

MIT Open Access Articles

*Item Aggregation and Column Generation
for Online-Retail Inventory Placement*

The MIT Faculty has made this article openly available. **Please share**
how this access benefits you. Your story matters.

Citation: Chen, Annie I. and Stephen C. Graves. "Item Aggregation and Column Generation for Online-Retail Inventory Placement." *Manufacturing & Service Operations Management* 23, 5 (September-October 2021): 1005–1331, C2. © 2020 INFORMS

As Published: <https://doi.org/10.1287/msom.2020.0867>

Publisher: Institute for Operations Research and the Management Sciences (INFORMS)

Persistent URL: <https://hdl.handle.net/1721.1/132952>

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Terms of use: Creative Commons Attribution-Noncommercial-Share Alike



Item Aggregation and Column Generation for Online-Retail Inventory Placement

Annie I. Chen

Marshall School of Business, University of Southern California, annieich@usc.edu,

Stephen C. Graves

Sloan School of Management, Massachusetts Institute of Technology, sgraves@mit.edu,

1. **Problem definition:** This paper studies an online retailer’s problem of choosing fulfillment centers in which to place items. We formulate the problem as a mixed-integer program that models thousands or millions of items to be placed in dozens of fulfillment centers and shipped to dozens of customer regions. The objective is to minimize the sum of shipping and fixed costs over one planning period.
2. **Academic / Practical Relevance:** A good placement plan can significantly reduce the operational cost, which is crucial for online retail businesses because they often have a low profit margin. The placement problem can be difficult to solve with existing techniques or off-the-shelf software, due to the large number of items and the fulfillment center fixed costs and capacity constraints.
3. **Methodology:** We propose a large-scale optimization framework that aggregates items into clusters, solves the cluster-level problem with column generation, and disaggregates the solution into item-level placement plans. We develop an a priori bound on the optimality gap, and also apply the framework to a numerical example that consists of 1,000,000 items.
4. **Results:** The a priori bound provides insights on how to select the appropriate aggregation criteria. For the numerical example, our framework produces a near-optimal solution in a few hours, significantly outperforming a sequential placement heuristic that approximates the status quo.
5. **Managerial Implications:** Our study provides a computationally efficient approach for solving online-retail inventory placement as well as similar large-scale optimization problems in practice.

Key words: online retail; inventory placement; column generation; aggregation

1. Introduction

1.1. Online retail background

Online retail is rapidly gaining traction as a business model. One distinct feature of online retail is that customer orders can be shipped from any fulfillment center, near or far, at different costs. In contrast, traditional brick-and-mortar retail customers are served primarily by the closest physical store. The flexibility of online retail shipping makes it possible to achieve lower costs and higher service levels, but also creates a coupling between fulfillment centers that calls for new inventory optimization techniques.

Online-retail inventory management decisions can be categorized as *strategic*, *tactical*, or *operational*, according to the planning horizon:

- *Strategic decisions* have a long-term impact, and usually involve substantial capital investments or contractual agreements that cannot be easily changed. Examples include network design (where to build new fulfillment centers, and with what storage and throughput capacities) and sourcing (what items to carry, from which vendors to buy, and the terms of procurement such as replenishment frequency, lead time, and minimum order quantities, etc).

- *Tactical decisions* are typically made every few months or on a seasonal basis, and may be periodically re-evaluated to adapt to the evolving retail landscape. Examples include inventory placement (in which fulfillment centers to store each item) and inventory replenishment (how much inventory to hold and reorder for each item).

- *Operational decisions* involve the day-to-day deployment of inventory, and often follow pre-determined policies that enable quick, real-time implementation. Examples include fulfillment (which inventory to use for an order and how to route it to the customer) and transshipment (when and how much inventory to transfer between fulfillment centers in order to rebalance the system).

All of these decisions are interdependent, but due to organizational realities and complexities, online retailers often consider them separately in a hierarchical fashion.

1.2. Problem statement

This paper focuses on the tactical decision of inventory placement. The goal is to determine a cost-minimizing *placement plan* for each item, while satisfying customer demand requirements and fulfillment center capacity constraints. We define a *placement plan* as a selected subset of fulfillment centers for placing each item, along with an estimate of the inventory levels.

