost ς_j in which o_j identical copies of item type j must be used for $j \in N$, does there exist a feasible solution to the following formulation with objective function value at most δ?

$$BMKP - Decision = \left\{ \begin{array}{l} \quad \textit{there exists an } x \textit{ with} \\ \sum_{l,j} \varsigma_j \ddot{x}_{jl} \leq \delta \\ \sum_{j=1}^{n} \lambda_j \ddot{x}_{jl} \leq \psi_l, \quad l = 1, \dots, M \\ \sum_{l=1}^{m} \ddot{x}_{jl} \geq o_j, \quad j = 1, \dots, N \\ \quad \ddot{x}_{jl} \in Z^+, \;\; l, j = 1, \dots, N \end{array} \right\}$$

The bounded Multiple knapsack problem has been shown to be NP-complete (Kellerer et al., 2004). In what follows, we describe a reduction from the bounded multiple knapsack problem to the decision dynamic online-retail inventory management problem.

Polynomial Reduction. Here, we take the instance of the I_{MKP} and transform it to the instance of the I_{DIM}. Let each knapsack $l \in M$ corresponds to a warehouse $f \in F$, and each item $j \in N$ in I_{MKP} corresponds to a product $p \in P$ in I_{DIM}. Let $P = N, F =$ M, and $|T| = 1$. For each item $p \in P$, demand points $i \in I$ and periods $t \in T$, the parameters of objective function are defined as $r_{pft} = \varsigma_j, h_{pft} = \varsigma_j, \beta_{fip} = \varsigma_j$. For each item $p \in P$, the volume is $v_p = \lambda_j$, and demand is set to be $d_{pit} = o_j$. The capacity of warehouse f is $\varpi_f = \psi_l$.

Finally, the objective function of I_{DIM} is equal to $\sum_{p,f,t} \varsigma_j * x_{pft} + \sum_{p,f,t} \varsigma_j * q_{pft} + \sum_{f,i,p,t} \varsigma_j * y_{fipt}$. According to this reduction $|T| = 1$, so $q_{pft} = x_{pft} - \sum_i y_{fipt}$, so we have $\sum_{p,f,t} \varsigma_j * x_{pft} + \sum_{p,f,t} \varsigma_j \left(x_{pft} - \sum_i y_{fipt}\right) + \sum_{f,i,p,t} \varsigma_j * y_{fipt} = 2 \sum_{p,f,t} \varsigma_j * x_{pft}$. Thus $\Omega = 2 \sum_{p,f,t} \varsigma_j * x_{pft}$. We can also map the objective function I_{MKP} to the objective function of the original problem. Since $\sum_{l,j} \varsigma_j \ddot{x}_{jl} \leq \delta$, the value of objective function I_{DIM} will be at most 2δ, therefore $\Omega = 2\delta$.

- **Proof that "yes" to I_{MKP} Implies "yes" to I_{DIM} .**

Given a feasible solution for I_{MKP} defined as follows:

$\exists\, \ddot{x}_{jl} \in Z^+ \text{ such that } \sum_{l=1}^{m} \ddot{x}_{jl} \geq o_j$ and $\sum_{j=1}^{n} \lambda_j \ddot{x}_{jl} \leq \psi_l \quad \forall\, l \in M,\ j \in N$, consider the following policy for I_{DIM} such that

$$\mathrm{A}_{(I_{DIM})} = \begin{cases} x_{pft} = \ddot{x}_{jl} & \\ y_{fipt} = min\,\{\ddot{x}_{jl}, d_{pit}\} & if \ \exists\, \ddot{x}_{jl} \in Z^+ \ such \ that \displaystyle\sum_{l=1}^{m} \ddot{x}_{jl} \geq o_j \\ q_{pft} = x_{pft} - \displaystyle\sum_{i} y_{fipt} & and \displaystyle\sum_{j=1}^{n} \lambda_j \ddot{x}_{jl} \leq \psi_l \ \ \forall\, l \in M, \quad j \in N \end{cases}$$

Where $d_{pit} = o_j$, $\varpi_f = \psi_l$, $v_p = \lambda_j$, and $d_{pit} \leq \alpha_{pt}$. The above policy $\mathrm{A}_{(I_{DIM})}$ is feasible for I_{DIM} as well. This is because if $\ddot{x}_{jl}$ is a feasible solution to I_{MKP}, the capacity and demand fulfilment constraints (1b) and (1e) are immediately satisfied. Since we have $d_{pit} \leq \alpha_{pt}$, the constraint (1d) is also satisfied. Thus, a feasible solution for I_{MKP} guarantees a feasible policy for I_{DIM}.

Moreover, since $\ddot{x}_{jl}$ is feasible for I_{MKP}, it implies: $\sum_{l,j} \varsigma_j \ddot{x}_{jl} \leq \delta$, Consider $r_{pft} = h_{pft} = \beta_{fip} = \varsigma_j \quad \forall p \in P, i \in I, f \in F$. Substituting these into the objective function (1-1), we have:

$$\sum_{p,f,t} \varsigma_j * x_{pft} + \sum_{p,f,t} \varsigma_j * \left(x_{pft} - \sum_{i} y_{fipt} \right) + \sum_{p,f,t} \varsigma_j * \sum_{i} y_{fipt} = 2 \sum_{p,f,t} \varsigma_j * x_{pft} = 2 \sum_{j,l} \varsigma_j * \ddot{x}_{jl} \leq 2\delta$$

Where $2\delta = \Omega$. Hence, given that there is a feasible solution for I_{MKP}, there exists an optimal policy to I_{DIM} with the cost at most Ω. □

- **Proof that "yes" to I_{DIM} Implies "yes" to I_{MKP}.**

A feasible solution for I_{DIM} implies all the constraints of the formulation (1) are satisfied. Given a feasible solution for I_{DIM}, consider the following policy for I_{MKP} such that

$$\mathrm{A}'_{(I_{MKP})} = \{ \ddot{x}_{jl} = \tilde{x}_{pft} \quad if \ \exists \ feasible \ solution \ for \ I_{DIM}$$

Having $|T| = 1$, the policy $\mathrm{A}'_{(I_{MKP})}$ will be also feasible for I_{MKP} as well. When $\tilde{x}_{pft}$ is feasible for I_{DIM}, it means the constraints $\sum_{j=1}^{n} \lambda_j \ddot{x}_{jl} \leq \psi_l$ and $\sum_{l=1}^{m} \ddot{x}_{jl} \geq o_j$ are met automatically. Additionally, since the feasible solution to I_{DIM} has a cost at most Ω, $\Omega \geq \sum_{p,f,t} \varsigma_j * \tilde{x}_{pft} + \sum_{p,f,t} \varsigma_j * \tilde{q}_{pft} + \sum_{f,i,p,t} \varsigma_j * \tilde{y}_{fipt}$. In the optimal solution $\tilde{x}_{pft} = \sum_i \tilde{y}_{fipt}$, then $\tilde{q}_{pft} = \left(\tilde{x}_{pft} - \sum_i \tilde{y}_{fipt}\right)$, so we have

$$\Omega \geq \sum_{p,f,t} r_{pft} * \tilde{x}_{pft} + \sum_{p,f,t} h_{pft} * \left(\tilde{x}_{pft} - \sum_{i} \tilde{y}_{fipt}\right) + \sum_{f,i,p,t} \beta_{fipt} * \tilde{y}_{fipt}$$

$$= \sum_{p,f,t} \left(r_{pft} + h_{pft}\right) * \tilde{x}_{pft} + \sum_{f,i,p,t} \left(\beta_{fipt} - h_{pft}\right) * \tilde{y}_{fipt}$$

$$= \sum_{p,f,t} \left(r_{pft} + h_{pft}\right) * \sum_{i} \tilde{y}_{fipt} + \sum_{f,i,p,t} \left(\beta_{fipt} - h_{pft}\right) * \tilde{y}_{fipt}$$

$$= \sum_{p,f,i,t} \left(r_{pft} + \beta_{fipt}\right) * \tilde{y}_{fipt}$$

Considering $r_{pft} = \beta_{fipt} = \varsigma_j$, we have $\Omega \geq 2 \sum_{p,f,i,t} \varsigma_j * \tilde{y}_{fipt}$ which leads to $\sum_{p,f,t} \varsigma_j * \sum_i \tilde{y}_{fipt} = \sum_{p,f,t} \varsigma_j * \tilde{x}_{pft} \leq \frac{\Omega}{2}$. According to the reduction described earlier $\Omega = 2\delta$. As a result, the constraint $\sum_{l,j} \varsigma_j \ddot{x}_{jl} \leq \delta$ will be satisfied. Therefore, given that there is an optimal policy to I_{DIM} with cost at most Ω, there exists a feasible solution for I_{MKP}. □

Considering the polynomial time reduction from the bounded multiple knapsack problem to the Decision dynamic online-retail inventory management problem and given that every instance of the bounded multiple knapsack problem, I_{MKP}, can be reduced to the Decision dynamic online-retail inventory management problem instance I_{DIM}, BMKP-decision has a feasible solution if and only if the corresponding decision dynamic online-retail inventory management problem has a feasible solution. In other words, the answer to I_{MKP} is "yes" if and only if, the answer to I_{DIM} is "yes". Therefore, it can be concluded that the Decision dynamic online-retail inventory management problem is equally hard as the problem of bounded multiple knapsack problem. Hence, the dynamic online-retail inventory management Problem is NP-complete.

4. Approximation Algorithm

Given that the optimal model (1) is NP-hard in the strong sense, finding an exact solution is computationally intractable even for small instances. As a result, heuristic techniques such as the Greedy algorithm offer a practical approach to achieve near-optimal solutions within a reasonable time. The Greedy algorithm is an optimization technique that makes the best local choice at each stage, but it does not guarantee a globally optimal solution. Although it usually provides a sub-optimal solution, its main advantage lies in its ability to solve problems much faster, often within polynomial time.

Below, we describe our suggested Greedy algorithm for solving the optimal model (1).

4.1. Greedy Algorithm

The greedy algorithm for solving model (1) begins by forming a group of bundles $\varphi = (p, i)$ for each period where p and i stand for the items and demand points respectively. It then prioritizes these bundles based on the maximum value potential function ρ. In the subsequent section, we will discuss the potential function and how it is designed. The algorithm is designed to determine the flow of each warehouse f for each bundle φ, denoted as $\tilde{y}_{f\varphi t}$, from which the values of other variables $\tilde{x}_{fpt}, \tilde{q}_{fpt}$ are derived. $\tilde{x}_{fpt}, \tilde{q}_{fpt}$ are the decision variables in the model (1) which are defined as the value of initial stocking, and the inventory level of warehouse f, respectively. Here, the tilde ($\sim$) is used for each variable to indicate that the solution is obtained through heuristic, distinguishing it from the optimal solution. Additionally, the algorithm employs a backward strategy to address the dynamic nature of model. Hence, starting from the last period, the bundles are arranged in decreasing order of potential function ρ, the highest priority bundle is picked, and among different warehouses, the one with minimum cost, denoted as f^*, is selected for this particular bundle φ. Subsequently, the value of $\tilde{y}_{f\varphi t}$ is determined by the minimum of $d_{\varphi t}$ and the current capacity of warehouse f^*, i.e., $\tilde{y}_{f^*\varphi t} = \min\left\{d_{\varphi t},\ \left(\varpi_{f^*t} - \overline{w}_{f^*t}\right)/v_p\right\}$ where $\overline{w}_{ft}$ is the auxiliary variable representing the utilized capacity of f^* . If $\tilde{y}_{f^*\varphi t} = d_{\varphi t}$, warehouse f^* can fully satisfy $d_{\varphi t}$, otherwise, the model proceeds to the next cheapest warehouses until $d_{\varphi t}$ is fully satisfied.

Once $\tilde{y}_{f^*\varphi t}$ has been established for every bundle φ, the algorithm now seeks to find out the values of $\tilde{x}_{fpt}$ and $\tilde{q}_{fpt}$. Up until to this point, it has been identified which warehouse is responsible for fulfilling the bundle $\varphi = (p, i)$. To determine how to stock the warehouses f with each item p, i.e., the value of $\tilde{x}_{fpt}$, we define new bundle $\varphi_1 = (f^*, p)$. This approach is adopted because variables of $\tilde{x}_{fpt}$ and $\tilde{q}_{fpt}$ depend on the warehouse f and the item p, but not on the demand point i. By introducing φ_1, we simplify the analysis as it allows us to focus on the relationships between warehouses and items without considering individual demand points directly.

The bundles φ_1 are arranged in decreasing order based on the total value of $\tilde{y}_{\varphi_1 it}$ for all demand points i, and inventory level of each warehouse for the current period, i.e., $\tilde{q}_{\varphi_1 t}$. Given that the algorithm starts from the last period and $\tilde{q}_{\varphi_1 t}$ represents the inventory level at the end of the period, the value of $\tilde{q}_{\varphi_1 t}$ is initially zero, meaning that $\tilde{q}_{\varphi_1 T} = 0$.

The value of $\tilde{x}_{\varphi_1 t}$ is determined based on the minimum of $\left(\sum_i \tilde{y}_{\varphi_1 it} + \tilde{q}_{\varphi_1 t}\right)$ and total current supply $(\alpha_{pt} - \bar{\alpha}_{pt})$ for item p, i.e., $\tilde{x}_{\varphi_1 t} = min\left\{\left(\sum_i \tilde{y}_{\varphi_1 it} + \tilde{q}_{\varphi_1 t}\right), \alpha_{pt} - \bar{\alpha}_{pt}\right\}$; where α_{pt} stands for the supply of item p, and $\bar{\alpha}_{pt}$ is the utilized portion of α_{pt}. If the supply is insufficient such that $\tilde{x}_{\varphi_1 t} \neq \left(\sum_i \tilde{y}_{\varphi_1 it} + \tilde{q}_{\varphi_1 t}\right)$, the remaining $\tilde{x}_{\varphi_1 t}$ will be fulfilled by inventory from previous period, $\tilde{q}_{\varphi_1, t-1}$. This ensures that the flow conservation equation $\tilde{q}_{\varphi_1, t-1} = \sum_i \tilde{y}_{\varphi_1 it} - \tilde{x}_{\varphi_1 t} + \tilde{q}_{\varphi_1 t}$ holds, maintaining the balance between total inflow and total outflow for each bundle φ_1. Once these three variables are established, the model proceeds to the next iteration, period $t - 1$, and keeps repeating the same procedure until the variables are determined for all periods.

The summary of algorithm is as follows:

Table 2. The summary of greedy algorithm for solving optimal model

Step 1	Define N as the total number of (P, I) combinations, set $t = T$, $\tilde{q}_{\varphi_1 T} = 0$
Step 2	Set $\bar{w}_{ft} = 0, \bar{\alpha}_{pt} = 0$
Step 3	Define $\varphi = (\mathrm{p}, i)$, sort them in decreasing order of potential function ρ
Step 4	Set $\varphi = 1$ (represents the bundle (p, i) with the highest potential function)
Step 5	Choose a warehouse with the least cost where $f^* = Argmin\left\{\left(r_{pft} + \mathrm{h}_{pft} + \beta_{fipt}\right)\right\}$ If $\bar{w}_{f^*t} + v_p \leq \varpi_{f^*t}$: Assign bundle φ to the warehouse f^*

	$\tilde{y}_{f^*\varphi t} = \min\{d_{\varphi t},\ (\varpi_{f^*t} - \bar{w}_{f^*t})/v_p\}$ $\bar{w}_{f^*t} = \bar{w}_{f^*t} + v_p * \tilde{y}_{f^*\varphi t}$ $d_{\varphi t} = d_{\varphi t} - \tilde{y}_{f^*\varphi t}$ Else move to Step 6
Step 6	While $d_{\varphi t} > 0$, then $f^* = f^* + 1$, move to step 5
Step 7	Set $\varphi = \varphi + 1$, move to step 5
Step 8	Define $\varphi_1 = (f^*, \text{p})$, sort them in decreasing order of $\left(\sum_i \tilde{y}_{\varphi_1 it} + \tilde{q}_{\varphi_1 t}\right)$
Step 9	If $\bar{\alpha}_{pt} \leq \alpha_{pt}$: $\tilde{x}_{\varphi_1 t} = min\left\{\left(\sum_i \tilde{y}_{\varphi_1 it} + \tilde{q}_{\varphi_1 t}\right), \alpha_{pt} - \bar{\alpha}_{pt}\right\}$ $\tilde{q}_{\varphi_1, t-1} = \sum_i \tilde{y}_{\varphi_1 it} - \tilde{x}_{\varphi_1 t} + \tilde{q}_{\varphi_1 t}$ $\bar{\alpha}_{pt} = \bar{\alpha}_{pt} + \tilde{x}_{\varphi_1 t}$ Else $\tilde{x}_{\varphi_1 t} = 0$, $\tilde{q}_{\varphi_1, t-1} = \sum_i \tilde{y}_{\varphi_1 it} + \tilde{q}_{\varphi_1 t}$
Step 10	Set $\varphi_1 = \varphi_1 + 1$, move to step 9
Step 11	$\bar{w}_{f,t-1} = \bar{w}_{f,t-1} + v_p * \tilde{q}_{\varphi_1, t-1}$
Step 12	Set $t = t - 1$, move to step 2

In the next section, we describe the performance of the above algorithm.

4.2. Potential function

Before delving into the performance of the greedy algorithm, let's discuss the selection criterion used for bundles, referred to as the potential function. This criterion is defined as the weighted average cost per unit volume for a bundle φ, denoted by $\rho = \frac{(\sum_f \vartheta_{pf}) * \text{d}_\varphi}{v_p}$, where φ represents the bundle (p, i), d_φ is the demand for bundle φ, and ϑ_{pft} is the cost of assigning bundle (p, i) to warehouse $f \in F$ where ϑ_{pft} is calculated as $\vartheta_{pft} = r_{pft} + h_{pft} + \beta_{fipt}$. In the following discussion, we aim to justify the rationale behind using this criterion.