The objective is to minimize the sum of two types of costs: *variable costs* and *fixed costs*. Variable costs are proportional to the volume of units handled. We focus on the outbound shipping cost, but there may also be other variable costs, such as inbound shipping and receiving costs, or the cost of labor for picking and packing items before outbound shipment. We assume that a fixed cost is charged if an item is placed in a fulfillment center, regardless of its inventory level. Fixed costs can represent actual monetary costs, such as a flat-rate inbound delivery cost charged by upstream suppliers for every delivery location, or the internal cost of splitting a single supplier shipment to multiple fulfillment centers. Fixed costs can also act as a modeling device to induce the desired level of inventory placement *sparsity*, such as the number of fulfillment centers holding an item, or the number of unique items placed in a fulfillment center. Sparsity can be an important consideration, for example, for low-demand items that cannot be split across too many fulfillment centers, or for fulfillment centers that have limited “bins” for carrying unique items.

The difficulty of the inventory placement problem arises from the large number of items and the fulfillment center fixed costs and capacity constraints. To focus on these challenges, we make the

following assumptions about the other non-placement decisions. First, since inventory replenishment is typically made upon knowing the placement plan, we omit the dynamics of replenishment and assume that demand is deterministic and constant. For the strategic decisions, we assume that network design and sourcing plans have been determined exogenously, so that cost parameters and fulfillment center capacities are given as input. For the operational decisions, we approximate the fulfillment decision with a single-period deterministic transportation problem, and assume that there is no transshipment between fulfillment centers.

Under these assumptions, we model the online-retail inventory placement problem as a mixed-integer program, with one binary variable per item-fulfillment center pair. Specifically, let $i \in \mathcal{I}$ denote the index of items, $n \in \mathcal{N}$ the fulfillment centers, and $r \in \mathcal{R}$ the regions from which customer demand originate. The decision variables that characterize the placement plan are:

- $x_{nr}^i \in [0, 1]$, the fraction of region r , item i demand that is shipped from fulfillment center n ;
- $y_n^i \in \{0, 1\}$, where $y_n^i = 1$ if fulfillment center n carries item i , and $y_n^i = 0$ otherwise.

For a given planning period (e.g., one week), for every $i \in \mathcal{I}$, $n \in \mathcal{N}$, and $r \in \mathcal{R}$, we assume that the following parameters are provided as input:

- b_n , the throughput capacity of fulfillment center n , i.e., the number of units it can receive and ship during the entire planning period;
- c_{nr}^i , the per-unit outbound shipping cost of item i from fulfillment center n to region r , as well as any other variable costs such as inbound shipping and handling costs at the fulfillment center;
- f_n^i , the fixed cost of carrying any amount of item i in fulfillment center n at any point during the planning period; and
- d_r^i , the demand for item i in region r for the planning period. Thus, $d_r^i x_{nr}^i$ is the throughput of inventory flowing from fulfillment center n to region r . Also, let $d^i = \sum_{r \in \mathcal{R}} d_r^i$ denote the total demand of item i across all regions.

Then, we can formulate the *online-retail inventory placement problem* as follows:

$$\begin{aligned}
 & \min_{\mathbf{x}, \mathbf{y}} \quad \sum_{i \in \mathcal{I}} \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} c_{nr}^i d_r^i x_{nr}^i + \sum_{i \in \mathcal{I}} \sum_{n \in \mathcal{N}} f_n^i y_n^i & (\mathbf{P}_1) \\
 \text{subject to} \quad & \sum_{n \in \mathcal{N}} x_{nr}^i = 1, & \forall r \in \mathcal{R}, \forall i \in \mathcal{I}, & (1) \\
 & \sum_{i \in \mathcal{I}} \sum_{r \in \mathcal{R}} d_r^i x_{nr}^i \leq b_n, & \forall n \in \mathcal{N}, & (2) \\
 & \sum_{r \in \mathcal{R}} d_r^i x_{nr}^i \leq d^i y_n^i, & \forall n \in \mathcal{N}, \forall i \in \mathcal{I}, & (3) \\
 & x_{nr}^i \geq 0, \quad y_n^i \in \{0, 1\}, & \forall n \in \mathcal{N}, \forall r \in \mathcal{R}, \forall i \in \mathcal{I}.
 \end{aligned}$$

The objective function is the sum of shipping and fixed costs over the entire planning period. The set of constraints (1) is for demand satisfaction; (2) is for throughput capacity (to model storage or other types of capacity constraints, one can adapt the units of b_n , d_r^i and c_{nr}^i accordingly); and

(3) ensures that for every given i and n , if $x_{nr}^i > 0$ for any $r \in \mathcal{R}$, then $y_n^i = 1$ (this is the most computationally efficient formulation compared to other alternatives, according to a small study in Chen 2017).

The choice to model customers as a set of regions \mathcal{R} is a deliberate one. The set need not include every physical address where customers reside. Instead, it suffices to consolidate customers into a finite set of regions, as long as customers in the same region incur the same shipping cost from every fulfillment center. (This consolidation is also a type of aggregation, though of a different nature than the item aggregation that is the subject of this paper. We simply assume that \mathcal{R} is given as input.) We can also capture multiple “classes” of customers, such as those requesting two-day versus same-day shipping, by creating a separate collection of customer regions for each class.

We remark that our model is agnostic as to how exactly the outbound transportation network operates. For instance, an online retailer might rely solely on third party carriers like UPS. In this case, c_{nr}^i reflects the negotiated rate that the retailer pays the carrier for the delivery of an item, plus any additional costs that the retailer incurs in handling a shipment. The set \mathcal{R} in this case is determined based on the carrier’s rate table.

Another way that online retailers perform fulfillment is with their own transportation network and resources, such as those which Amazon and JD.com have developed in recent years. In these operations, items travel from fulfillment centers to retailer-controlled regional facilities from which the last-mile delivery to customers occurs. In the process, items may pass through intermediate facilities that sort items by destination and consolidate items into fewer packages for the same customer order. In this case, \mathcal{R} represents the set of regional delivery facilities, and c_{nr}^i is the sum of an item’s average transportation cost from the fulfillment center to the regional facility, the average last-mile delivery cost, and any handling costs at each facility in the process.

Solving (\mathbf{P}_1) directly appears impractical even with state-of-the-art commercial software. For instance, Chen (2017) reports that a problem with $|\mathcal{N}| = 88$ fulfillment centers, $|\mathcal{R}| = 99$ regions and $|\mathcal{I}| = 1000$ items solved by the Gurobi Optimizer (version 7.0.1) reaches an optimality gap of around 6-7% after 5 hours on a personal laptop, and the gap does not improve much after an additional 5 hours. For problems with more items, even constructing the formulation becomes prohibitive in terms of computer memory. Thus, a more efficient approach is needed.

We propose a large-scale optimization framework that aggregates items into clusters, solves the cluster-level problem with column generation, and disaggregates the solution into item-level placement plans. Our framework finds a near-optimal solution in a computationally efficient manner.

The remainder of the paper is organized as follows: Section 2 reviews related work. Section 3 presents the mathematical formulations of our framework. Section 4 develops an a priori bound on the optimality gap. Section 5 applies our network to a large numerical example with 1,000,000 items. Section 6 concludes with future directions.

2. Related work

2.1. Online retail inventory management

Our review of online retail literature is focused on related work under the three categories of inventory decisions: strategic, tactical, and operational. This paper belongs to the tactical category.

The intent of our work is similar to that of Xu (2005), who studies the problem of allocating inventory across fulfillment centers, and characterizes the allocation plan that minimizes the outbound shipping cost for a small problem (placing 2 units of a low-demand item, with stochastic demand, in 2 or 3 fulfillment centers). However, addressing inventory allocation and stochastic demand simultaneously makes it difficult to solve larger instances of this model. Instead, we follow a different modeling approach, proposed by Chen (2017), that decomposes the problem into a *placement* problem with deterministic demand for multiple items, and a *replenishment* problem with stochastic demand for a single item. The placement problem is the subject of this paper, while the replenishment problem can be solved subsequently. This decomposition maintains tractability for large, realistic problem instances, and also allows for more complex conditions such as capacity constraints and fixed costs.

Some methods for solving the subsequent replenishment problem have been proposed. Acimovic (2012) and Acimovic and Graves (2017) consider the case with identical replenishment lead times, periodic review, and lost sales. They develop the *projected base-stock policy*, a heuristic that outperforms the status quo base-stock policy in its ability to mitigate persistent inventory imbalance and reduce outbound shipping costs. Chen (2017) further assumes non-identical lead times, and uses discrete optimization via simulation to find low-cost replenishment policies.

Regarding operational decisions, much of the research interest has been focused on inventory fulfillment. It is worth pointing out that, while approximating the outbound shipping cost with a transportation problem is sufficient for our purposes, it is not always the best practice to ship items from the cheapest fulfillment center, as might be implied by a transportation problem. In fact, Acimovic (2012) and Acimovic and Graves (2015) show that this so-called *myopic* fulfillment policy is not optimal, and they propose a computationally-tractable heuristic policy that performs well in practice. Other recent approaches to the fulfillment problem include Torabi et al. (2015), who use Benders decomposition to jointly optimize the fulfillment and transshipment decisions, and Lei et al. (2016), who study the benefit of jointly optimizing fulfillment and pricing.