Let's first consider the potential function excluding $\sum_f \vartheta_{pf}$, defined as the ratio of demand to volume, denoted by $\rho = \frac{\text{d}_\varphi}{v_p}$. When employing the greedy algorithm, bundles are arranged in descending order of this efficiency measure. The algorithm selects the bundle with the highest efficiency and assigning it to the most cost-effective warehouses to meet the demand for that bundle, while taking capacity restrictions into account. Utilizing this approach can yield a feasible solution, but in certain special cases, the outcomes may be unsatisfactory. For instance, consider a scenario with two bundles, each with volumes $v_1 = 1$, $v_2 = 2$ and equal demands of 1. These bundles need to be allocated to two warehouses with capacities $\varpi_1 = \varpi_2 = 2$. The associated matrix ϑ_{pft} for a specific period t is as follows:

$$\vartheta_{pft} = \begin{bmatrix} 2 & 4 \\ 1 & M' \end{bmatrix}$$

Comparing the efficiencies of the bundles, the Greedy algorithm assigns bundle 1 to warehouse 1 and bundle 2 to warehouse 2 with a value of 2+ M', while the optimal solution is 5. For a suitably large selection of M', any relative performance guarantee for greedy can be forced to be arbitrarily close to infinity. As a result, to avoid this pathological special case, we must modify our sorting criteria. The aforementioned issue can be avoided if we swap out $\frac{\mathrm{d}_\varphi}{v_p}$ with $\frac{(\sum_f \vartheta_{pf}) * \mathrm{d}_\varphi}{v_p}$, i.e., sort the bundles according to the weighted average cost per unit volume rather than simply the demand. Thus, if we repeat the algorithm for the above instance with the new potential function, the solutions provided by Greedy and the optimal solution would be the same, and the gap would be zero as M' approaches a suitably large number.

4.3. Greedy performance

The benefit of using the greedy algorithm is twofold: the approach is straightforward to implement. In fact, the algorithm follows a clear set of steps, enabling it to handle an NP-hard dynamic programming problem through an iterative structure. Additionally, the greedy generally has lower computational complexity making it faster and more suitable for large-scale problems where speed is critical. However, this is not the case for other optimal algorithms. For instance, the branch and bound algorithm requires to do branching on non-integer variables and solving a linear programming model at each iteration. In contrast, the greedy algorithm makes a series of locally optimal choices at each step (e.g., selecting the warehouse with the least cost) without backtracking. Thus, the algorithm can quickly provide a feasible solution to the optimal model.

However, there are some weaknesses in the algorithm. The solution obtained by the greedy method is sub-optimal. As illustrated in Section 4.2, the algorithm's effectiveness heavily relies on the potential function ρ. This function aims to minimize the worst-case scenario by focusing on bundles with the highest cost. Consequently, the algorithm prioritizes assigning these costly bundles, which can lead to suboptimal outcomes. This assignment strategy might result in bundles with lower potential functions being assigned to more expensive warehouses, even though they could have been assigned to cheaper ones. As a result, this can increase the overall cost, leading to a higher optimality gap and a less efficient solution. Consider the following instance for a specific one period. Let $\varphi = (p, i)$ and F denote the respective sets of bundles and warehouses. We define P_h, and P_l as sets of high-potential and low-potential bundles where the unit assignment cost is defined as follows:

$$\vartheta_{\varphi f} = \begin{cases} 1 & if\ f = 1\ and\ \varphi \in P_l \\ M' & \forall f \in F \backslash \{1\}\ and\ \varphi \in P_l \\ M' & \forall f \in F\ and\ \varphi \in P_h \end{cases}$$

The bundles in P_h have higher potential function than those in P_l, i.e., $\rho_h > \rho_l$. The total number of bundles are N which includes N_h for the number of P_h, and N_l for the number of P_l where $N = N_h + N_l$. Bundles in P_h and P_l might be different in volume and demand; however, the following relationship holds:

$$\sum_{\varphi \in P_l} v_p * d_\varphi \leq \sum_{\varphi \in P_h} v_p * d_\varphi$$

There are M number of warehouses where the capacity of the first one is equal to $\varpi_1 = \sum_{\varphi \in P_l} v_p * d_\varphi$.

The greedy algorithm prioritizes bundles from P_h over those from P_l due to their higher potential functions. As a result, the algorithm assigns the first bundle from P_h to the minimum cost warehouse and continues to assign bundles from P_h to this warehouse until its capacity is exhausted. Given the assumption that the capacity of the first warehouse ($\varpi_1 = \sum_{\varphi \in P_l} v_p * d_\varphi$) is insufficient to accommodate all bundles from P_h, remaining bundles from P_h and all bundles from P_l are assigned to subsequent warehouses with unit cost M'. Consequently, the cost of the greedy approach is $Z(S^G) = M' \sum_{\varphi \in P_h} d_\varphi + M' \sum_{\varphi \in P_l} d_\varphi$. In contrast, the optimal model assigns all bundles from P_l to the first warehouse with unit cost 1, which can accommodate them, and assigns bundles from P_h to the remaining warehouses. This results in a cost $Z(S^*) = \sum_{\varphi \in P_l} d_\varphi + M' \sum_{\varphi \in P_h} d_\varphi$. Therefore, the relative performance of the greedy for this specific example will be greater then 1, illustrating how the algorithm's prioritization of high-potential bundles can lead to suboptimal solutions.

Moreover, although the greedy algorithm performs well in a single period, it does not necessarily extend its effectiveness to a multi-period programming model. The algorithm addresses the dynamic nature of the problem by breaking it down into |T| individual iterations. However, the greedy approach does not account for the interactions between periods when solving a multi-period problem. Each period is treated independently without considering the effect of previous periods, whereas optimal algorithms consider all periods simultaneously to determine the values of decision variables. This lack of integration across periods can limit the effectiveness of the greedy approach for multi-period planning.

The algorithm's strategy to determine the inventory level of warehouses is suboptimal. In step 9, after sorting warehouses in decreasing order of $\left(\sum_i \tilde{y}_{\varphi_1 it} + \tilde{q}_{\varphi_1 t}\right)$, the algorithm selects the first bundle $\varphi_1 = (f^*, p)$. The value of $\tilde{x}_{\varphi_1 t}$ is then determined based on the minimum of $\left(\sum_i \tilde{y}_{\varphi_1 it} + \tilde{q}_{\varphi_1 t}\right)$ and total current supply $(\alpha_{pt} - \bar{\alpha}_{pt})$ for each item p. This approach results in an unbalanced inventory distribution across warehouses. Initially selected warehouses end up with $\tilde{q}_{\varphi_1, t-1} = 0$, while those chosen later will accumulate a large value $\tilde{q}_{\varphi_1, t-1} = \left(\sum_i \tilde{y}_{\varphi_1 it} + \tilde{q}_{\varphi_1 t}\right)$. This imbalance can lead to increased storage costs for some warehouses, thereby raising the overall cost of the greedy algorithm in certain cases.

Although the greedy is an efficient method to cope with the NP-hard nature of model (1), it does not resolve the issue of computational intractability for large-scale problems. In fact, the computation time of greedy is $O(TNF)$ where T is the number of periods, N is the number of bundles (p, i) and F is the number of warehouses. The algorithm requires enumerating all possible combinations of (p, i) throughout each iteration. Given that the optimal model (1) is large scale, this results in an extremely large number of bundles φ. Therefore, we must develop a tractable algorithm which creates a trade-off between the running time and approximation ratio. In Section 5. we describe our new strategy to address this issue.

5. Aggregation algorithm

The greedy algorithm described in Section 4.1 does not completely resolve the issue of computational intractability for large-scale problems. With a computation time of $O(TNF)$, the algorithm necessitates enumerating all combinations of bundles (p, i) in each iteration, leading to a significant number of bundles N in a large-scale model. Therefore, a more tractable algorithm is needed to create a trade-off between the running time and the approximation ratio. Here, we propose a new methodology focused on an aggregation approach, which utilizes the greedy algorithm as a subroutine. Using this methodology, we can further reduce the computation time required to solve a large-scale model, albeit at the expense of a larger optimality gap.

This section will explain the aggregation approach employed in this work. As mentioned previously, we conduct aggregation on three sets: demand zones, fulfillment centers, and items. The rationale behind this decision comes from the original model's nature as a NP-hard problem, wherein computation time exponentially increases with problem dimensions. While the greedy algorithm can somewhat reduce computation time, it is insufficient for large-scale instances due to its dependence on the number of bundles. As the problem size increases, the number of bundles grows significantly, making the greedy approach impractical for handling large instances. Therefore, a more tractable algorithm is necessary to efficiently manage these large-scale problems.

Reducing the model's dimensionality is an effective strategy to make the model tractable from both computational and theoretical perspectives. However, it's essential to acknowledge that aggregation and disaggregation method results in errors into the decision-making process and consequently suboptimality of optimization model. We will delve into this issue in Section 5.4. In this section, our focus is on elaborating the aggregation mechanism.

Handling a substantial number of items complicates the management of product for inventory control, particularly as transaction volumes increase. This challenge can be addressed through a more effective approach, employing an aggregation method to categorize all products into manageable clusters based on their similarities. The values of these various groups can then be estimated and assessed by appropriate techniques. Therefore, the aggregation approach will facilitate the resource management decision-making process by reducing the dimensionality of the problem (Gustriansyah et al., 2020). Here, we perform aggregation on the three sets of items, demand points, and warehouses, based on the historical data from all demand points for various items. These aggregated groups remain constant throughout the planning horizon.

Let the items $P = \{1, \dots, l\}$ be a set of l items. Suppose P is clustered into C groups, where the set of indices in the c-th group is denoted P_c such that $\bigcup_{c=1}^{C} P_c = \{1, \dots, l\}$ and $P_c \cap P_{c'} = \emptyset \ \forall c$ and $c', c \neq c'$. We also have clustering on demand points and fulfilment points. Let $I = \{1, \dots, n\}$ be a set of n demand points, and $F = \{1, \dots, m\}$ be a set of m fulfilment points. Suppose I and F are grouped into set of clusters $I_k, k = 1, \dots, K$ and $F_w, w = 1, \dots, W$, such that $I_k \cap I_{k'} = \emptyset \ \forall k$, $\bigcup_{k=1}^{K} I_k = \{1, \dots, n\}$, $F_w \cap F_{w'} = \emptyset \ \forall w$ and $\bigcup_{w=1}^{W} F_w = \{1, \dots, m\}$. We employ a well-known clustering algorithm, k-means, to efficiently perform aggregation. K-means is a centroid-based algorithm which aims to aggregate data into a predetermined number of groups while minimizing the sum of distances between the points and the centroid of their particular cluster. The clustering procedure is performed independently for each set of items, warehouses, and demand points. This process involves the consideration of suitable clustering features, such as the geographical locations of warehouses and demand sites, as well as item volume, cost, and demand, among others. Having completed the clustering process, we will now proceed to determine the parameters of the aggregation model. The following explains how to find out the aggregated model's parameters.

To determine the model parameters for the reduced problem, we need to use a method which determines weights that reflect the relative importance of the parameters in the original problem and using these weights to derive parameters for the reduced problem. To do so, we introduce weight vectors $g^{wc} = \left\{ {g_r}_{fp}^{wc} \right\}, c = 1, \dots, C, w = 1, \dots, W,$ such that $\sum_{f \in F_w} \sum_{p \in P_c} {g_r}_{fp}^{wc} = 1$ for $c = 1, \dots, C$ and $w = 1, \dots, W$ to convert r_{pft} to the centroid aggregation r_{cwt}. The nonnegative weight vectors ${g_h}_{fp}^{wc}$, ${g_\beta}_{fip}^{wkc}$ are also defined to transfer parameters h_{pft} and β_{fipt} to their aggregation format in the reduced model, respectively. These weight vectors must satisfy the normalization condition:

$$\sum_{f \in F_w} \sum_{p \in P_c} {g_h}_{fp}^{wc} = 1 \text{ for } w = 1, \dots, W, c = 1, \dots, C$$

$$\sum_{i \in I_k} \sum_{f \in F_w} {g_\beta}_{fip}^{wkc} = 1 \text{ for } w = 1, \dots, W, k = 1, \dots, K, c = 1, \dots, C$$

One way to compute these vectors can be a fixed weight approach which involves a convex weighting of the elements inside a cluster, and if elements are homogenous, we can replace it with equal weighting. For instance, each element of vector g^{wc} can be calculated as $\frac{1}{|P_c||F_w|}$.

This type of aggregation is similar to fixed-weight combination approach can be seen as an example of column aggregation, when three sets of items, demand and fulfilment points are aggregated simultaneously, we have a combination of both columns (variables) and rows (constraints) aggregation.

5.1. Aggregation formulation

The set of clusters $K = \{I_k | k = 1, \dots, \mathrm{K}\}$, $W = \{F_w | w = 1, \dots, \mathrm{W}\}$, $C = \{P_c | c = 1, \dots, \mathrm{C}\}$, and the collection of weight vectors completely define an aggregated problem (6). In order to obtain a concise formulation of the aggregate problem, we define:

$\bar{d}_{ckt} = \sum_{p\in P_c, i\in I_k} d_{pit}$, $\quad \bar{\alpha}_{ct} = \sum_{p\in P_c} \alpha_{pt}$, $\quad \bar{\varpi}_{wt} = \sum_{f\in F_w} \varpi_{ft}$, $\quad \bar{v}_c = \sum_{p\in P_c} g^c_p v_p$, $\quad \bar{r}_{cwt} = \sum_{f\in F_w} \sum_{p\in P_c} g_{r}{}^{wc}_{fp} r_{pft}$, $\quad \bar{h}_{cwt} = \sum_{f\in F_w} \sum_{p\in P_c} g_{h}{}^{wc}_{fp} h_{pft}$, $\quad \bar{\beta}_{wkc} = \sum_{f\in F_w} \sum_{i\in I_k} g_{\beta}{}^{wkc}_{fip} \beta_{fip}$

The following is then an aggregated version of original problem (1):

$$\bar{Z}_{AGG} = \begin{matrix} min \\ \bar{x}, \bar{q}, \bar{y} \end{matrix} \left(\sum_{c,w,t} \bar{r}_{cwt} \bar{x}_{cwt} + \sum_{c,w,t} \bar{h}_{cwt}\, \bar{q}_{cwt} + \sum_{c,w,k,t} \bar{\beta}_{wkct} \bar{y}_{wkct} \right) \qquad (2a)$$

Subject to:

$$\sum_{w} \bar{y}_{wkct} \geq \bar{d}_{ckt} \qquad \forall\, c, k, t \qquad (6b)$$

$$\bar{q}_{cwt} = \bar{q}_{cw,t-1} + \bar{x}_{cwt} - \sum_{k} \bar{y}_{wkct} \qquad \forall\, c, w, t \qquad (6c)$$

$$\sum_{w} \bar{x}_{cwt} \leq \bar{\alpha}_{ct} \qquad \forall c, t \qquad (6d)$$

$$\sum_{c} \bar{v}_c\, \bar{q}_{cw,t-1} + \sum_{c} \bar{v}_c \bar{x}_{cwt} \leq \bar{\varpi}_{wt} \qquad \forall\, w, t \qquad (6e)$$

$$\bar{x}_{cwt},\ \bar{q}_{cwt}, \bar{y}_{wkct} \geq 0, Integer \qquad (6f)$$

This problem represents a reduced form of the original problem (1), whereby the original sets I, F and P are reduced to K, W and C, respectively. Therefore, in the worst-case scenario, the time required to solve the aggregated problem grows exponentially with respect to the size of the clusters rather than the original input size. The resulting policy from the aggregated model is denoted by $\bar{x}^*{}_{cwt}$, $\bar{q}^*{}_{cwt}$, and $\bar{y}^*{}_{wkct}$.

5.2. Disaggregation steps and formulations

In this section, we detail the process of converting the aggregated solution of problem (6) into a feasible solution for the original problem (1). This process involves breaking down the aggregated decisions into more detailed, and individual decisions, corresponding to the specific variables in the original problem.