Regarding strategic decisions, Agatz et al. (2008) review network design problems in online retail, and Swaminathan and Tayur (2003) review sourcing problems. A noteworthy strategic trend is to integrate online and brick-and-mortar businesses to form an *omni-channel* setting; see Agatz et al. (2008), Hübner et al. (2016), and Govindarajan et al. (2017) for surveys. Though beyond the scope of this paper, omni-channel inventory problems are often very large in scale, and could be a promising direction to extend our optimization framework.

2.2. Facility location and network flow

If we omit the capacity constraints in (\mathbf{P}_1) , the problem separates by item and reduces to the *simple plant location problem (SPLP)* for each item, i.e., a transportation problem on a bipartite graph, with two sets of nodes representing fulfillment centers and customer regions, respectively, and fixed costs on the fulfillment center nodes. This problem has been shown to be NP-hard (Krarup and Pruzan 1983). With capacity constraints, the problem is a generalization of the *capacitated facility location problem*, reviewed by Sridharan (1995). It is also a special case of the *three-echelon warehouse location problem* of Pirkul and Jayaraman (1998), which in turn is a special case of the *multi-commodity fixed-charge network flow problem*, surveyed by Gendron (2011).

Existing solution methods for facility location and network flow problems fall into three categories: (1) Variants of branch-and-bound (e.g., Gendron 2011); (2) Benders decomposition (e.g., Costa et al. 2012); (3) Heuristics (e.g., Hewitt et al. 2010 and references therein). In most cases, the fixed cost is charged per facility regardless of the number of commodities (items) using it, so the number of binary variables is independent of the number of items. In contrast, for the online-retail inventory placement problem, there is one binary variable per commodity-facility (item-fulfillment center) pair, making it much larger than the scale for which existing methods are intended.

2.3. Aggregation and column generation

The large-scale optimization framework we propose is a heuristic that integrates *item aggregation* and *column generation*. While each is well established in the literature as a separate technique, the integration and synergy of the two is relatively under-explored and, we believe, under-exploited.

Our approach fits within the aggregation framework described by Rogers et al. (1991), who also provide an excellent summary of existing work; see also Francis (1985) and Litvinchev and Tsurkov (2013) for surveys. Aggregation is widely recognized as a technique for reducing computational complexity, and has been applied to contexts involving a large number of items. For example, Hax and Meal (1975) propose a *hierarchical production planning* model in which items with a common production setup are aggregated as a *family* for scheduling, and families with similar seasonal demand patterns and production rates are aggregated as a *type* for capacity planning.

We make two main contributions to the research on aggregation. First, we derive an a priori optimality gap bound for the aggregation of binary variables. Most of the existing literature focuses on continuous variables, using duality theory to bound the optimality gap that arises from the aggregation of variables; e.g., Zipkin (1980a) and Zipkin (1980b). There are relatively few results on bounds for integer variables. Moreover, studies on integer variables focus primarily on the aggregation of constraints (see Rogers et al. 1991 for a review) rather than that of variables. Closest in spirit to our work is Hallefjord and Storøy (1990), who aggregate binary variables

and relax the integrality of aggregate variables to obtain duality-based bounds, the quality of which depends on valid inequalities identified with heuristics. However, their problem consists of purely binary variables, whereas our problem has a mix of binary and continuous variables. Also, they bound the cost difference between the original and aggregated problems while leaving out the disaggregation part, whereas we bound the cost difference between the original and the disaggregated final solutions. Our bound does not invoke duality theory, but instead utilizes a series of intermediate formulations as stepping stones, carefully designed based on problem structure.

The second contribution is that we integrate aggregation with column generation to produce a synergy of computational complexity reduction. Most existing work focus only on the benefit of aggregation in reducing the number of variables. In addition to this, we take advantage of aggregation to move from an integer program to a continuous reformulation that can be solved by column generation. Specifically, columns in the reformulation represent placement plans, and the continuous variables represent “fractions” of an aggregated cluster of items that use those placement plans. The notion of “fractions” is only meaningful for clusters, which can be split up accordingly because they consist of multiple items; fractions do not make sense for non-aggregated items, which are indivisible. Thus, aggregation and column generation go hand in hand to make our framework computationally efficient. Our definition of columns is closest to that of Klose and Drexl (2005), in which columns correspond to integer solutions of the capacitated facility location problem. However, they only consider a single commodity, so there is no notion of aggregation, and they employ a rounding heuristic to make use of the fractional solutions of columns. In contrast, we solve a disaggregation optimization problem that transforms fractional columns into integer solutions for individual items.