The simplest way to obtain a feasible solution to the original problem (1) from an aggregated solution (6) is to use the fixed-weight disaggregation. Denoting $\hat{x}_{fpt}$, $\hat{q}_{fpt}$, and $\hat{y}_{fpit}$ as the solutions directly derived from the aggregated problem (6), under the fixed-weight disaggregation approach, they would be represented as follows: $\hat{x}_{fpt} = g_{r_{fp}}^{wc}\bar{x}_{cwt}$, $\hat{q}_{fpt} = g_{h_{fp}}^{wc}\bar{q}_{cwt}$, and $\hat{y}_{fpit} = g_{\beta_{fip}}^{wkc}\bar{y}_{wkct}$, respectively. However, in the integer programming case, fixed weight disaggregation does not in general result in an integer solution to the original problem. Thus, in our case, problem-specific procedures can be applied which ensures feasibility. In the following, we have defined a three-step hierarchal disaggregation approach as follows:

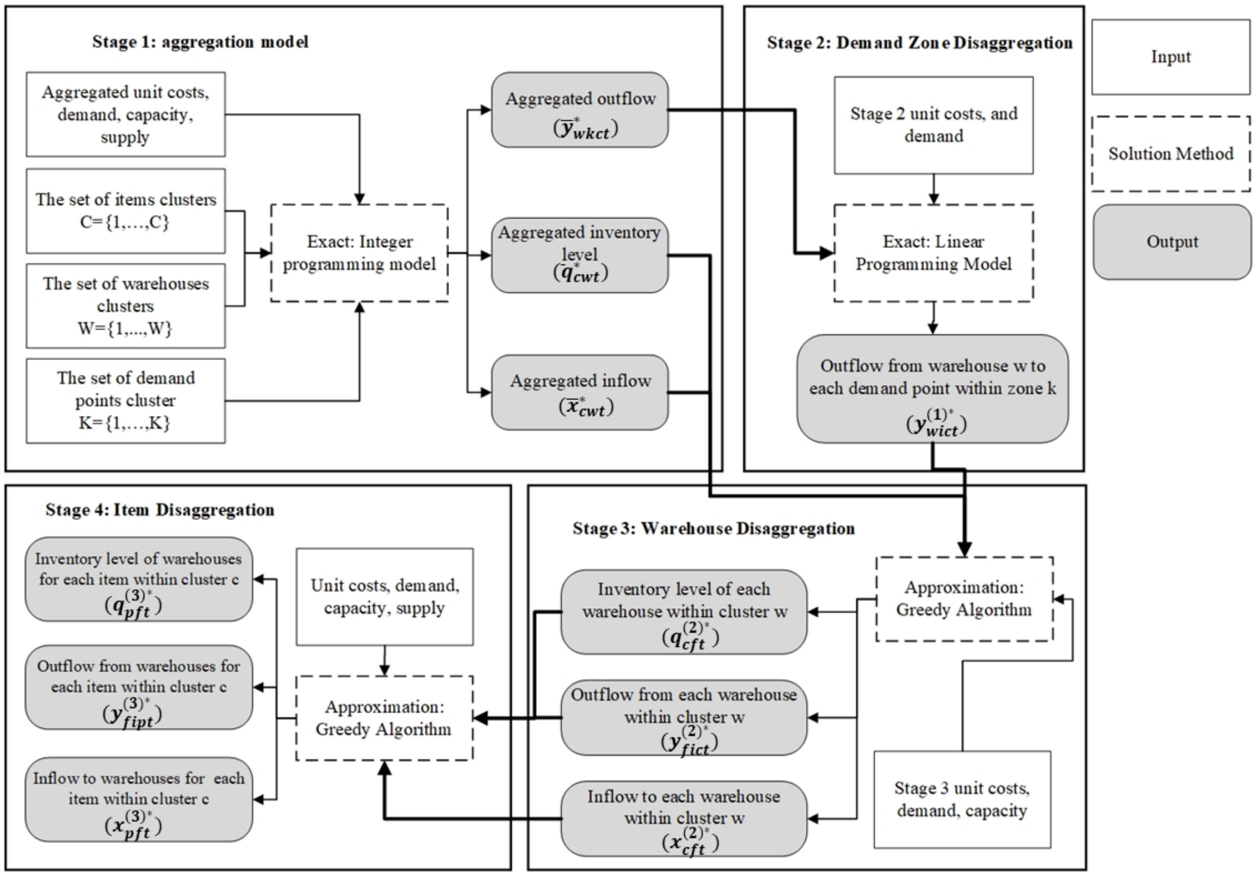
- Demand zones disaggregation

  The goal of demand zones disaggregation stage is to determine the flow between each aggregated warehouse w and each demand point i within zone k, while considering the optimal solution of the aggregated model, represented as $(\bar{x}^*_{cwt}, \bar{q}^*_{cwt}, \bar{y}^*_{wkct})$. Since the clustering process ensures no intersection between points of different clusters, each demand point is assigned to only one zone. Additionally, the flow between the group of warehouses w and each zone k is predetermined; as obtained by solving the aggregated model (6) which is denoted as $\bar{y}^*_{wkct}$. As a result, we can perform the process of disaggregation for each zone k independently without impacting the final solution.
- Warehouses disaggregation

Once the flow between each group of warehouse w and each demand zone i is specified, we proceed to disaggregate the warehouses. To do so, we require the solutions of both the aggregation model (6) and the previous stage, i.e., $\left(\bar{x}^*_{cwt}, \bar{q}^*_{cwt}, y^{(1)^*}_{wict}\right)$. The amount of inflow, outflow, and inventory level of each fulfillment center f within the aggregated warehouse w are the decision variables must be determined in this stage. Similar to the previous logic, the solution space of this stage is divided into distinct subspaces, each representing a specific cluster w. Therefore, this allows us to perform disaggregation for each group w individually.

- Items disaggregation

The decision variables for the original problem have been almost estimated up until this point, albeit with values obtained for the aggregated items. Now, our task is to specify the inventory level, inflow to fulfillment center f, and the amount of shipment between fulfillment center f and demand zone i for each specific item p. Thus, the final stage of disaggregation is conducted in a similar manner, with the input parameters being the solutions obtained from the previous step, represented as $\left(x^{(2)^*}_{cft}, q^{(2)^*}_{cft}, y^{(2)^*}_{fict}\right)$. Similar to the preceding stages, this stage can be disaggregated individually, as each item is assigned to a single cluster and the solution of each cluster c is predetermined from the previous stage.



Stage 1: aggregation model
Aggregated unit costs, demand, capacity, supply
The set of items clusters C={1,…,C}
The set of warehouses clusters W={1,...,W}
The set of demand points cluster K={1,…,K}
Exact: Integer programming model
Aggregated outflow ($\bar{y}^*_{wkct}$)
Aggregated inventory level ($\bar{q}^*_{cwt}$)
Aggregated inflow ($\bar{x}^*_{cwt}$)
Stage 2: Demand Zone Disaggregation
Stage 2 unit costs, and demand
Exact: Linear Programming Model
Outflow from warehouse w to each demand point within zone k ($y^{(1)^*}_{wict}$)
Input
Solution Method
Output
Stage 4: Item Disaggregation
Inventory level of warehouses for each item within cluster c ($q^{(3)^*}_{pft}$)
Outflow from warehouses for each item within cluster c ($y^{(3)^*}_{fipt}$)
Inflow to warehouses for each item within cluster c ($x^{(3)^*}_{pft}$)
Unit costs, demand, capacity, supply
Approximation: Greedy Algorithm
Stage 3: Warehouse Disaggregation
Inventory level of each warehouse within cluster w ($q^{(2)^*}_{cft}$)
Outflow from each warehouse within cluster w ($y^{(2)^*}_{fict}$)
Inflow to each warehouse within cluster w ($x^{(2)^*}_{cft}$)
Approximation: Greedy Algorithm
Stage 3 unit costs, demand, capacity

Figure 2. The detailed workflow of aggregation and disaggregation stages

Now, we will delve into the details of the optimization models corresponding to each of the previously described stages. We will provide a comprehensive explanation of the optimization models that utilized to disaggregate demand zones, warehouses, and items, respectively.

5.2.1. Optimal Disaggregation of demand zones

Considering the optimal solution of the aggregated model, represented as $(\bar{x}^*_{cwt}, \bar{q}^*_{cwt}, \bar{y}^*_{wkct})$, the optimal disaggregation of $\bar{y}^*_{wkct}$ can be achieved by solving subproblems for each $k = 1, \dots, K$. Each subproblem k includes only the portion of the original problem that coincides with the members of each I_k. The objective function value for disaggregation of all demand zones will be equal to $Z^D_{Diss} = \sum_{k \in K} Z^D_{Diss_k}$ where $Z^D_{Diss_k}$ stands for the objective value of disaggregating demand zone k.

To determine the disaggregation of demand zones, we define the decision variable $y^{(1)}_{wict}$ for each subproblem k, representing the quantity of aggregated items c transferred from aggregated warehouses w to demand point i within zone k. Additionally, the parameter $\beta^{(1)}_{wict}$ is defined to denote the unit cost of transporting aggregated items c from aggregated warehouses w to demand point i within zone k. This parameter is calculated as $\beta^{(1)}_{wict} = \sum_{p\in P_c} \sum_{f\in F_w} g_{\beta}{}^{wc}_{fp} \beta_{fipt}$. Therefore, we can formulate the fist stage disaggregation model (7) in the following way:

$$Z^{D}_{Diss_k} = \frac{min}{y^{(1)}} \sum_{w,i\in I_k,c,t} \beta^{(1)}_{wict} * y^{(1)}_{wict} \tag{3a}$$

Subject to:

$$\sum_{w} y^{(1)}_{wict} \geq \bar{d}_{cit} \qquad \forall\ i \in I_k, c, t \tag{7b}$$

$$\sum_{i\in I_k} y^{(1)}_{wict} \leq \bar{y}^{*}_{wkct} \qquad \forall\ w, c, t \tag{7c}$$

$$y^{(1)}_{wict} \in Z^{+} \tag{7d}$$

Where the objective function (7a) minimizes total transportation costs for each subproblem k. The constraint set (7b) ensures the fulfillment of the demand for aggregated item c at each demand point i within each zone k. The constraint set (7c) guarantees that total quantity assigned to all demand points i within zone k does not surpass the aggregated value $\bar{y}^{*}_{wkct}$. The last constraint (7d) determines the type of the decision variable $y^{(1)}_{wict}$.

5.2.2. Optimal Disaggregation of warehouses

Considering the solution of the first stage disaggregation denoted by $y^{(1)^*}_{wict}$, and the solution of the aggregated model, i.e., $(\bar{x}^{*}_{cwt}, \bar{q}^{*}_{cwt})$, we can formulate the optimization model (8) for warehouses disaggregation by defining new sets of decision variables and parameters. These are outlined below, and the model can be solved independently for each subproblem $w = 1,..,W$. The objective function value of this stage will be equal to $Z^{w}_{Diss} = \sum_{w\in W} Z^{W}_{Diss_w}$ where $Z^{W}_{Diss_w}$ represents the objective value of solving the subproblem w.

Table 3: The lists of sets and notations used in the warehouse disaggregation problem

Sets

I_w	The subset of demand points supplied from the aggregated warehouse w defined as: $I_w = \{i \in I \mid y^{(1)^*}_{wict} > 0 \;\; \forall\, c \in C, t \in T\}$
Parameters	
$\beta^{(2)}_{fict}$	The unit of shipment cost from the fulfilment centre $\mathrm{f} \in \mathrm{F_w}$ to demand zone $i \in I_w$ for cluster c in period t
$h^{(2)}_{cft}$	The unit of storage cost of cluster c in fulfilment center $\mathrm{f} \in \mathrm{F_w}$ in period t
$r^{(2)}_{cft}$	The receiving cost of cluster c in the fulfilment center $\mathrm{f} \in \mathrm{F_w}$ in period t
Variables	
$x^{(2)}_{cft}$	Quantity of cluster c shipped to fulfilment center $\mathrm{f} \in \mathrm{F_w}$ in period t
$y^{(2)}_{fict}$	Quantity of cluster c shipped from fulfilment center $\mathrm{f} \in \mathrm{F_w}$to demand zone $i \in I_w$ in period t
$q^{(2)}_{cft}$	The inventory level of fulfilment center $\mathrm{f} \in \mathrm{F_w}$ for cluster c in period t

The parameters $\beta^{(2)}_{fict}, h^{(2)}_{cft}, and\ r^{(2)}_{cft}$ are obtained as follows:

$$\beta^{(2)}_{fict} = \sum_{p \in P_c} g^c_p \beta_{fipt} \qquad \forall\, f \in F_w, i \in I_w, \mathrm{t} \in \mathrm{T}$$

$$h^{(2)}_{cft} = \sum_{p \in P_c} g^c_p h_{fpt} \qquad \forall\, f \in F_w, \mathrm{t} \in \mathrm{T}$$

$$r^{(2)}_{cft} = \sum_{p \in P_c} g^c_p h_{fpt} \qquad \forall\, f \in F_w, \mathrm{t} \in \mathrm{T}$$

Each sub-problem w takes the following form:

$$Z^W_{Diss_w} = \min_{x^{(2)}, q^{(2)}, y^{(2)}} \left(\sum_{f \in F_w, c, t} r^{(2)}_{cft} * x^{(2)}_{cft} + \sum_{c, f \in F_w, t} h^{(2)}_{cft} * q^{(2)}_{cft} + \sum_{f \in F_w, i \in I_w, c, t} \beta^{(2)}_{fict} * y^{(2)}_{fict} \right) \tag{4a}$$

$subject\ to$:

$$\sum_{f \in F_w} y_{fict}^{(2)} = y_{wict}^{(1)^*} \qquad \forall\, c, i \in \mathrm{I_w}, t \qquad (8b)$$

$$q_{cft}^{(2)} = q_{cf,t-1}^{(2)} + x_{cft}^{(2)} - \sum_{i \in I_w} y_{fict}^{(2)} \qquad \forall\, c, f \in F_w, t \qquad (8c)$$

$$\sum_{f \in F_w} x_{cft}^{(2)} = \bar{x}^*{}_{cwt} \qquad \forall\, c, t \qquad (8d)$$

$$\sum_{c} \bar{v}_c q_{cf,t-1}^{(2)} + \sum_{c} \bar{v}_c x_{cft}^{(2)} \leq \varpi_{ft} \qquad \forall\, f \in F_w, t \qquad (8e)$$

$$\sum_{f \in F_w} q_{cft}^{(2)} = \bar{q}_{cwt}^{*} \qquad \forall\, c, t \qquad (8f)$$

$$x_{cft}^{(2)}, q_{cft}^{(2)}, y_{fict}^{(2)} \in Z^+ \qquad (8g)$$

The objective function (8a) minimizes the total receiving, storage, and transportation costs for each subproblem w. The constraint set (8b) guarantees that total flow from all fulfilment canters within the cluster w must match the flow leaving the cluster w for a specific demand point i and aggregated item c. The constraint (8c) represents the flow conservation. The constraint set (8d) ensures that the total quantity assigned to all fulfilment centres within cluster w is equal to the total aggregated value. Additionally, the remaining constraints include warehouse capacity constraint (8e), upper bound for the inventory level of warehouses (8f), and (8g) defines defines the domain and type of decision variables.

5.2.3. Optimal Disaggregation of items

Here, we formulate an optimization model similar to warehouse disaggregation for this stage using the solutions from the previous stage, denoted as $\left(x_{cft}^{(2)^*}, q_{cft}^{(2)^*}, y_{fict}^{(2)^*}\right)$. The disaggregated solution can be achieved by solving the subproblems for each $c = 1, \dots, C$. The overall objective function value after disaggregating all clusters $c = 1, \dots, C$ will be equal to $Z_{Diss}^{p} = \sum_{c \in C} Z_{Diss_c}^{p}$ where $Z_{Diss_c}^{p}$ represents the objective value of solving the subproblem c. We introduce new sets of decision variables including $x_{pft}^{(3)}, q_{pft}^{(3)}$, and $y_{fipt}^{(3)}$ to develop the optimization model (9) for each subproblem c. Since the set of parameters are identical to those in the original problem, hence won't go into additional detail here. Each sub-problem c takes the following form:

$$Z_{Diss_c}^{P} = \min_{x^{(3)}, q^{(3)}, y^{(3)}} \left(\sum_{f, p \in P_c, t} r_{pft} * x_{pft}^{(3)} + \sum_{p \in P_c, f, t} h_{pft} * q_{pft}^{(3)} + \sum_{p \in P_c, i, f, t} \beta_{fipt} * y_{fipt}^{(3)} \right) \quad (5a)$$

$Subject\ to$:

$$\sum_{p \in P_c} y_{fipt}^{(3)} = y_{fict}^{(2)^*} \qquad \forall\, f, i, t \qquad (9b)$$

$$\sum_{f} y_{fipt}^{(3)} \geq d_{pit} \qquad \forall\, p \in P_c, i, t \qquad (9c)$$

$$q_{pft}^{(3)} = q_{pf,t-1}^{(3)} + x_{pft}^{(3)} - \sum_{i} y_{fipt}^{(3)} \qquad \forall\, p \in P_c, f, t \qquad (9d)$$

$$\sum_{p \in P_c} v_p q_{pf,t-1}^{(3)} + \sum_{p \in P_c} v_p x_{pft}^{(3)} \leq \bar{v}_c \left(x_{cft}^{(2)^*} + q_{cf,t-1}^{(2)^*} \right) \qquad \forall\, f, t \qquad (9e)$$

$$\sum_{p \in P_c} x_{pft}^{(3)} = x_{cft}^{(2)^*} \qquad \forall\, f, t \qquad (9f)$$

$$\sum_{p \in P_c} q_{pft}^{(3)} = q_{cft}^{(2)^*} \qquad \forall\, f, t \qquad (9g)$$

$$x_{pft}^{(3)}, q_{pft}^{(3)}, y_{fipt}^{(3)} \in Z^+ \qquad (9h)$$

Although this problem has the same objective and constraints as the original (1), it is significantly more computationally tractable as it only concerns the subset of items P_c in each sub-problem c. However, the level of complexity of this model is the same as warehouse disaggregation.

5.3. Solution approaches

In this section, we will proceed to describe the solution methods employed to deal with each of the disaggregation stages.

5.3.1. Exact method for demand zone disaggregation

Due to the total unimodularity property of the constraint matrix used in formulation (7), the first stage disaggregation can be solved optimally. In the following we aim to prove this claim.

Lemma 1. The first stage disaggregation problem (7) can be reduced to Linear programming model.

Proof.

Formulation (7) is an instance of integral network flow problem. Let (V, A) be the corresponding directed graph with capacity constraint $(0 \leq x \leq u)$ and flow conservation constraint $Bx = 0$. B is the node-arc incidence matrix and can be defined as $B = B_{v,a} \in \{-1,0,1\}^{|V|*|A|}$ where $B_{v,a} = 1$ if $a = (s, v)$, $B_{v,a} = -1$ if $a = (v, s)$, and $B_{v,a} = 0$ otherwise. According to the Ghouila-Houri theorem (Conforti et al., 2014b), the incidence matrix B of any directed graph is TU. Additionally, based on Hoffman and Kruskal theorem (Hoffman et al., 1956), let $B \in Z^{m \times n}$, for $(d, b, l, u) \in (R \cup \{\infty\})^m \times (R \cup \{-\infty\})^m \times (R \cup \{-\infty\})^n \times (R \cup \{\infty\})^n$, define the polyhedron Δ as:

$$\Delta(d, b, l, u) := \{x \in R^n : d \leq Bx \leq b, and\ l \leq x \leq u\}$$

B is TU if and only if $\Delta(d, b, l, u)$ is integral for every integer-valued choice of (d, b, l, u).