3. A large-scale optimization framework

Our proposed framework consists of three steps. First, items are aggregated into clusters. Next, an approximation of the cluster-level problem is solved via column generation. Lastly, the cluster-level solution is disaggregated into item-level placement plans. The subsequent sections describe each step in detail, and end with the computational advantages of this framework.

To distinguish between cluster- and item-level problems, we henceforth denote all cluster parameters and variables with a “bar” and with the index j , e.g., $\bar{d}^j, \bar{x}_{nr}^j$. Also, let the bold font denote a vector of decision variables, e.g., \mathbf{x} represents $(x_{nr}^i)_{n \in \mathcal{N}, r \in \mathcal{R}, i \in \mathcal{I}}$, and $\bar{\mathbf{y}}^j$ represents $(\bar{y}_n^j)_{n \in \mathcal{N}}$.

3.1. Aggregation

The goal of aggregation is to create a partition of items, denoted $\mathcal{I} = \bigcup_{j \in \mathcal{J}} \mathcal{I}_j$, where each $j \in \mathcal{J}$ is called a *cluster*. The ideal outcome of aggregation, loosely speaking, is that items with similar optimal placement plans are partitioned in the same cluster. Borrowing the characterization of Rogers et al. (1991), our aggregation method has the following features:

Cluster entity. The entities to cluster are the items.

Similarity measure and cluster procedure. These two features are closely related, as the choice of one directly impacts the choice of the other. In Section 5, we give a numerical example that uses k-means as the cluster procedure, and for similarity measure, uses the Euclidean distance on a weighted vector of item attributes, with weights chosen to minimize the a priori bound derived in Section 4. In practice, similarity measures and cluster procedures should be chosen according to the business context. For instance, existing item categorization may naturally lend itself to some partition. Also, if historical demand data identifies certain items that are likely to be purchased together, it may be a good idea to aggregate them in the same cluster, so that they are predisposed to being shipped from the same fulfillment center in one package.

Level of clustering (number of clusters). The number of clusters should also be selected by the user based on available computing resources. As we shall see shortly, the cluster count controls the trade-off between optimality gap and computation time. Having more clusters results in a smaller gap and greater computational effort. Fortunately, the most computationally intensive parts of our framework (column generation and disaggregation) are decomposable by cluster, so if parallel processors are available, more clusters can be accommodated without sacrificing computation time.

Method of combination. To obtain the cluster-level placement problem, we assume that items in the same cluster take on identical decision variables values in (\mathbf{P}_1) , i.e.,

$$x_{nr}^i = \bar{x}_{nr}^j, \quad y_n^i = \bar{y}_n^j, \quad \text{for every } (i, j) \in \mathcal{I} \times \mathcal{J} \text{ such that } i \in \mathcal{I}_j. \quad (4)$$

where $\bar{x}_{nr}^j, \bar{y}_n^j$ are the decision variables for cluster $j \in \mathcal{J}$. The cluster-level placement problem looks essentially identical to (\mathbf{P}_1) :

$$\begin{aligned} \min_{\bar{\mathbf{x}}, \bar{\mathbf{y}}} \quad & \sum_{j \in \mathcal{J}} \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} \bar{c}_{nr}^j \bar{d}_r^j \bar{x}_{nr}^j + \sum_{j \in \mathcal{J}} \sum_{n \in \mathcal{N}} \bar{f}_n^j \bar{y}_n^j & (\mathbf{P}_2) \\ \text{subject to} \quad & \sum_{n \in \mathcal{N}} \bar{x}_{nr}^j = 1, & \forall r \in \mathcal{R}, \forall j \in \mathcal{J}, \\ & \sum_{j \in \mathcal{J}} \sum_{r \in \mathcal{R}} \bar{d}_r^j \bar{x}_{nr}^j \leq b_n, & \forall n \in \mathcal{N}, \\ & \sum_{r \in \mathcal{R}} \bar{d}_r^j \bar{x}_{nr}^j \leq \bar{d}^j \bar{y}_n^j, & \forall n \in \mathcal{N}, \forall j \in \mathcal{J}, \\ & \bar{x}_{nr}^j \geq 0, \quad \bar{y}_n^j \in \{0, 1\}, & \forall n \in \mathcal{N}, \forall r \in \mathcal{R}, \forall j \in \mathcal{J}. \end{aligned}$$