In other words, if the constraint matrix is TU, the polyhera is integral for every integer-valued of right-hand side. Hence, the linear relaxation of the integral network flow problem has an optimal integer-valued vertex solution. Therefore, we can omit the integrality constraints from the problem, and reduce it to the linear program. □

5.3.2. Approximation method for warehouse disaggregation

The complexity of optimization model (8) is equivalent to that of the model (1), which is classified as NP-hard. Accordingly, we can adjust the greedy algorithm we described in Section 4.1 to solve the model (8). Given our prior discussion on the algorithm, a detailed explanation is omitted here. Instead, a summary of the algorithm is provided below to keep the document concise. As mentioned earlier, the warehouse disaggregation model can be broken down into w subproblems and solved separately. The greedy algorithm can be employed for each cluster w, resulting in $Z^G_{Diss} = \sum_w Z^G_{Diss_w}$, where $Z^G_{Diss_w}$ represents the cost of greedy algorithm for disaggregating each group of w.

The greedy algorithm for solving the disaggregated model w requires a group of bundles $\varphi = (c, i)$ for each period where c and i stand for the items and demand points respectively. The value of $y^{(2)}_{f^*\varphi t}$ is determined by the minimum of $y^{(1)^*}_{w\varphi t}$ and the current capacity of warehouse f^*, i.e.,

$y^{(2)}_{f^*\varphi t} = \min\left\{y^{(1)^*}_{w\varphi t},\ \left(\varpi_{f^*t} - \bar{w}_{f^*t}\right)/\bar{v}_c\right\}$ where $\bar{w}_{ft}$ is the utilized capacity of f^*. Subsequently, we define new bundle $\varphi_1 = (f^*, c)$ to determine the variables $x^{(2)}_{fct}$ and $q^{(2)}_{fct}$. The value of $x^{(2)}_{\varphi_1 t}$ is determined based on the minimum of $\left(\sum_i y^{(2)}_{\varphi_1,i,t} + q^{(2)}_{\varphi_1 t}\right)$ and total current supply $(\eta_{ct} - \bar{\varepsilon}_{ct})$ for item c, i.e., $x^{(2)}_{\varphi_1 t} = min\left\{\left(\sum_i y^{(2)}_{\varphi_1,i,t} + q^{(2)}_{\varphi_1 t}\right), \eta_{ct} - \bar{\varepsilon}_{ct}\right\}$; here, η_{ct} stands for the total supply for item c, i.e., $\eta_{ct} = \bar{x}^*{}_{cwt}$ and $\bar{\varepsilon}_{ct}$ is the utilized portion of supply η_{ct} .The summary of algorithm is provided as follows:

Table 4. The summary of greedy algorithm for each cluster (w)

Step 1	Define N_w as the total number of (C, I) combinations in cluster w Set $t = T$, $q^{(2)}_{\varphi_1 T} = 0$
Step 2	Set $\bar{w}_{ft} = 0, \bar{\varepsilon}_{ct} = 0, \eta_{ct} = \bar{x}^*{}_{cwt}$
Step 3	Define $\varphi = (c, i)$, sort them in decreasing order of potential function $\rho = \frac{\sum_{f\in F_w}\left(r^{(2)}_{cft} + h^{(2)}_{cft} + \beta^{(2)}_{f\varphi t}\right) y^{(1)^*}_{w\varphi t}}{\bar{v}_c}$
Step 4	Set $\varphi = 1$ (represents the bundle (p, i) with the highest potential function)
Step 5	Choose a warehouse with the least cost where $f^* = Argmin\left\{\left(r^{(2)}_{cft} + h^{(2)}_{cft} + \beta^{(2)}_{f\varphi t}\right)\right\}$ If $\bar{w}_{f^*t} + v_\varphi \le \varpi_{f^*t}$: Assign bundle φ to the warehouse f^* $y^{(2)}_{f^*\varphi t} = \min\left\{y^{(1)^*}_{w\varphi t},\ \left(\varpi_{f^*t} - \bar{w}_{f^*t}\right)/\bar{v}_c\right\}$ $\bar{w}_{f^*t} = \bar{w}_{f^*t} + \bar{v}_c * y^{(2)}_{f^*\varphi t}$ $y^{(1)^*}_{w\varphi t} = y^{(1)^*}_{w\varphi t} - y^{(2)}_{f^*\varphi t}$ Else move to step 6
Step 6	While $y^{(1)^*}_{w\varphi t} > 0$, then $f^* = f^* + 1$, move to step 5
Step 7	Set $\varphi = \varphi + 1$, move to step 5
Step 8	Define $\varphi_1 = (f^*, c)$, sort them in decreasing order of $(\sum_i y^{(2)}_{\varphi_1,i,t} + q^{(2)}_{\varphi_1 t})$

Step 9	If $\varepsilon_{ct} \leq \eta_{ct}$: $x^{(2)}_{\varphi_1 t} = min\left\{\left(\sum_i y^{(2)}_{\varphi_1,i,t} + q^{(2)}_{\varphi_1 t}\right), \eta_{ct} - \varepsilon_{ct}\right\}$ $q^{(2)}_{\varphi_1,t-1} = \sum_i y^{(2)}_{\varphi_1,i,t} - x^{(2)}_{\varphi_1 t} + q^{(2)}_{\varphi_1 t}$ $\varepsilon_{ct} = \varepsilon_{ct} + x^{(2)}_{\varphi_1 t}$ Else $x^{(2)}_{\varphi_1 t} = 0, q^{(2)}_{\varphi_1,t-1} = \sum_i y^{(2)}_{\varphi_1,i,t} + q^{(2)}_{\varphi_1 t}$
Step 10	Set $\varphi_1 = \varphi_1 + 1$, move to step 9
Step 11	$\bar{w}_{f,t-1} = \bar{w}_{f,t-1} + \bar{v}_c * q^{(2)}_{\varphi_1,t-1}$
Step 12	Set $t = t - 1$, move to step 3

5.3.3. Approximation method for item disaggregation

The optimization model (9) is characterized as NP-hard, the same as model (1). Consequently, we can adjust and apply the greedy algorithm described earlier to solve the model (9) in polynomial time. The items disaggregation model can be decomposed into c subproblems and solved individually. The greedy algorithm can be utilized for each cluster c, resulting in $Z^p_{Diss} = \sum_{c\in C} Z^p_{Diss_c}$, where $Z^p_{Diss_c}$ represents the cost of greedy algorithm for disaggregating each cluster c.

We employ a similar approach used for solving the warehouse disaggregation, with the exception that certain parameters are modified as outlined in the table below. Since we are solving the algorithm for each cluster c, we first define bundle $\varphi = (p, i)$ for $p \in p_c$. Subsequently, we introduce parameter γ_{ft} representing the warehouse capacity for each cluster c, defined as $\gamma_{\text{ft}} = \bar{v}_c(x^{(2)^G}_{cft} + q^{(2)^G}_{cf,t-1})$. In addition, the supply value for each cluster c equals to $\eta'_{ft} = x^{(2)^G}_{cft}$. We also incorporate new conditions in step 5, ensuring that the total values assigned from warehouse f to demand point i for a specific cluster c do not exceed the predetermined flow obtained from the previous stage, i.e., $y^{(2)^G}_{f^*ict}$. The summary of algorithm is provided as follows:

Table 5. The summary of the greedy algorithm for each cluster (c)

Step 1	Define N_c as the total number of (P, I) combinations in cluster c

```
        Set $t = T$, $q^{(3)}_{\varphi_1,\mathrm{T}} = 0$
Step 2  Set $\bar{w}_{ft} = 0, \bar{\varepsilon}_{ft} = 0, \bar{\chi}_{fit} = 0, \gamma_{\mathrm{ft}} = \bar{v}_c(x^{(2)^G}_{cft} + q^{(2)^G}_{cf,t-1}), \eta'_{ft} = x^{(2)^G}_{cft}$
Step 3  Define $\varphi = (\mathrm{p}, i)$, sort them in decreasing order of potential function $\rho = \frac{\sum_{f\in F}(r_{pft}+h_{p,f,t}+\beta_{f\varphi t})d_{\varphi t}}{v_p}$
Step 4  Set $\varphi = 1$ (represents the bundle $(\mathrm{p}, i)$ with the highest potential function)
Step 5  Choose a warehouse with the least cost where $f^* = Argmin\{(r_{pft} + h_{p,f,t} + \beta_{fipt})\}$

        If $y^{(2)^G}_{f^*ict} - \bar{\chi}_{f^*it} > 0$:
            If $\bar{w}_{f^*t} + v_p \le \gamma_{f^*t}$:
                Assign bundle $\varphi$ to the warehouse $f^*$
                $y^{(3)}_{f^*\varphi t} = \min\{d_{\varphi t}, (\gamma_{f^*t} - \bar{w}_{f^*t})/v_p\}$
                $\bar{w}_{f^*t} = \bar{w}_{f^*t} + v_p * y^{(3)}_{f^*\varphi t}$
                $d_{\varphi t} = d_{\varphi t} - y^{(3)}_{f^*\varphi t}$
                $\bar{\chi}_{f^*it} = \bar{\chi}_{f^*it} + y^{(3)}_{f^*\varphi t}$
            Else move to step 6
        Else move to step 6
Step 6  While $d_{\varphi t} > 0$, then $f^* = f^* + 1$, move to step 5
Step 7  Set $\varphi = \varphi + 1$, move to step 5
Step 8  Define $\varphi_1 = (f^*, \mathrm{p})$, sort them in decreasing order of $(\sum_i y^{(3)}_{f^*\varphi t} + q^{(3)}_{\varphi_1 t})$
Step 9  If $\bar{\varepsilon}_{ft} < \eta'_{ft}$:
            $x^{(3)}_{\varphi_1 t} = min\{(\sum_i y^{(3)}_{\varphi_1,i,t} + q^{(3)}_{\varphi_1 t}), \eta'_{ft} - \bar{\varepsilon}_{ft}\}$
            $q^{(3)}_{\varphi_1,t-1} = \sum_i y^{(3)}_{\varphi_1,i,t} - x^{(3)}_{\varphi_1 t} + q^{(3)}_{\varphi_1 t}$
            $\bar{\varepsilon}_{ft} = \bar{\varepsilon}_{ft} + x^{(3)}_{\varphi_1 t}$
        Else $x^{(3)}_{\varphi_1 t} = 0$, $q^{(3)}_{\varphi_1,t-1} = \sum_i y^{(3)}_{\varphi_1,i,t} + q^{(3)}_{\varphi_1 t}$
```

Step 10	Set $\varphi_1 = \varphi_1 + 1$, move to step 9
Step 11	$\overline{w}_{f,t-1} = \overline{w}_{f,t-1} + v_p * q^{(3)}_{\varphi_1,t-1}$
Step 12	Set $t = t - 1$, move to step 3

5.4. The algorithm error bound

In the aggregation approach, a crucial aspect involves quantifying the extent of accuracy loss resulting from aggregation. While accurately calculating this difference necessitates solving both the original and aggregated problems, it's possible to establish bounds for the loss of accuracy without solving the original problem. Typically, two types of error bounds are considered: a priori and a posteriori. A priori error bounds are determined after aggregation but before optimizing the aggregated problem, whereas calculating a posteriori error bound necessitates knowledge of an optimal solution for the aggregated problem. A priori error bounds tend to be more conservative than a posteriori ones and are often employed to guide decisions regarding clustering and cluster procedures, despite no evidence suggesting that methods yielding the tightest a priori error bound necessarily yield the tightest a posteriori error bound (Litvinchev & Tsurkov, 2003; Rogers et al., 1991).

Using duality theory, several authors measure the error bounds by comparing the actual optimal value of the objective function with the optimal value for the aggregated problem, while ignoring the disaggregation part. For example, Zipkin (1980c) develops a bound on the optimality gap for linear programming models resulting from column aggregations, as well as both column and row aggregations (Zipkin, 1980a). Subsequently, the improvement on the Zipkin’s bound are presented by Mendelssohn (1980) and Shetty & Taylor (1987). A. Hallefjord & Storoey (1986) also modify one version of Zipkin’s bound for the generalized assignment problem. To the best of our knowledge, Chen & Graves (2021) is the only work employing a series of intermediate formulations based on their problem structure to bound the cost between the original and the disaggregated final solutions. In this section, our goal is to establish a posteriori error bound for the algorithm which is derived from the disaggregated solution. Before delving into that, it's essential to demonstrate that the solution derived from the disaggregation remains feasible for the original model.

5.4.1. Proof of feasibility

To derive a feasible solution to the optimal problem (1) from the aggregated solution, we conducted three stages of hierarchical partial disaggregation, as previously described. Here, our objective is to demonstrate that the solution obtained from the disaggregation stages satisfies the constraints of the original problem.

At each stage of disaggregation, we solve a series of subproblems for each k, w and c subject to their respective constraints. In addition, the hierarchical approach to disaggregation maintains solution consistency, ensuring that the final disaggregated solution from the last step does not violate any of the original constraints. For instance, $y^{(3)}_{f_w i_k p_c t}$ obtained from the last step of the disaggregation ensures to fulfill the demand d_{pit}. This solution depends on the outcomes of previous steps, thereby ensuring overall consistency.

$$\sum_{w} y^{(1)}_{wci_kt} = \sum_{w} \sum_{f\in F_W} y^{(2)}_{f_wi_kct} = \sum_{w} \sum_{p\in P_c} \sum_{f\in F_W} y^{(3)}_{f_wi_kp_ct} \geq d_{pit} \tag{6}$$

Similarly, this holds true for $x^{(3)}_{pft}$ and $q^{(3)}_{pf,t}$:

$$\sum_{w} \bar{X}_{cwt} = \sum_{w} \sum_{f\in F_w} x^{(2)}_{cft} = \sum_{p\in P_c} \left(\sum_{w} \sum_{f\in F_w} x^{(3)}_{pft} \leq \alpha_{pt} \right) \leq \bar{\alpha}_{ct} \tag{7}$$

$$\sum_{p\in P_c} v_p q^{(3)}_{pf,t-1} + \sum_{p\in P_c} v_p x^{(3)}_{pft} \leq \bar{v}_c x^{(2)}_{cft} \leq \varpi_{ft} \tag{8}$$

5.4.2. Proof of optimality gap

Any feasible solution to the original minimization problem provides an upper bound on its optimal objective value. Therefore, to determine the gap between the disaggregated and optimal solutions of the original model, we must compute:

$$\boldsymbol{Z^{p*}_{Diss} - Z^{*}{}_{opt} = \sum_{c\in\overline{C}} Z^{p*}_{Diss_c} - Z^{*}{}_{Opt} \leq |C| \max_c Z^{p*}_{Diss_c} - Z^{*}{}_{R}} \tag{9}$$

Where $Z^{*}{}_{R}$ represents the optimal solution of the relaxed original problem (14) formulated as follows:

$$Z_R = min\left(\sum_{f} r_{pft} * x_{pft} + \sum_{p,f,t} h_{pft} * q_{pft} + \sum_{f,i,p,t} \beta_{fipt} * y_{fipt}\right) \qquad (10a)$$

$subject\ to$:

Constraints (1b) to (1e)

$$x_{pft},\ q_{pft}, y_{fipt} \in R^+ \qquad (14b)$$

To find the optimal solution of the relaxed problem, $Z^*{}_R$, we can use the concept of LP duality. The dual form of the problem (14) is formulated as follows:

$$Z_R^D = \underset{u,\zeta,\Theta}{Max}\left(\sum_{p,i,t} \mathrm{u}_{pit} d_{pit} - \sum_{p,t} \zeta_{pt}\ \alpha_{pt} - \sum_{f,t} \Theta_{ft}\varpi_{ft}\right) \qquad (11a)$$

Subject to:

$$\mathrm{u}_{pit} + \Gamma_{pft} \leq \beta_{fipt} \qquad \forall\, p, i, t \qquad (15b)$$

$$-\Gamma_{pft} - \zeta_{pt} - v_p\Theta_{ft} \leq r_{pft} \qquad \forall\, p, f, t \qquad (15c)$$

$$\Gamma_{pf,t} \leq h_{pf,t} \qquad \forall\, p, f, t \qquad (15d)$$

$$-\Gamma_{pf,t+1} - v_p\Theta_{f,t+1} \leq h_{pf,t} \qquad \forall\, p, f, t = 2, \dots T \qquad (15e)$$

$$\mathrm{u}_{pit}, \zeta_{pt}, \Theta_{ft} \geq 0 \qquad \forall\, p, f, i, t \qquad (15f)$$

Where $(\mathrm{u}_{pit}, \zeta_{pt}, \Theta_{ft}, \Gamma_{pft})$ are introduced as decision variables of dual model (14) which are assigned as follows:

- u_{pit} is a decision variable for constraint (1b)
- Γ_{pft} is considered for constraint (1c)
- ζ_{pt} is considered for constraint (1d)
- Θ_{ft} is considered for constraint (1e)

Using strong duality, the objective function value of the optimal relaxed original model (14) is equal to its corresponding dual objective value (15), i.e., $Z_R^{*D} = Z^*{}_R$, therefore we have:

$$Z^*{}_R = \sum_{p,i,t} u^*_{pit} d_{pit} - \sum_{p,t} \zeta^*_{pt}\, \alpha_{pt} - \sum_{f,t} \Theta^*_{ft} \varpi_{ft} \tag{12}$$

Furthermore, to find the upper-bound for the first term of the optimality gap, i.e., $|C| \max_c Z^{p*}_{Diss_c}$, we can define the following optimization model:

$$\begin{aligned}
&|C|\, max_c\, Z^{p*}_{Diss_c} \\
&= |C|\, max_c\, min \left(\sum_{f,p \in P_c,t} r_{pft} * x^{(3)^*}_{pft} + \sum_{p \in P_c,f,t} h_{pft} * q^{(3)^*}_{pft} + \sum_{p \in P_c,i,f,t} \beta_{fipt} * y^{(3)^*}_{fipt} \right) \\
&= |C|\, max\, \delta \qquad\qquad (13a)
\end{aligned}$$

Subject to:

$$\delta \leq \sum_{f,p \in P_c,t} r_{pft} * x^{(3)^*}_{pft} + \sum_{p \in P_c,f,t} h_{pft} * q^{(3)^*}_{pft} + \sum_{p \in P_c,i,f,t} \beta_{fipt} * y^{(3)^*}_{fipt} \qquad \forall c \in C \tag{17b}$$

Constraints (9b) to (9h).

If we rewrite $q^{(3)^*}_{pft} = \sum_{j=1}^{t} \left(x^{(3)^*}_{pfj} - \sum_i y^{(3)^*}_{fipj} \right) \quad \forall\, p \in P_c, f, t$, we obtain:

$$\delta \leq \sum_{f,p \in P_c,t} r_{pft} * x^{(3)^*}_{pft} + \sum_{p \in P_c,f,t} h_{pft} * \sum_{j=1}^{t} \left(x^{(3)^*}_{pfj} - \sum_i y^{(3)^*}_{fipj} \right) + \sum_{p \in P_c,i,f,t} \beta_{fipt} * y^{(3)^*}_{fipt} \qquad \forall c \in C$$

Then, we have:

$$\begin{aligned}
\delta \leq &\sum_{f,p \in P_c,t} \left(r_{pft} + h_{pft} \right) * x^{(3)^*}_{pft} + \sum_{p \in P_c,i,f,t} \left(\beta_{fipt} - h_{pft} \right) * y^{(3)^*}_{fipt} + \sum_{p \in P_c,f,t} h_{pft} \left(\sum_{j=1}^{t-1} x^{(3)^*}_{pfj} \right) \\
&- \sum_{p \in P_c,f,i,t} h_{pft} \left(\sum_{j=1}^{t-1} y^{(3)^*}_{fipj} \right) \qquad \forall c \in C \qquad (17c)
\end{aligned}$$

Using Lagrangian relaxation on constraint (17b), the following relations hold for all $\lambda_c \geq 0$:

$$
\begin{aligned}
|C|\delta^* \le |C|\delta^* + \sum_{c} \lambda_c \Bigg(& \sum_{f,p\in P_c,t} \left(r_{pft} + h_{pft}\right) * x_{pft}^{(3)^*} + \sum_{p\in P_c,i,f,t} \left(\beta_{fipt} - h_{pft}\right) * y_{fipt}^{(3)^*} \\
& + \sum_{p\in P_c,f,t} h_{pft} \left(\sum\nolimits_{j=1}^{t-1} x_{pfj}^{(3)^*}\right) - \sum_{p\in P_c,f,i,t} h_{pft} \left(\sum\nolimits_{j=1}^{t-1} y_{fipj}^{(3)^*}\right) - \delta^* \Bigg) \\
& \le \left(|C| - \sum_{c} \lambda_c\right) \delta^* \\
& + \sum_{c} \lambda_c \Bigg[\sum_{f,p\in P_c,t} \left(r_{pft} + h_{pft}\right) * x_{pft}^{(3)^*} + \sum_{p\in P_c,i,f,t} \left(\beta_{fipt} - h_{pft}\right) * y_{fipt}^{(3)^*} \\
& + \sum_{p\in P_c,f,t} h_{pft} \left(\sum\nolimits_{j=1}^{t-1} x_{pfj}^{(3)^*}\right) \Bigg] \qquad (17d)
\end{aligned}
$$

Since the inventory level is non-negative, $q_{pft}^{(3)^*} \ge 0$, we have $q_{pf,t-1}^{(3)^*} + x_{pft}^{(3)^*} - \sum_i y_{fipt}^{(3)^*} \ge 0 \quad \forall\, p \in P_c, f, t,$ then we have $\sum_i y_{fipt}^{(3)^*} \le q_{pf,t-1}^{(3)^*} + x_{pft}^{(3)^*}$, which in turn $\sum_i y_{fipt}^{(3)^*} \le \sum_{j=1}^{t} x_{pfj}^{(3)^*} \quad \forall\, p \in P_c, f, t.$

We can rewrite the above expression as follows:

$$\left(|C|-\sum_{c}\lambda_c\right)\delta^*$$

$$+\sum_{c}\lambda_c\left[\sum_{f,p\in P_c,t}\left(r_{pft}+h_{pft}\right)*x_{pft}^{(3)^*}+\sum_{p\in P_c,i,f,t}\left(\beta_{fipt}-h_{pft}\right)*y_{fipt}^{(3)^*}\right.$$

$$\left.+\sum_{p\in P_c,f,t}h_{pft}\left(\sum\nolimits_{j=1}^{t-1}x_{pfj}^{(3)^*}\right)\right]$$

$$\leq\left(|C|-\sum_{c}\lambda_c\right)\delta^*$$

$$+\sum_{c}\lambda_c\left[\sum_{f,p\in P_c,t}\left(r_{pft}+h_{pft}\right)*x_{pft}^{(3)^*}+\sum_{p\in P_c,f,t}max_i\left(\beta_{fipt}-h_{pft}\right)*\sum_{i}y_{fipt}^{(3)^*}\right.$$

$$\left.+\sum_{p\in P_c,f,t}h_{pft}\left(\sum\nolimits_{j=1}^{t-1}x_{pfj}^{(3)^*}\right)\right]$$

$$\leq\left(|C|-\sum_{c}\lambda_c\right)\delta^*$$

$$+\sum_{c}\lambda_c\left[\sum_{f,p\in P_c,t}\left(r_{pft}+h_{pft}\right)*x_{pft}^{(3)^*}+\sum_{p\in P_c,f,t}max_i\left(\beta_{fipt}-h_{pft}\right)*\sum\nolimits_{j=1}^{t}x_{pfj}^{(3)^*}\right.$$

$$\left.+\sum_{p\in P_c,f,t}h_{pft}\left(\sum\nolimits_{j=1}^{t-1}x_{pfj}^{(3)^*}\right)\right]$$

$$= \left(|C| - \sum_{c} \lambda_c\right)\delta^*$$

$$+ \sum_{c} \lambda_c \left[\sum_{f,p\in P_c,t} (r_{pft} + h_{pft}) * x_{pft}^{(3)^*} + \sum_{p\in P_c,f,t} max_i(\beta_{fipt}) * \sum\nolimits_{j=1}^{t} x_{pfj}^{(3)^*} \right.$$

$$\left. - \sum_{p\in P_c,f,t} h_{pft}\left(x_{pft}^{(3)^*}\right) \right]$$

$$\leq \left(|C| - \sum_{c} \lambda_c\right)\delta^* + \sum_{c} \lambda_c \left[\sum_{f,p\in P_c,t} r_{pft} * x_{pft}^{(3)^*} + \sum_{p\in P_c,f,t} max_i(\beta_{fipt}) * \sum\nolimits_{j=1}^{t} x_{pfj}^{(3)^*} \right]$$

$$\leq \left(|C| - \sum_{c} \lambda_c\right)\delta^*$$

$$+ \sum_{c} \lambda_c \left[\sum_{f,p\in P_c,t} r_{pft} * x_{pft}^{(3)^*} + \sum_{p\in P_c,f,t} |t| max_i(\beta_{fipt}) * max_{j=1,..t}\left(x_{pfj}^{(3)^*}\right) \right]$$

According to the inequality (9e), we have $\sum_{p\in P_c} x_{pft}^{(3)^*} \leq \frac{\bar{v}_c\left(x_{cft}^{(2)^*} + q_{cf,t-1}^{(2)^*}\right)}{min_{p\in P_c}\{v_p\}} \quad \forall f, t$, therefore we can obtain the following results:

$$\left(|C| - \sum_{c} \lambda_c\right)\delta^* + \sum_{c} \lambda_c \left[\sum_{f,p\in P_c,t} r_{pft} * x_{pft}^{(3)^*} + \sum_{p\in P_c,f,t} |t| max_i(\beta_{fipt}) * max_{j=1,..t}\left(x_{pfj}^{(3)^*}\right) \right]$$

$$\leq \left(|C| - \sum_{c} \lambda_c\right)\delta^*$$

$$+ \sum_{c} \lambda_c \left[\sum_{f,t} max_{p\in P_c}\, r_{pft} \sum_{p\in P_c} x_{pft}^{(3)^*} \right.$$

$$\left. + \sum_{p\in P_c,f,t} |t| max_{i,p\in P_c}(\beta_{fipt}) * \sum_{p\in P_c} max_{j=1,..t}\left(x_{pfj}^{(3)^*}\right) \right]$$

$$\leq \left(|C| - \sum_c \lambda_c\right)\delta^*$$
$$+ \sum_c \lambda_c \left[\sum_{f,t} max_{p\in P_c}\, r_{pft} \frac{\bar{v}_c \left(x_{cft}^{(2)^*} + q_{cf,t-1}^{(2)^*}\right)}{min_{p\in P_c}\{v_p\}}\right.$$
$$\left. + \sum_{f,t} |t| max_{i,p\in P_c}\left(\beta_{fipt}\right) * \left(\frac{\bar{v}_c \left(x_{cfj}^{(2)^*} + q_{cf,j-1}^{(2)^*}\right)}{min_{p\in P_c}\{v_p\}}\right)\right]$$
$$= \left(|C| - \sum_c \lambda_c\right)\delta^*$$
$$+ \sum_c \lambda_c \left[\sum_{f,t}\left(max_{p\in P_c}\, r_{pft} + |t| max_{i,p\in P_c}\left(\beta_{fipt}\right)\right)\frac{\bar{v}_c \left(x_{cft}^{(2)^*} + q_{cf,t-1}^{(2)^*}\right)}{min_{p\in P_c}\{v_p\}}\right]$$

Since $x_{cft}^{(2)^*} + q_{cf,t-1}^{(2)^*} \leq \sum_c x_{cft}^{(2)^*} + q_{cf,t-1}^{(2)^*}$, we have:

$$\left(|C| - \sum_c \lambda_c\right)\delta^* + \sum_c \lambda_c \left[\sum_{f,t}\left(max_{p\in P_c}\, r_{pft} + |t| max_{i,p\in P_c}(\beta_{fipt})\right)\frac{\bar{v}_c \left(x_{cft}^{(2)^*} + q_{cf,t-1}^{(2)^*}\right)}{min_{p\in P_c}\{v_p\}}\right]$$
$$\leq \left(|C| - \sum_c \lambda_c\right)\delta^*$$
$$+ \sum_c \lambda_c \left[\sum_{f,t}\left(max_{p\in P_c}\, r_{pft} + |t| max_{i,p\in P_c}(\beta_{fipt})\right)\frac{\bar{v}_c \left(\sum_c x_{cft}^{(2)^*} + q_{cf,t-1}^{(2)^*}\right)}{min_{p\in P_c}\{v_p\}}\right]$$

According to the inequality (8e):

$$\sum_c x_{cft}^{(2)^*} + q_{cf,t-1}^{(2)^*} \leq \frac{\varpi_{ft}}{min_c\{\bar{v}_c\}}$$

Then,

$$\left(|C| - \sum_{c} \lambda_c\right)\delta^* + \sum_{c} \lambda_c \left[\sum_{f,t} \left(max_{p\in P_c}\, r_{pft} + |t| max_{i,p\in P_c}\left(\beta_{fipt}\right)\right) \frac{\overline{v}_c \left(\sum_c x_{cft}^{(2)^*} + q_{cf,t-1}^{(2)^*}\right)}{min_{p\in P_c}\{v_p\}}\right]$$

$$\leq \left(|C| - \sum_{c} \lambda_c\right)\delta^*$$

$$+ \sum_{c} \lambda_c \left[\sum_{f,t} \left(max_{p\in P_c}\, r_{pft} + |t| max_{i,p\in P_c}\left(\beta_{fipt}\right)\right) \frac{\overline{v}_c(\varpi_{ft})}{min_{p\in P_c}\{v_p\}\, min_c\{\overline{v}_c\}}\right]$$

As a result, for all $\lambda_c \geq 0$:

$$|C| \max_c Z_{Diss\,c}^{p*} \leq \left(|C| - \sum_{c} \lambda_c\right)\delta^* + \sum_{c} \lambda_c \left[\sum_{f,t} \left(max_{p\in P_c}\, r_{pft} + |t| max_{i,p\in P_c}\left(\beta_{fipt}\right)\right) \frac{\overline{v}_c(\varpi_{ft})}{min_{p\in P_c}\{v_p\} \min_c\{\overline{v}_c\}}\right] \qquad (17g)$$

Consequently,

$$Z_{Diss}^{p*} - Z^*{}_{ORG} \leq \left(|C| - \sum_{c} \lambda_c\right)\delta^* + \sum_{c} \lambda_c \left[\sum_{f,t} max_{p\in P_c} \left(r_{pft} + |t| max_i\left(\beta_{fipt}\right)\right) \frac{\overline{v}_c(\varpi_{ft})}{min_{p\in P_c}\{v_p\} \min_c\{\overline{v}_c\}}\right] - \left(\sum_{p,i,t} u_{pit}^* d_{pit} - \sum_{p,t} \zeta_{pt}^*\, \alpha_{pt} - \sum_{f,t} \Theta_{ft}^* \varpi_{ft}\right) \qquad (17h)$$

6. Computational experiments

In this section, we aim to implement our proposed algorithm on a specific instance of an online retail inventory system and assess the algorithm performance. First, we create a synthetic network inspired from Amazon fulfillment network in the United States. The network characteristics, which include fulfillment centers and demand zone features, cost parameters, and item attributes are detailed in Section 6.1. Then we provide the following computational experiments:

- In Section 6.2, we investigate the impact of exclusive cluster counts on the optimality gap.
- In Section 6.3, we study how cluster combinations affect a posteriori optimality gap and computation time.
- In Section 6.4, we analyze how variations in item sizes affect both the algorithm's running time and the optimality gap.
- In Section 6.5, we study how changes in the ratio of the objective function parameters affect the algorithm performance.

All experiments are implemented in Python 3.10.14 and its interface to Gurobi 9.5.2. We use UBC computational cluster (Sockeye) to run our experiments under the following configuration:

Operating system: Enterprise Linux 7 (EL7)

CPU: 2 x Intel Xeon Gold 6130 (2.1GHz)

Total Cores: 32

Mem per Core: 24GB

6.1. Simulation description

Our goal is to create a realistic simulation of the Amazon fulfillment network. Given the computational intensity of considering the actual number of demand points, fulfillment centers, and items, we select a representative subset. Our example involves $|P| = 1000$ items over $|T| = 4$ planning periods. Most parameters follow a structure similar to that described by Chen & Graves, (2021).

Based on the dataset from Amazon (2022), there are a total of $|F| = 545$ fulfilment centers in the US, from which we select 200 major centers for our case. The total capacities of fulfillment centers are determined by multiplying the total demand of all items by their volume. In other words, we assume that $\sum_{f=1}^{F} \varpi_f = \sum_{t=1}^{T} \sum_{p=1}^{P} \sum_{i=1}^{I} d_{pit} * v_p$. This total capacity is then divided among the fulfillment centers in proportion to their square footage, which is uniformly generated between the minimum and maximum possible sizes. According to Amazon (2024) the size of fulfilment centers varies from 600,000 to 1 million square feet.

The number of counties in the United States is utilized to determine demand points. The US Census data from 2022 indicates there are 3,145 counties. We select a subset of counties with populations exceeding 160,000, resulting in $|I| = 410$ demand points. Demand for each item at each demand point is determined proportionally to the population. First, we generate the total demand across all demand points, then distribute it based on each demand point's population. We assume total demand of each item ranges within $d_{pt} \in [10000, 20000]$, and demand at each demand point is given by $d_{pit} \in d_{pt} * \varepsilon_i$, where ε_i is the population ratio of demand point i.

Shipping costs are calculated using a piecewise-linear function of the distance between fulfillment centers and demand points, as well as item weight, following the cost structure described in Chen & Graves, (2021).

$$\beta_{fip} = \begin{cases} 0.247847\varrho_{fi}\tau_p - 0.227878\varrho_{fi} + 0.132063\tau_p + 7.553833, & \varrho_{fi} \leq 1 \\ 1.37674\varrho_{fi}\tau_p + 4.02487\varrho_{fi} + 0.75604\tau_p + 11.10890, & otherwise \end{cases}$$

Where τ_p represents the weight of item p and ϱ_{fi} is the spherical distance (in thousands of miles) between fulfillment center f and region i . The haversine formula is used to calculate ϱ_{fi} which uses the longitude and latitude coordinates of the demand points and warehouses.

Other parameter values are defined as follows:

$v_p \in [1,35]$ in cubic meters

$\tau_p \in [0,36]$ in pounds

$r_{pft} \in [1,6]$ in USD

$h_{pft} \in [1,5]$ in USD

In this experiment, we use k-means clustering technique for aggregation. The normalized vectors of $(\tau_p, v_p, d_{pt}, \tilde{h}_p, \tilde{r}_p)$ are used as the clustering features for items aggregation, with $\tilde{h}_p$ and $\tilde{r}_p$ standing for the average holding and receiving costs across all warehouses and periods. Furthermore, we use the latitude and longitude coordinates for warehouses and demand zone aggregation. The aggregated parameters are determined as follows:

$$\bar{r}_{cwt} = \sum_{f \in F_w} \sum_{p \in P_c} \frac{r_{pft}}{|P_c||F_w|} \qquad \forall\, c, w, t$$

$$\bar{h}_{cw} = \sum_{f \in F_w} \sum_{p \in P_c} \frac{h_{pf}}{|P_c||F_w|} \qquad \forall\, c, w, t$$

$$\bar{\beta}_{wkc} = \sum_{i \in I_k} \sum_{f \in F_w} \sum_{p \in P_c} \frac{\beta_{fip}}{|P_c||F_w||I_k|} \qquad \forall\, w, k, c$$

$$\bar{v}_c = max_{p \in P_c}\{v_p\} \qquad \forall\, c$$

Considering the instance of our model which includes $|P| = 1000, |T| = 4, |I| = 410, |F| = 200$, the optimal model (1) has an objective function value of $Z^*_{OPT} = 532{,}936{,}115.33$ and requires 3.68 hours to compute. The solution of the greedy algorithm for model (1) is $Z^G_{OPT} = 549645540.5$ with a running time of 21.5 minutes.