It is straightforward to verify that for every cluster $j \in \mathcal{J}$, the demand and cost parameters are

$$\bar{d}_r^j = \sum_{i \in \mathcal{I}_j} d_r^i, \quad \bar{d}^j = \sum_{i \in \mathcal{I}_j} d^i, \quad \bar{c}_{nr}^j = \frac{\sum_{i \in \mathcal{I}_j} c_{nr}^i d_r^i}{\sum_{i \in \mathcal{I}_j} d_r^i}, \quad \bar{f}_n^j = \sum_{i \in \mathcal{I}_j} f_n^i. \quad (5)$$

(This is similar to the *fixed-weight combination* procedure common to many aggregation methods, except that the weights for fixed cost parameters do not form a convex combination.)

The optimal cost of (\mathbf{P}_2) is an upper bound for the optimal cost of (\mathbf{P}_1) , because the former is obtained by adding the constraints (4) to the latter.

3.2. Column generation

Next, we approximate (\mathbf{P}_2) with a column-based reformulation that can be solved with column generation. Define a *column* for cluster $j \in \mathcal{J}$ to be an *integral* placement plan in which each customer region is fully served by a single fulfillment center, i.e., the \bar{x}_{nr}^j decision variables of (\mathbf{P}_2) are either 0 or 1. For every cluster $j \in \mathcal{J}$, let

$$X_j = \left\{ (\bar{\mathbf{x}}^j, \bar{\mathbf{y}}^j) \mid \sum_{n \in \mathcal{N}} \bar{x}_{nr}^j = 1, \sum_{r \in \mathcal{R}} \bar{d}_r^j \bar{x}_{nr}^j \leq \bar{d}_n^j \bar{y}_n^j, \bar{x}_{nr}^j \in \{0, 1\}, \bar{y}_n^j \in \{0, 1\}, \forall n \in \mathcal{N}, \forall r \in \mathcal{R} \right\}$$

be the set of all such integral placement plans.

To develop the column-based reformulation, we enumerate placement plans in the set X_j and label them with an index set called \mathcal{K}_j . That is, we denote the k -th placement plan of cluster j by $\bar{\mathbf{x}}^j(k) = (\bar{x}_{nr}^j(k))_{n \in \mathcal{N}, r \in \mathcal{R}}$ and $\bar{\mathbf{y}}^j(k) = (\bar{y}_n^j(k))_{n \in \mathcal{N}}$, where $k \in \mathcal{K}_j$ and $(\bar{\mathbf{x}}^j(k), \bar{\mathbf{y}}^j(k)) \in X_j$. Each of these placement plans corresponds to a column, so \mathcal{K}_j can also be thought of as a set of columns.

The *cost* of a column $k \in \mathcal{K}_j, j \in \mathcal{J}$ is defined as its objective function value, $\bar{c}_k^j = \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} \bar{c}_{nr}^j \bar{d}_r^j \bar{x}_{nr}^j(k) + \sum_{n \in \mathcal{N}} \bar{f}_n^j \bar{y}_n^j(k)$. The column's *capacity usage* at fulfillment center $n \in \mathcal{N}$ is defined as $\bar{u}_{kn}^j = \sum_{r \in \mathcal{R}} \bar{d}_r^j \bar{x}_{nr}^j(k)$. The column-based reformulation of (\mathbf{P}_2) finds a cost-minimizing convex combination of columns for each cluster while respecting fulfillment center capacities:

$$\begin{aligned} & \min_{\bar{\lambda}} \quad \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}_j} \bar{c}_k^j \bar{\lambda}_k^j & (\mathbf{P}_3) \\ \text{subject to} \quad & \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}_j} \bar{u}_{kn}^j \bar{\lambda}_k^j \leq b_n, & \forall n \in \mathcal{N}, \\ & \sum_{k \in \mathcal{K}_j} \bar{\lambda}_k^j = 1, & \forall j \in \mathcal{J}, \\ & \bar{\lambda}_k^j \geq 0, & \forall k \in \mathcal{K}_j, \forall j \in \mathcal{J}. \end{aligned}$$

The decision variables $\bar{\lambda}_k^j$ are the column weights in the convex combinations.