6.2. Effect of exclusive cluster size on optimality gap

This experiment investigates how the size of each cluster ($|C|$, $|K|$, and $|W|$) exclusively affects the optimality gap. It aims to understand the relationship between different cluster sizes and their exclusive effects on the algorithm performance. Four different values are assigned to each cluster: $|C| = (10, 50, 100, 200)$, $|K| = (4, 20, 40, 100)$, and $|W| = (2, 10, 20, 50)$, resulting in a total of $4^3 = 64$ cluster combinations.

We first investigate how variations in item cluster C affect the optimality gap for a fixed combination of K and W, and similarly replicate the same experiment for K and W. Hence, this comprehensive analysis not only enables us to comprehend the relationship among different clusters but also effectively identifies the appropriate cluster combination. Thus, the experiment is conducted for 16 fixed combinations of two clusters, e.g., K and W, to evaluate the impact of variations in one cluster, e.g., C, on algorithm performance.

The results of this experiment are depicted in Figure 3, and Figure 4, with the combinations arranged in the order of $|C|$, $|K|$, and $|W|$, respectively. The cluster being analyzed is denoted by (*). For instance, when we refer to (*)-4-2, it indicates that the analysis focuses on cluster C, with the sizes of $|K|$ and $|W|$ being held constant at 4 and 2. The same logic applies to the remaining experiments.

Based on Figure 3, there is a linear relationship between the size of $|C|$ and optimality gap. Regardless of the sizes of $|K|$ and $|W|$, we consistently observe improved performance as the size of C increases. This improvement is due to the fact that the accuracy of algorithm is contingent upon the aggregated model (6). As the size of $|C|$ increases, the model (6) becomes a more precise approximation of the optimal model (1), leading to an enhancement of the overall algorithm performance.

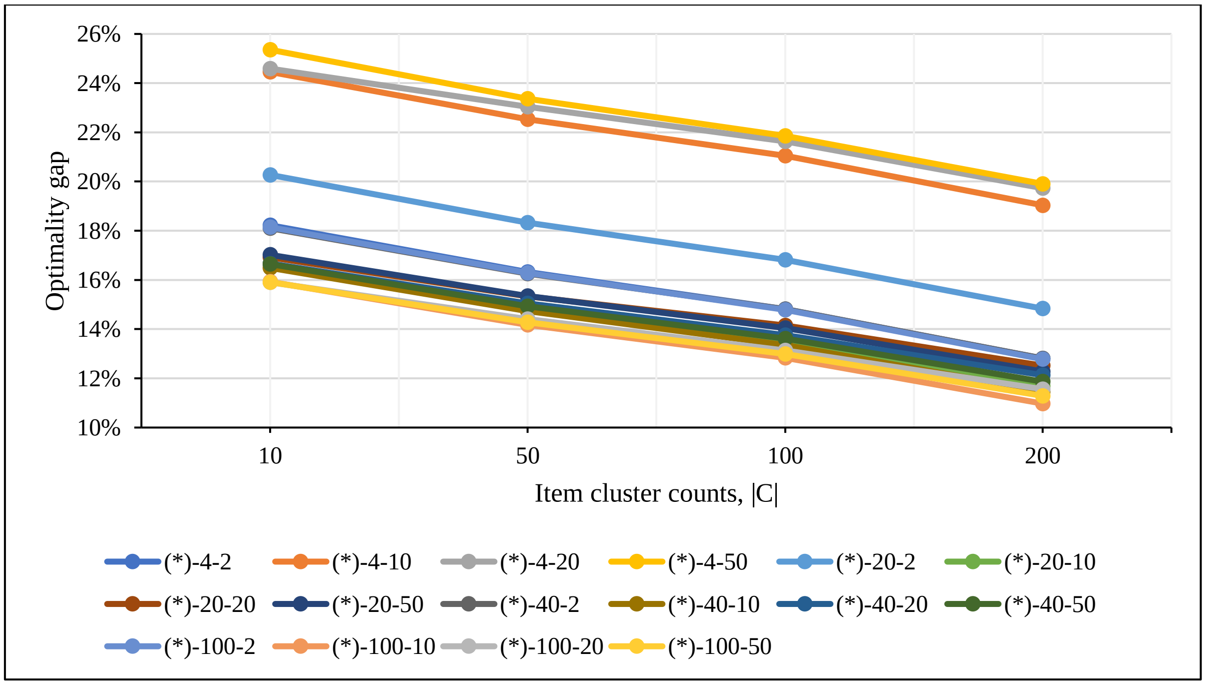


Figure 3. Optimality gap versus cluster size |C|

However, when we repeat the experiment for $|K|$ and $|W|$, we do not observe a similar linear relationship. Generally, the algorithm's performance is enhanced by increasing the size of W; yet this is only true when $|K|$ is equal to or greater than $|W|$. For instance, the combination $|K| = 100$, $|W| = 50$ yields a superior bound (i.e., 11%) than $|K| = 100, |W| = 2$ $(i.e., 16\%)$ for a fixed size of $|C|$. Therefore, as long as $|W| \leq |K|$, an improved bound is achieved by increasing $|W|$ for fixed values of $|K|$ and $|C|$.

As illustrated in Figure 4, the majority of cases do not yield favorable results when the size of $|W|$ exceeds the size of $|K|$. For instance, the maximum gap of 25% occurs at $|K| = 4$, and $|W| = 50$ for $|C| = 10$, whereas the gap is reduced to 18% at $|K| = 100, |W| = 2$. This pattern also confirms that the optimality gap will be reduced if the size of $|K|$ is greater than the size of $|W|$.

Increasing the size of $|K|$ improves the optimality gap in cases when $|K| < |W|$. For instance, the gap is reduced by nearly 9% when $|K|$ is increased from 4 to 20 for $|C| = 10$ and $|W| = 50$. It is worth noting that the optimality gap is not always positively influenced by an increase in $|K|$, as illustrated in Figure 4. In other words, the gap either remains constant or becomes worse as $|K|$ increases when $|K|$ exceeds $|W|$. This is evident in Figure 4, where the optimality gap increases by 2% when the value of $|K|$ is increased from 4 to 20 for the blue and gray lines with $|W| = 2$. In addition, the optimality gap remains constant at approximately 12.8% when the value of $|K|$ is increased from 4 to 40, for a fixed values of $|W|$ and $|C|$, when $|C| = 200$ and $|W| = 2$. It does not improve the approximation; rather, it merely increases computation time.

One possible explanation for the improved performance observed when the size of $|K|$ is greater than $|W|$ is the geographical distribution of demand points compared to warehouses, as it is shown in Figure 6. Having more clusters $|K|$ allows for greater similarity of demand points within the same cluster. Consequently, the number of clusters $|K|$ must be at least as large as the number of warehouse clusters $|W|$. Furthermore, the first stage of disaggregation begins with demand zones, where the value of $y_{wict}^{(1)^*}$ is determined optimally. The accuracy of this stage significantly impacts subsequent stages, leading to enhanced overall performance.

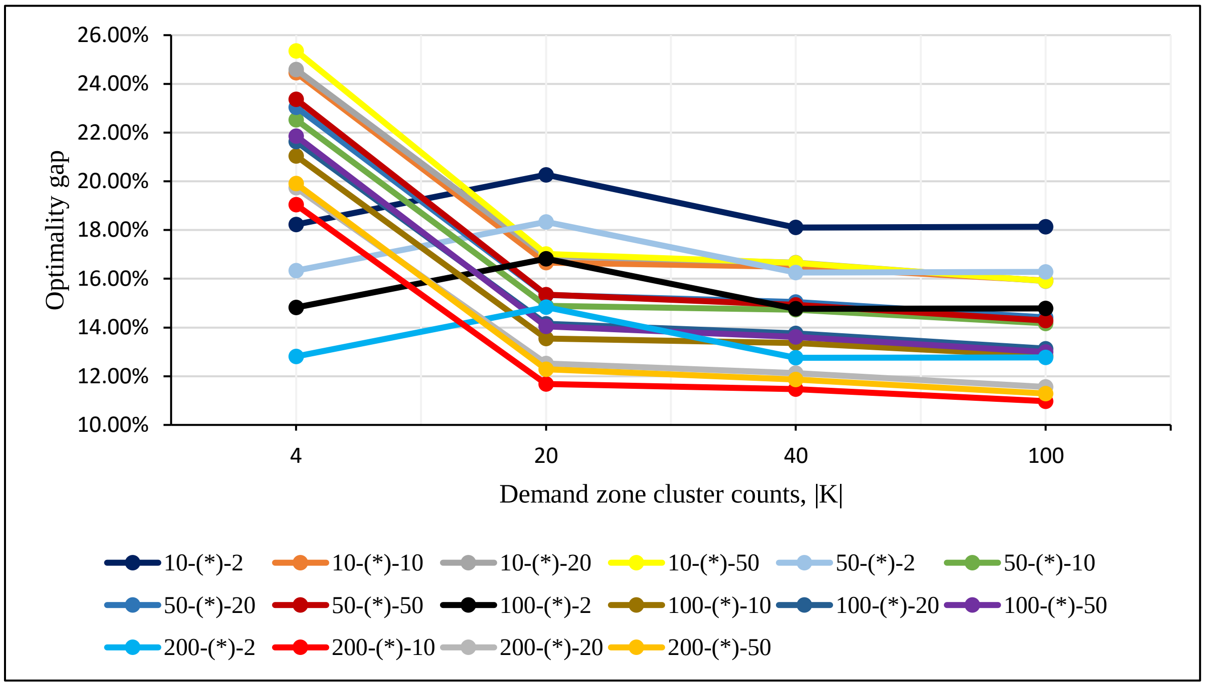


Figure 4. Optimality gap versus cluster size |K|

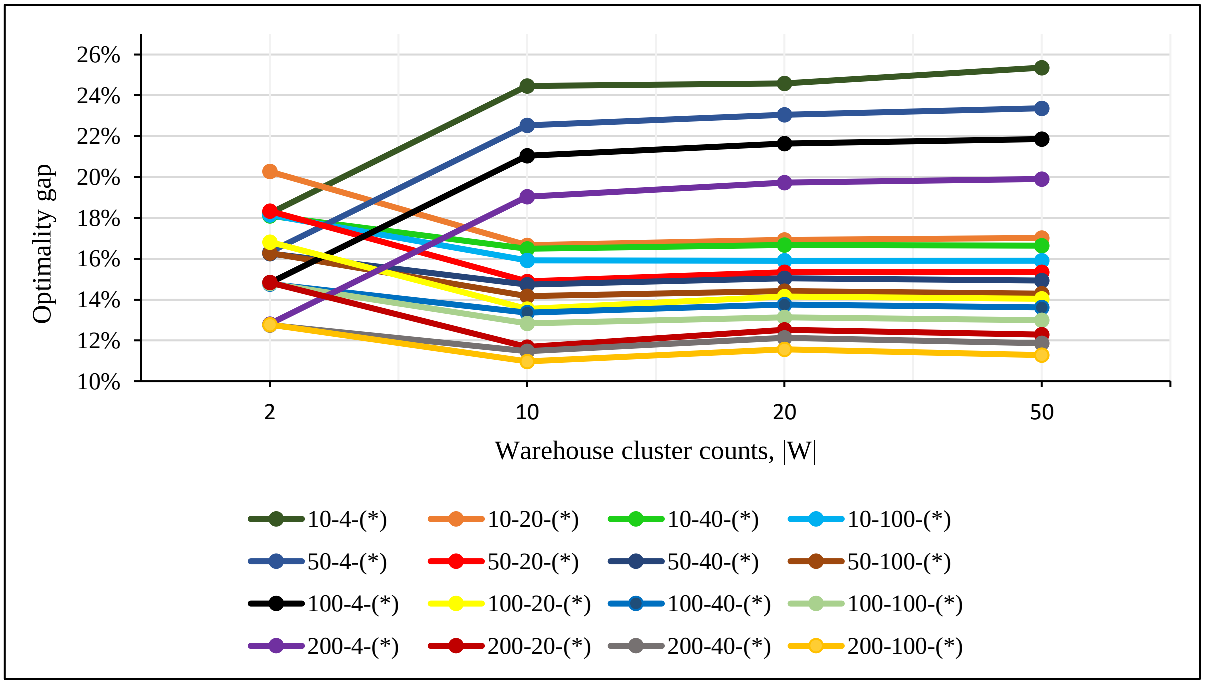


Figure 5. Optimality gap versus cluster size |W|

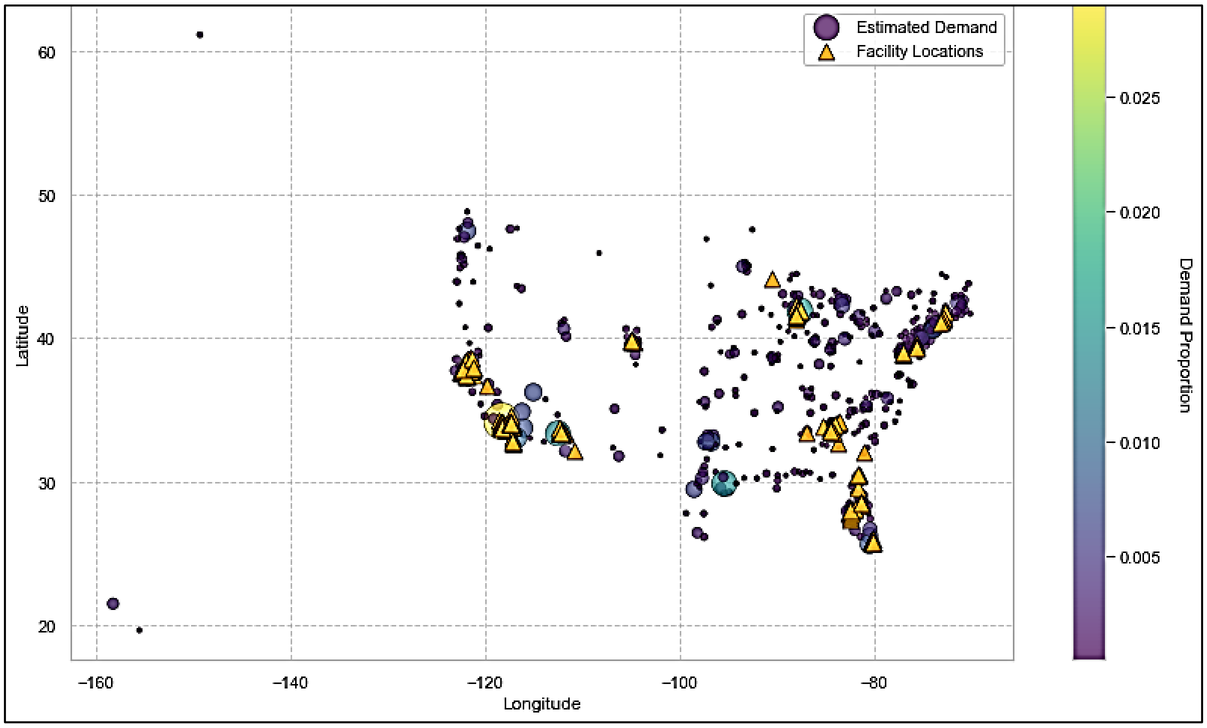


Figure 6. Distribution of demand points and fulfilment centres

6.3. Effects of cluster count on optimality gap and computation time

This section investigates how the sizes of clusters (C, K, W) influence the trade-off between the optimality gap and computation time. Our results are depicted in Figure 7, where cluster sizes are expressed as percentages of the dataset. For example, a cluster size of 1% means that the clusters (C, K, W) are 1% of the dataset size, resulting in $C = 10, K = 4, W = 2$. The optimality gap is calculated as $\frac{(Z^p_{Diss} - Z^*{}_{opt})}{Z^*{}_{opt}}$. Each aggregation stage is computed using parallel processing, i.e., each cluster is distributed on a single processor.

Larger cluster sizes result in a smaller optimality gap because they allow the aggregation model (6) to be a better approximation of the original model (1). The findings indicate that very small cluster sizes do not yield satisfactory bounds. However, for medium-sized clusters (above 10%), the cost of the solution from our algorithm approaches the cost of the optimal solution to the original problem (1). The 7% gap with a cluster size of 40% can be considered an appropriate solution compared to other cases and requires 43 minutes of computation.

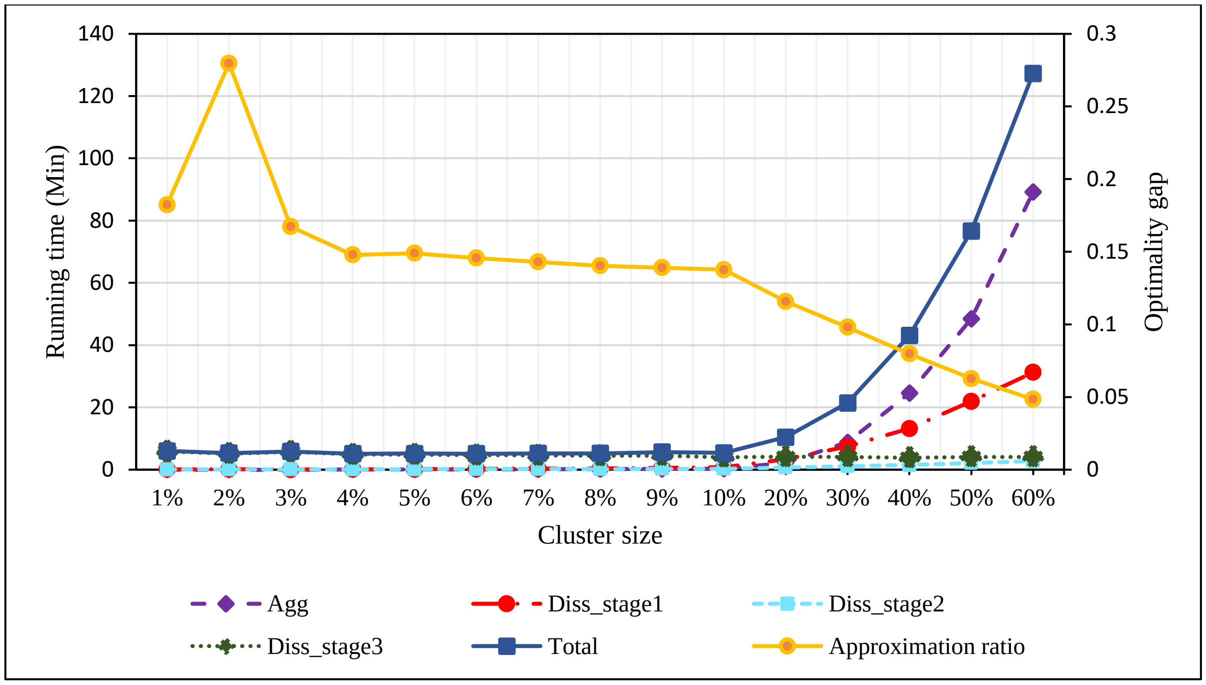


Figure 7. Optimality gap and computation time versus cluster size

On the other hand, the computation time increases with cluster size. For instance, computation time is about 5 minutes for cluster sizes up to 10%. Beyond this point, each additional 10% in cluster size roughly doubles the computation time. This is because larger cluster sizes make the computation time dominated by the aggregated model (3), requiring us to solve a larger integer programming problem, which grows exponentially in running time as dimensions increase.

However, the disaggregation stages, particularly stages 2 and 3, do not significantly increase with larger cluster sizes. This is because not only we apply the greedy algorithm for the last two stages, which significantly improve our running time but also all disaggregation stages can be executed independently for each cluster $c \in C, k \in K, w \in W$. As a result, the computation time could be reduced with parallel processors. With n parallel processors, we can speed up the computation roughly by a factor of n by distributing clusters across processors.

6.4. Effects of item size variations on optimality gap and running time

We conduct an experiment to see how the variations in the item dimensions affects the solution quality and running time. We change the amount of volume upper-bound within the range of (5,65), with the step size of 10. Therefore, the intervals span from $[1, 5]$ to $[1, 65]$. For each interval, we solve both the optimal model and the algorithm with the configuration of $(|P|, |I|, |W|) = (400, 160, 80)$.

The resulting performance of the algorithm is depicted in Figure 8. The findings indicate that a greater dispersion in item sizes corresponds to a slightly higher optimality gap. The optimality gap increases by approximately 0.1%, i.e., from 0.0791 for $v = [1, 5]$ to 0.08 for $v = [1, 65]$. The increase in optimality gap comes from the way we calculate the aggregated volume, defined as $\bar{v}_c = max_{p \in P_c}\{v_p\}$. Since the aggregated model uses the maximum size volume within each cluster $c \in C$, it becomes overly conservative, causing the solution obtained by the reduced model to deviate significantly from the optimal solution. This demonstrates how the quality of the reduced model's solution influences the subsequent stages and the final solution.

Furthermore, this result aligns with the optimality gap formulation (17h) caused by disaggregation discussed in the previous section. As expected, increasing disparity in item volumes leads to a higher value of $\bar{v}_c$ of each cluster $c \in C$, which in turn increases the optimality gap.

Furthermore, the computation time of the algorithm remains almost constant while the optimal model's computation time is more sensitive but without a clear trend, indicating the robustness of our algorithm in terms of running time. This can be seen from the fact that we employ the Greedy method to handle two disaggregation stages which is less sensitive to the volume fluctuations.

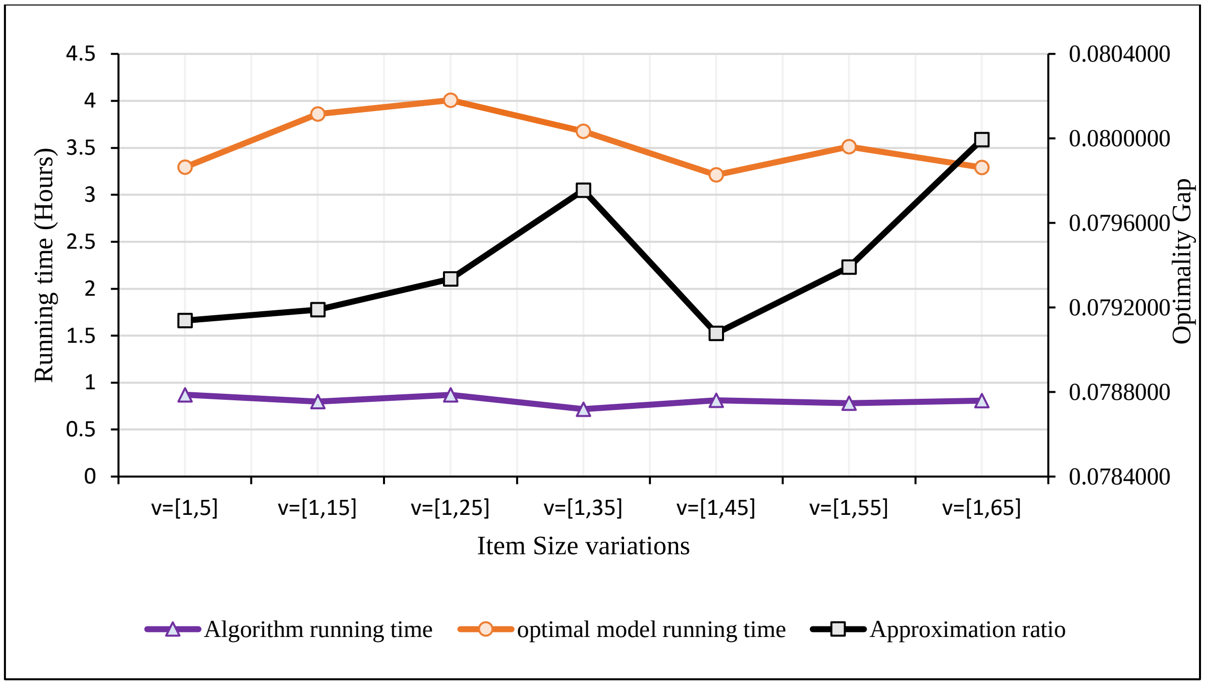


Figure 8. The effect of sparsity in item sizes on optimality gap and running time

6.5. Effects of objective function parameter ratios on algorithm performance

We conducted an experiment to evaluate the sensitivity of the algorithm's performance to the parameters of the objective function, including the unit costs of receiving, holding, and shipment. The algorithm is tested three times, each time normalizing a different parameter. For instance, while normalizing r_{pft} in the first experiment, we examine the effects of change in the ratios $\left(\frac{h_{pft}}{r_{pft}}\right)$ and $\left(\frac{\beta_{fip}}{r_{pft}}\right)$. Then, the ratios are adjusted by factors of 2 and 1/2. The results are summarized in Table 6.

The results indicate the optimality gap has a negative relationship with the ratio of $\left(\frac{h_{pft}}{r_{pft}}\right)$ and $\left(\frac{\beta_{fip}}{r_{pft}}\right)$. In other words, halving the ratio almost doubles the optimality gap, while doubling the ratio significantly reduces the gap. However, by normalizing h_{pft} in the second experiment, varying the ratios has a minimal effect on the optimality gap. The changes are slight, indicating a weak relationship between the ratio of $\left(\frac{r_{pft}}{h_{pft}}\right), \left(\frac{\beta_{fip}}{h_{pft}}\right)$ and the optimality gap. For the last experiment, the relationship is opposite to that observed for the first one. The optimality gap increases as the ratio increases, showing a positive relationship. Halving the ratios $\left(\frac{r_{pft}}{\beta_{fip}}\right)$ and $\left(\frac{h_{pft}}{\beta_{fip}}\right)$ results in the smallest optimality gap, while doubling it leads to the largest gap.

The results show that the ratio $\left(\frac{\beta_{fip}}{r_{pft}}\right)$ has a more significant impact on the optimality gap compared to the ratio $\left(\frac{r_{pft}}{h_{pft}}\right)$ and $\left(\frac{\beta_{fip}}{h_{pft}}\right)$. This highlights the importance of accurately computing the parameters β_{fip} and r_{pft}. If the ratio $\left(\frac{\beta_{fip}}{r_{pft}}\right)$ is underestimated, the optimality gap may worsen. In addition, efforts to reduce the unit receiving cost, r_{pft}, not only lower the total costs, but it also increases the ratio $\left(\frac{\beta_{fip}}{r_{pft}}\right)$, potentially leading to a smaller optimality gap. This also is consistent with the optimality gap formulation ($17h$) previously discussed, where the optimality gap is a function of r_{pft} and β_{fip}. Therefore, the algorithm is more sensitive to the ratio $\left(\frac{r_{pft}}{\beta_{fip}}\right)$ than to $\left(\frac{\beta_{fip}}{h_{pft}}\right)$ or $\left(\frac{r_{pft}}{h_{pft}}\right)$.

Table 6. Sensitivity analysis on the ratio of objective function parameters

#	Unit Cost	Ratio		Optimality Gap
1	r_{pft}	$\left(\frac{h_{pft}}{r_{pft}}\right),\left(\frac{\beta_{fip}}{r_{pft}}\right)$	*(0.5)	0.140331377
			*(1)	0.079754131
			*(2)	0.043944369
2	h_{pft}	$\left(\frac{r_{pft}}{h_{pft}}\right),\left(\frac{\beta_{fip}}{h_{pft}}\right)$	*(0.5)	0.082758264
			*(1)	0.079754131
			*(2)	0.077929631
3	β_{fip}	$\left(\frac{r_{pft}}{\beta_{fip}}\right),\left(\frac{h_{pft}}{\beta_{fip}}\right)$	*(0.5)	0.042020462
			*(1)	0.079754131
			*(2)	0.143236938

6.6. Comparison with the greedy algorithm

In this section, we compare our proposed framework with the greedy algorithm when used as the primary method to solve model (1). The experiment aims to evaluate the impact of scaling up the problem size on both the greedy algorithm and our proposed framework in terms of running time and performance.

To scale up the problem, we increase the item counts from 1000 to 20,000. We evaluate the effect of scaling up on algorithm performance by considering four cluster sizes (10%, 20%, 30%, 40%). The performance is assessed by computing the ratio of the algorithm solution Z_{Diss}^{p} to the greedy solution $Z^{G}{}_{opt}$, i.e., $\left(\frac{Z_{Diss}^{p}}{Z^{G}{}_{opt}}\right)$ for each cluster. The average ratio across the four clusters is then calculated for each item size. The results of this experiment are depicted in Figure 9.

The results indicate that while the greedy algorithm is appropriate for handling small to medium-sized problems, it struggles with large-scale problems. As the number of items increases, the greedy algorithm's running time grows substantially compared to our algorithm, and this growth rate becomes more pronounced with larger problem sizes. For example, when increasing the number of items from 2,000 to 5,000, the slope of the running time graph for the greedy algorithm is 0.03. However, this slope increases to 0.05 when scaling from 5,000 to 10,000 items, indicating an accelerating growth rate. In contrast, our proposed algorithm shows only a modest increase in running time for each cluster size. Specifically, the running time for a 40% cluster size grows by approximately 89% (from 43 minutes with 1000 items to 81 minutes with 10,000 items), whereas the greedy algorithm's running time increases by 1700% for the same item range.

To quantify the relative performance of our algorithm, we can define the ratio $\frac{Z_{Diss}^{p}}{Z^{*}{}_{opt}} = \frac{Z_{Diss}^{p}}{Z^{G}{}_{opt}} \times \frac{Z^{G}{}_{opt}}{Z^{*}{}_{opt}}$, where the first part $\frac{Z_{Diss}^{p}}{Z^{G}{}_{opt}}$ reflects the approximation error due to aggregation, and the second part $\frac{Z^{G}{}_{opt}}{Z^{*}{}_{opt}}$ represents the approximation error of the greedy algorithm itself. This experiment specifically evaluates the value of $\frac{Z_{Diss}^{p}}{Z^{G}{}_{opt}}$. The results show that the average ratio $\frac{Z_{Diss}^{p}}{Z^{G}{}_{opt}}$ increases by 4% as the problem size scales from 1,000 to 10,000 items. However, as we scale up the problem size, we observe a decreasing growth rate in the average ratio $\frac{Z_{Diss}^{p}}{Z^{G}{}_{opt}}$, which emphasizes our algorithm’s stable performance for handling large-scale models. This stability implies that our proposed framework maintains a manageable optimality gap, thereby providing a more reliable and scalable approach against greedy for large-scale problems.

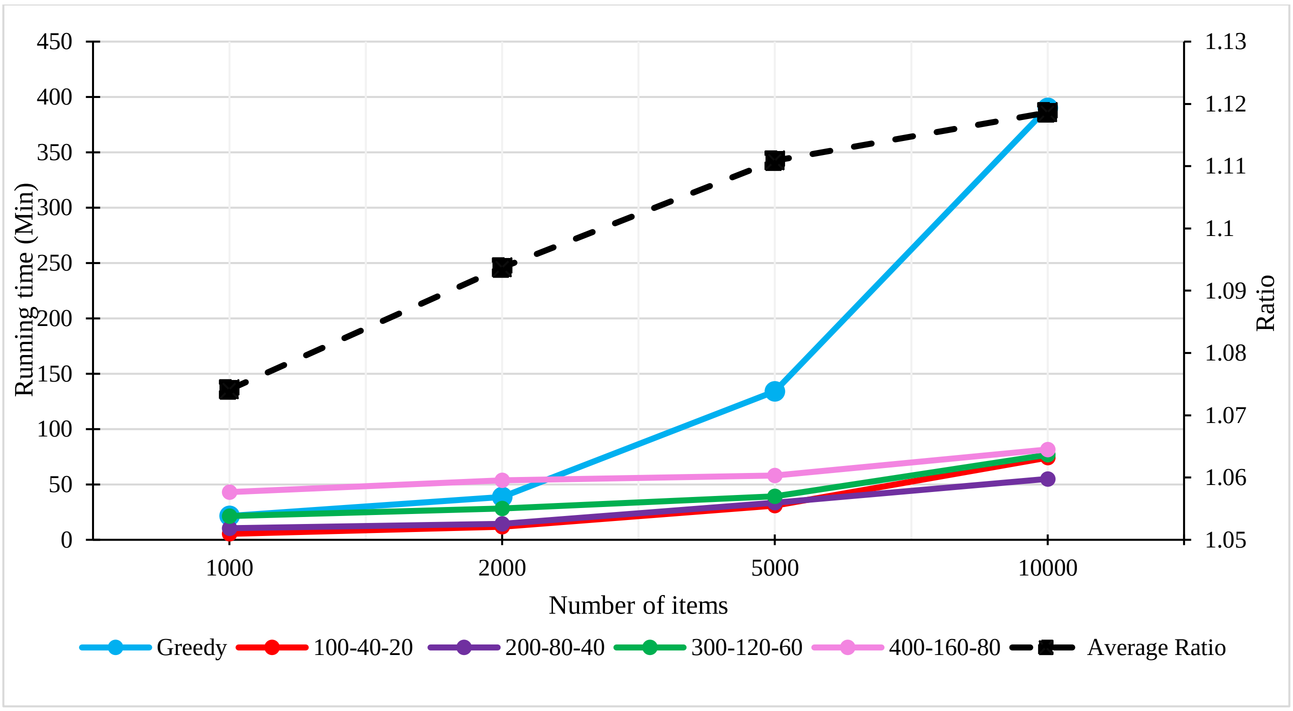


Figure 9. Comparison of the algorithm with greedy performance

6.7. Comparison with the linear programming relaxation solution

In this section, we aim to compare our algorithm with the LP relaxation model. In Section 5.4.1, we demonstrated that the solution obtained by our algorithm ensures feasibility. Now, we want to evaluate whether we can relax the integrity constraint (1f) and solve model (1) by rounding linear programming solution. The first step is to determine if the solution obtained by rounding the linear relaxation model is feasible for the online-retail inventory management problem (1). To test this, we solve model (1) by relaxing the integrality constraint (1f), and then rounding off the optimal value of each decision variable to the nearest integer point. Subsequently, we construct a new linear programming model to verify the feasibility of this rounded solution for the model (1). We define a new set of variables $(\varepsilon, \epsilon^+, \epsilon^-, \tau, \varphi)$ corresponding to each constraint (1b) to (1e) to measure the degree of violation. The model (18) is formulated as follows:

$$Z_{FS} = \underset{\varepsilon, \epsilon^{+}, \epsilon^{-}, \tau, \varphi}{Min} \left(\sum_{p,i,t} \Gamma_{pit} * \varepsilon_{pit} + \sum_{p,f,t} \Pi_{pft} * \left(\epsilon_{pft}^{+} + \epsilon_{pft}^{-}\right) + \sum_{p,t} \Upsilon_{pt} * \tau_{pt} \right.$$

$$\left. + \sum_{f,t} \Xi_{ft} * \varphi_{ft} \right) \quad (18a)$$

Subject to:

$$\sum_{f} \bar{\bar{y}}_{fipt} + \varepsilon_{pit} \geq d_{pit} \qquad \forall\, p, i, t \qquad (18b)$$

$$\bar{\bar{q}}_{pft} - \bar{\bar{q}}_{pf,t-1} - \bar{\bar{x}}_{pft} + \sum_{i} \bar{\bar{y}}_{fipt} = \epsilon_{pft}^{+} - \epsilon_{pft}^{-} \qquad \forall\, p, f, t \qquad (18c)$$

$$\sum_{f} \bar{\bar{x}}_{pft} - \tau_{pt} \leq \alpha_{pt} \qquad \forall\, p, t \qquad (18d)$$

$$\sum_{p} v_p \bar{\bar{q}}_{pf,t-1} + \sum_{p} v_p \bar{\bar{x}}_{pft} - \varphi_{ft} \leq \varpi_{ft} \qquad \forall\, f, t \qquad (18e)$$

$$\varepsilon_{pit}, \epsilon_{pft}^{+}, \epsilon_{pft}^{-}, \tau_{pt}, \varphi_{ft} \in R^{+} \qquad (18f)$$

Where $(\Gamma_{pit}, \Pi_{pft}, \Upsilon_{pt}, \Xi_{ft})$ denote the penalty units for possible violations of constraint (18b) to (18e). The values $(\bar{\bar{x}}_{pft}, \bar{\bar{y}}_{fipt}, \bar{\bar{q}}_{pft})$ represent the rounded solution derived from the linear relaxation model. If the optimal objective function value, Z_{FS}^{*}, equals zero, i.e., $Z_{FS}^{*} = 0$, the solution is feasible for model (1); otherwise, it is not. For the particular instance described in Section 6.1, the solution from linear relaxation Z_{OPT}^{R} is equal to the optimal model, i.e., $Z_{R}^{*} = Z_{OPT}^{*}$ and $Z_{FS}^{*} = 0$. However, this is not typical for most instances. According to the findings in Table 7, the solution obtained from LP relaxation is not even feasible for most cases, even though it has a lower optimality gap compared to our algorithm.