The reformulation (\mathbf{P}_3) is an approximation of (\mathbf{P}_2) in the sense that their objective functions are different, but their solution spaces are equivalent. Intuitively, the integral placement plans that correspond to columns in (\mathbf{P}_3) can be thought of as a set of “bases” that span the solution space of (\mathbf{P}_2) . We show in Online Appendix A that every solution $(\bar{\mathbf{x}}^j, \bar{\mathbf{y}}^j)$ of (\mathbf{P}_2) can be converted into a solution $\bar{\lambda}$ (along with $\bar{\mathbf{x}}^j(k), \bar{\mathbf{y}}^j(k)$ corresponding to each $\bar{\lambda}_k^j$) of (\mathbf{P}_3) (Theorem 2), and vice versa (Theorem 3), according to the following relationships:

$$\bar{x}_{nr}^j = \sum_{k \in \mathcal{K}_j} \bar{x}_{nr}^j(k) \bar{\lambda}_k^j, \quad \forall n \in \mathcal{N}, r \in \mathcal{R}, j \in \mathcal{J}, \quad (6)$$

$$\bar{y}_n^j = \max\{\bar{y}_n^j(k) \mid k \in \mathcal{K}_j, \bar{\lambda}_k^j > 0\}, \quad \forall n \in \mathcal{N}, j \in \mathcal{J}. \quad (7)$$

The optimal cost of (\mathbf{P}_3) is a lower bound on that of (\mathbf{P}_2) because the former can be thought of as a relaxation of the latter in the binary variables for fixed costs. Indeed, the objective function of (\mathbf{P}_3) is no greater than that of (\mathbf{P}_2) for equivalent solutions that satisfy (6) and (7), i.e.,

$$\sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}_j} \bar{c}_k^j \bar{\lambda}_k^j \leq \sum_{j \in \mathcal{J}} \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} \bar{c}_{nr}^j \bar{d}_r^j \bar{x}_{nr}^j + \sum_{j \in \mathcal{J}} \sum_{n \in \mathcal{N}} \bar{f}_n^j \bar{y}_n^j.$$

To see this, we apply the definition of \bar{c}_k^j and note that the shipping cost terms on both sides are identical by definition. Thus, it suffices to show that the fixed cost terms satisfy

$$\sum_{j \in \mathcal{J}} \sum_{n \in \mathcal{N}} \bar{f}_n^j \left(\sum_{k \in \mathcal{K}_j} \bar{\lambda}_k^j \bar{y}_n^j(k) \right) \leq \sum_{j \in \mathcal{J}} \sum_{n \in \mathcal{N}} \bar{f}_n^j \bar{y}_n^j.$$

This inequality holds due to the fact that, if $\sum_{k \in \mathcal{K}_j} \bar{\lambda}_k^j \bar{y}_n^j(k) > 0$, then it must be no greater than 1, since it is a convex combination of $\bar{y}_n^j(k) \in \{0, 1\}$. Furthermore, $\sum_{k \in \mathcal{K}_j} \bar{\lambda}_k^j \bar{y}_n^j(k) > 0$ indicates that there exists some $k \in \mathcal{K}_j$ such that $\bar{\lambda}_k^j > 0$ and $\bar{y}_n^j(k) = 1$; therefore, $\bar{y}_n^j = 1$ by (7), and thus $\sum_{k \in \mathcal{K}_j} \bar{\lambda}_k^j \bar{y}_n^j(k) \leq 1 = \bar{y}_n^j$.

An important property of (\mathbf{P}_3) is that, by the fundamental theorem of linear programming and assuming the existence of an optimal solution, there exists an optimal solution with at most $|\mathcal{N}| + |\mathcal{J}|$ nonzero decision variables. In other words, across all clusters, the total number of columns used in the optimal solution is at most $|\mathcal{N}| + |\mathcal{J}|$. As a result, if $|\mathcal{J}| \gg |\mathcal{N}|$, then most clusters use only one column. This latter observation has significant implications on disaggregation and the optimality gap bound, as we shall see in the sections below.

The main advantage of (\mathbf{P}_3) over (\mathbf{P}_2) is that the former is a linear program which can be solved at scale using column generation. This well-established technique works by decomposing the problem into a *restricted master problem* that contains a limited subset of columns, and *column generation subproblems* that iteratively identify new columns to be added to the master problem so as to potentially improve its objective cost. We give the problem formulations here and leave the details to Algorithm B.1 in Online Appendix B.