Moreover, even if the solution is feasible, LP relaxation does not resolve the issue of problem's scale; the problem still remains large-scale. For instance, solving the LP relaxation of model (1) for our specific instance takes 2.11 hours, whereas our algorithm can handle it in a few minutes, demonstrating superior computational efficiency compared to LP relaxation.

Table 7. Comparison with LP relaxation solution

#	Unit Demand	Linear relaxation objective function value (Z_R^*)	Optimal model objective function value (Z_{OPT}^*)	Z_{FS}^*
1	$\sim U[0,5]$	41,822,762.46	41,823,385.5	536.1143
2	$\sim U[0,10]$	94,085,771.75	94,087,700.5	1130.641
3	$\sim U[0,50]$	513,028,291.2	513,028,292	671.1964
4	$\sim U[0,100]$	1,041,416,914	1,041,416,917	167.799109

7. Conclusion

This study focuses on optimizing inventory management for an online retailer, aiming to minimize total costs associated with receiving, holding, and shipping. We formulate the problem as a multi-period integer programming model. Given the complexities posed by its large-scale and NP-hard nature, we propose an approximation algorithm. The contributions of this study can be summarized as follows:

- The formulation is general enough which can capture different types of variable costs and constraints. One crucial assumption addressed in our problem is the variation in item sizes, enhancing the accuracy of inventory planning while also adding to the problem's complexity.
- We propose a large-scale framework for the problem, which integrates three sets of items, warehouses, and customers as a technique for reducing the problem dimension. The proposed algorithm decomposes the problem into different tractable stages and leverages parallel computing. Each disaggregation stage can be computed independently for each cluster, and these clusters can be distributed across processors. To address the NP-hard nature of the last two stages of disaggregation, we develop a greedy algorithm capable of solving the model within polynomial time. The combination of the parallel computing and the Greedy algorithm leads to a huge reduction in running time. The effectiveness of the algorithm is demonstrated through numerical examples.
- We develop a theoretical optimality gap bound that address the effect of applying aggregation on handling the large-scale integer program. The bound provides insights into the algorithm's performance and indicates how far the algorithm can deviate from the optimal model.

Our empirical analysis reveals the impact of several factors on the solution quality, including the number of clusters, the item characteristics, and variable costs. These insights can be used to guide the choice of parameters when designing the inventory management for online-retail businesses using the proposed large-scale algorithm. The results indicate that the algorithm can find near optimal solutions within a few minutes for a medium number of clusters. Additionally, a greater dispersion in item sizes corresponds to a slightly higher optimality gap. The sensitivity analysis on the ratio of objective function parameters highlights the crucial role of the ratio of unit receiving and shipment costs, $\left(\frac{\beta_{fip}}{r_{pft}}\right)$, in controlling the optimality gap.

We also analyzed the effectiveness of our algorithm to that of the greedy algorithm when the latter is used as the primary method to solve model (1). Although the greedy algorithm offers a lower optimality gap, it faces significant challenges with large instances. Specifically, the computational time for the greedy approach exhibits an accelerating growth rate that becomes increasingly pronounced with larger problem sizes. In contrast, our algorithm demonstrates superior scalability and maintains stable performance across larger datasets. This is achieved through breaking down the problem into four manageable stages and using greedy as a subroutine, which allows our algorithm to manage large datasets effectively without significant reduction in solution quality. Thus, for large-scale inventory management problems, our algorithm proves to be more robust and efficient than the greedy algorithm.

Moreover, our comparison with LP relaxation reveals that recovering the optimal solution from LP relaxation is often impossible. Rounding the optimal values from LP relaxation often fails to ensure feasibility and, in rare cases where feasibility is achieved, the computation time remains significant. Therefore, an efficient algorithm is required to handle large-scale instances within a reasonable time frame. This underscores the importance of our algorithm's computational efficiency in addressing these challenges.

Although the proposed large-scale framework is a powerful tool for online retail inventory management, it has several limitations:

- The model does not account for shortages, whether in the form of backorders or lost sales.
- Uncertainty in parameters such as demand, which affects the solution quality, is not considered.
- The model simplifies the network to include only fulfillment centers and customers, excluding intermediate centers such as sortation centers and delivery stations between warehouses and customers.

For future work, one promising direction is to develop a customized clustering method for our specific problem. The findings indicate that standard algorithms, such as K-means, may not be an ideal technique for our case. This is because the algorithm does not give us satisfactory solutions for small size of clusters. We need to develop a customized clustering method that effectively group together items, warehouses and demand points in a way that makes the reduced model be a better approximation of the optimal model. Another approach to improve the aggregation process is to employ the iterative aggregation schemes, similar to those applied in the network design problem (Bärmann et al., 2015).

Incorporating uncertainty into parameters such as demand represents another important area for future work. Current model assumes deterministic demand, which may not always produce accurate solutions, particularly when historical data is insufficient or unreliable. Future research could focus on developing data-driven inventory optimization strategies that account for uncertainty, thereby the solution will be better aligned with real-world complexities.

References

Abouelrous, A., Gabor, A. F., & Zhang, Y. (2022). Optimizing the inventory and fulfillment of an omnichannel retailer: A stochastic approach with scenario clustering. *Computers & Industrial Engineering*, *173*, 108723. https://doi.org/10.1016/j.cie.2022.108723

Amazon. (2022, December 19). *List of Amazon Fulfillment Center Locations [2024 Updated]*. https://amzprep.com/fba-locations/

Amazon. (2024). *Our Facilities*. US About Amazon. https://www.aboutamazon.com/workplace/facilities

Amazon Distribution Network Strategy | MWPVL International. (n.d.). Retrieved 8 March 2024, from https://mwpvl.com/html/amazon_com.html

Arlotto, A., Keskin, I. N., & Wei, Y. (2023). *Online Demand Fulfillment Problem with Initial Inventory Placement: A Regret Analysis* (SSRN Scholarly Paper 4666493). https://doi.org/10.2139/ssrn.4666493

Bärmann, A., Liers, F., Martin, A., Merkert, M., Thurner, C., & Weninger, D. (2015). Solving network design problems via iterative aggregation. *Mathematical Programming Computation*, *7*(2), 189–217. https://doi.org/10.1007/s12532-015-0079-1

Bretthauer, K. M., Mahar, S., & Venakataramanan, M. A. (2010). Inventory and distribution strategies for retail/e-tail organizations. *Computers & Industrial Engineering*, *58*(1), 119–132. https://doi.org/10.1016/j.cie.2009.09.005

Chen, A. I., & Graves, S. C. (2021). Item Aggregation and Column Generation for Online-Retail Inventory Placement. *Manufacturing & Service Operations Management*, *23*(5), 1062–1076. https://doi.org/10.1287/msom.2020.0867

Chen, C. D. (2018). *Operations management in a large online retailer: Inventory, scheduling and picking* [Thesis, Massachusetts Institute of Technology]. https://dspace.mit.edu/handle/1721.1/120189

Cohen, R., Katzir, L., & Raz, D. (2006). An efficient approximation for the Generalized Assignment Problem. *Information Processing Letters*, *100*(4), 162–166. https://doi.org/10.1016/j.ipl.2006.06.003

Conforti, M., Cornuéjols, G., & Zambelli, G. (2014a). Integer Programming Models. In M. Conforti, G. Cornuéjols, & G. Zambelli (Eds.), *Integer Programming* (pp. 45–84). Springer International Publishing. https://doi.org/10.1007/978-3-319-11008-0_2

Conforti, M., Cornuéjols, G., & Zambelli, G. (2014b). Integer Programming Models. In M. Conforti, G. Cornuéjols, & G. Zambelli (Eds.), *Integer Programming* (pp. 45–84). Springer International Publishing. https://doi.org/10.1007/978-3-319-11008-0_2

Dell'Amico, M., Delorme, M., Iori, M., & Martello, S. (2019). Mathematical models and decomposition methods for the multiple knapsack problem. *European Journal of Operational Research*, *274*(3), 886–899. https://doi.org/10.1016/j.ejor.2018.10.043

DeValve, L., Wei, Y., Wu, D., & Yuan, R. (2021). Understanding the Value of Fulfillment Flexibility in an Online Retailing Environment. *Manufacturing & Service Operations Management*. https://doi.org/10.1287/msom.2021.0981

Evans, J. R. (1979). Aggregation in the generalized transportation problem. *Computers & Operations Research*, *6*(4), 199–204. https://doi.org/10.1016/0305-0548(79)90003-0

Fathollahi-Fard, A. M., Wong, K. Y., & Aljuaid, M. (2023). An efficient adaptive large neighborhood search algorithm based on heuristics and reformulations for the generalized quadratic assignment problem. *Engineering Applications of Artificial Intelligence*, *126*, 106802. https://doi.org/10.1016/j.engappai.2023.106802

Ge, D., Pan, Y., Shen, Z.-J. (Max), Wu, D., Yuan, R., & Zhang, C. (2019). Retail supply chain management: A review of theories and practices. *Journal of Data, Information and Management*, *1*(1), 45–64. https://doi.org/10.1007/s42488-019-00004-z

Geoffrion, A. M. (1976). *Customer Aggregation in Distribution Modeling,*. https://apps.dtic.mil/sti/citations/ADA033973

Govindarajan, A., Sinha, A., & Uichanco, J. (2021). Joint inventory and fulfillment decisions for omnichannel retail networks. *Naval Research Logistics (NRL)*, *68*(6), 779–794. https://doi.org/10.1002/nav.21969

Hallefjord, Å., Jörnsten, K. O., & Värbrand, P. (1993). Solving large scale generalized assignment problems—An aggregation / disaggregation approach. *European Journal of Operational Research*, *64*(1), 103–114. https://doi.org/10.1016/0377-2217(93)90011-B

Hallefjord, A., & Storoey, S. (1986). *Aggregation and disaggregation in integer programming problems*. https://www.osti.gov/etdeweb/biblio/7768114

Hitchcock, F. L. (1941). The Distribution of a Product from Several Sources to Numerous Localities. *Journal of Mathematics and Physics*, *20*(1–4), 224–230. https://doi.org/10.1002/sapm1941201224

Hoffman, A. J., Kruskal, J. B., DANTZIG, G. B., DUFFIN, R. J., FAN, K., FORD, L. R., FULKERSON, D. R., GALE, D., GOLDMAN, A. J., HELLER, I., KRUSKAL, J. B., KUHN, H. W., MILLS, H. D., THOMPSON, G. L., TOMPKINS, C. B., TUCKER, A. W., & WOLFE, P. (1956). INTEGRAL BOUNDARY POINTS OF CONVEX POLYHEDRA. In H. W. Kuhn & A. W. Tucker (Eds.), *Linear Inequalities and Related Systems. (AM-38)* (pp. 223–246). Princeton University Press; JSTOR. http://www.jstor.org/stable/j.ctt1b9x27g.16

Hübner, A. H., Kuhn, H., & Sternbeck, M. G. (2013). Demand and supply chain planning in grocery retail: An operations planning framework. *International Journal of Retail & Distribution Management*, *41*(7), 512–530. https://doi.org/10.1108/IJRDM-05-2013-0104

Jiang, D.-P., & Li, X. (2020). Order fulfilment problem with time windows and synchronisation arising in the online retailing. *International Journal of Production Research*, *59*, 1–29. https://doi.org/10.1080/00207543.2020.1721589

Kazemi, A. (2021). *New Aggregation Methods for Multicommodity Network Flow Problems: Theory and Applications to Locomotive Refueling* [Doctoral dissertation]. Monash University, Australia.

Kellerer, H., Pferschy, U., & Pisinger, D. (2004). Multidimensional Knapsack Problems. In H. Kellerer, U. Pferschy, & D. Pisinger (Eds.), *Knapsack Problems* (pp. 235–283). Springer. https://doi.org/10.1007/978-3-540-24777-7_9

Leisten, R. (1998). An LP-aggregation view on aggregationin multi-level production planning. *Annals of Operations Research*, *82*(0), 233–250. https://doi.org/10.1023/A:1018931224060

Li, X., Li, J., Aneja, Y. P., Guo, Z., & Tian, P. (2019). Integrated order allocation and order routing problem for e-order fulfillment. *IISE Transactions*, *51*(10), 1128–1150. https://doi.org/10.1080/24725854.2018.1552820

Litvinchev, I., & Tsurkov, V. (2003). Aggregated Problem and Bounds for Aggregation. In I. Litvinchev & V. Tsurkov (Eds.), *Aggregation in Large-Scale Optimization* (pp. 1–60). Springer US. https://doi.org/10.1007/978-1-4419-9154-6_1

Martello, S., & Toth, P. (1990). *Knapsack problems: Algorithms and computer implementations*. John Wiley & Sons, Inc.

Mendelssohn, R. (1980). Technical Note—Improved Bounds for Aggregated Linear Programs. *Operations Research*, *28*(6), 1450–1453. https://doi.org/10.1287/opre.28.6.1450

Rodrigue, J.-P. (2020). The distribution network of Amazon and the footprint of freight digitalization. *Journal of Transport Geography*, *88*, 102825. https://doi.org/10.1016/j.jtrangeo.2020.102825

Rogers, D. F., Plante, R. D., Wong, R. T., & Evans, J. R. (1991). Aggregation and Disaggregation Techniques and Methodology in Optimization. *Operations Research*, *39*(4), 553–582. https://doi.org/10.1287/opre.39.4.553

Saeedi, M., Parhazeh, S., Tavakkoli-Moghaddam, R., & Khalili-Fard, A. (2024). Designing a two-stage model for a sustainable closed-loop electric vehicle battery supply chain network: A scenario-based stochastic programming approach. *Computers & Industrial Engineering*, *190*, 110036. https://doi.org/10.1016/j.cie.2024.110036

Salam, M., Panahifar, F., & Byrne, P. (2016). Retail supply chain service levels: The role of inventory storage. *Journal of Enterprise Information Management*, *29*, 887–902. https://doi.org/10.1108/JEIM-01-2015-0008

Sarin, S. C., Sherali, H. D., & Kim, S. K. (2014). A branch-and-price approach for the stochastic generalized assignment problem. *Naval Research Logistics (NRL)*, *61*(2), 131–143. https://doi.org/10.1002/nav.21571

Seyedan, M., Mafakheri, F., & Wang, C. (2022). Cluster-based demand forecasting using Bayesian model averaging: An ensemble learning approach. *Decision Analytics Journal*, *3*, 100033. https://doi.org/10.1016/j.dajour.2022.100033

Shetty, C. M., & Taylor, R. W. (1987). Solving large-scale linear programs by aggregation. *Computers & Operations Research*, *14*(5), 385–393. https://doi.org/10.1016/0305-0548(87)90035-9

Torabi, S. A., Hassini, E., & Jeihoonian, M. (2015). Fulfillment source allocation, inventory transshipment, and customer order transfer in e-tailing. *Transportation Research Part E: Logistics and Transportation Review*, *79*, 128–144. https://doi.org/10.1016/j.tre.2015.04.004

U.S. Census Bureau. (2023). *QUARTERLY RETAIL E-COMMERCE SALES 4th QUARTER 2023*. https://www.census.gov/retail/mrts/www/data/pdf/ec_current.pdf

Wang, K., Li, Y., & Zhou, Y. (2022). Execution of Omni-Channel Retailing Based on a Practical Order Fulfillment Policy. *Journal of Theoretical and Applied Electronic Commerce Research*, *17*(3), Article 3. https://doi.org/10.3390/jtaer17030060

Woodcock, A. J., & Wilson, J. M. (2010). A hybrid tabu search/branch & bound approach to solving the generalized assignment problem. *European Journal of Operational Research*, *207*(2), 566–578. https://doi.org/10.1016/j.ejor.2010.05.007

Zipkin, P. H. (1980a). Bounds for aggregating nodes in network problems. *Mathematical Programming*, *19*(1), 155–177. https://doi.org/10.1007/BF01581638

Zipkin, P. H. (1980b). Bounds for Row-Aggregation in Linear Programming. *Operations Research*. https://doi.org/10.1287/opre.28.4.903

Zipkin, P. H. (1980c). Bounds on the Effect of Aggregating Variables in Linear Programs. *Operations Research*, *28*(2), 403–418. https://doi.org/10.1287/opre.28.2.403