Restricted master problem. The restricted master problem is identical to (\mathbf{P}_3) except that, in iteration t , the set of columns \mathcal{K}_j for cluster $j \in \mathcal{J}$ is replaced by a subset $\mathcal{K}_j^t \subset \mathcal{K}_j$:

$$\min_{\lambda} \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}_j^t} \bar{c}_k^j \bar{\lambda}_k^j \tag{RMP}$$

$$\text{subject to } \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}_j^t} \bar{u}_{kn}^j \bar{\lambda}_k^j \leq b_n, \quad \forall n \in \mathcal{N}, \tag{8}$$

$$\sum_{k \in \mathcal{K}_j^t} \bar{\lambda}_k^j = 1, \quad \forall j \in \mathcal{J}, \tag{9}$$

$$\bar{\lambda}_k^j \geq 0, \quad \forall k \in \mathcal{K}_j^t, \forall j \in \mathcal{J}.$$

We set the initial subset for (\mathbf{RMP}) to

$$\mathcal{K}_j^0 = \bigcup_{n' \in \mathcal{N}} \{k \in \mathcal{K}_j \mid x_{n'r}^j(k) = y_{n'}^j(k) = 1 \text{ and } \bar{x}_{nr}^j(k) = \bar{y}_n^j(k) = 0 \ \forall n \in \mathcal{N}, n \neq n'\}, \quad \forall j \in \mathcal{J}, \tag{10}$$

i.e., the set of columns in which the cluster uses only one fulfillment center. An initial feasible solution can be obtained by setting $\bar{\lambda}_k^j = \frac{b_n}{\sum_{n' \in \mathcal{N}} b_{n'}}$ for the column $k \in \mathcal{K}_j^0$ that uses $n \in \mathcal{N}$.

Column generation subproblems. The column generation subproblem for cluster $j \in \mathcal{J}$ is

$$\min_{\bar{x}^j, \bar{y}^j} \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} (\bar{c}_{nr}^j - p_n) \bar{d}_r^j \bar{x}_{nr}^j + \sum_{n \in \mathcal{N}} \bar{f}_n^j \bar{y}_n^j - q_j \tag{CG}_j$$

subject to $(\bar{\mathbf{x}}^j, \bar{\mathbf{y}}^j) \in X_j$,

where $\{p_n\}_{n \in \mathcal{N}}$ and $\{q_j\}_{j \in \mathcal{J}}$ are the dual variables of constraints (8) and (9), respectively. The goal is to find the column with the minimum reduced cost, defined as $\bar{c}_k^j - \sum_{n \in \mathcal{N}} p_n \bar{u}_{kn}^j - q_j$, which is a bound on how much the column may decrease the objective of **(RMP)** if it were added to \mathcal{K}_j^t . If no columns with negative reduced costs can be found for any cluster, then the algorithm terminates with an optimal solution for **(P₃)**. It is also possible to terminate the algorithm at any iteration (Algorithm B.1, line 3) and obtain a feasible solution that is not necessarily optimal.

3.3. Disaggregation

Once we have a solution for **(RMP)** from the previous step, we can disaggregate each cluster independently. Given a cluster $j \in \mathcal{J}$, let $(\bar{\lambda}_k^j)_{k \in \mathcal{K}_j}$ be the solution of **(RMP)**, and let $\mathcal{K}_j^+ = \{k \in \mathcal{K}_j \mid \bar{\lambda}_k^j > 0\}$ be the set of columns used. We define the cluster's *allocated capacity* as

$$\bar{u}_n^j = \sum_{k \in \mathcal{K}_j^+} \bar{u}_{kn}^j \bar{\lambda}_k^j + \frac{\bar{d}^j}{\sum_{j' \in \mathcal{J}} \bar{d}^{j'}} \Delta b_n, \quad j \in \mathcal{J}, n \in \mathcal{N}.$$

The first term is the capacity that is actually utilized by the cluster. The second term, which is optional, is the additional capacity distributed to the cluster in proportion to its total demand, where $\Delta b_n = b_n - \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}_j^+} \bar{u}_{kn}^j \bar{\lambda}_k^j$ is the amount of unused capacity, i.e., the slack in constraint (8), for fulfillment center $n \in \mathcal{N}$.

To disaggregate cluster $j \in \mathcal{J}$, we solve the following placement problem for items in the cluster, which has the same objective and constraints as the original placement problem **(P₁)** except that the (allocated) fulfillment center capacities are given by $\bar{u}_n^j, n \in \mathcal{N}$, and that the placement plans are restricted to convex combinations of columns in \mathcal{K}_j^+ :

$$\begin{aligned} \min_{\lambda^i, \mathbf{x}^i, \mathbf{y}^i} \quad & \sum_{i \in \mathcal{I}_j} \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} c_{nr}^i d_r^i x_{nr}^i + \sum_{i \in \mathcal{I}_j} \sum_{n \in \mathcal{N}} f_n^i y_n^i & (\mathbf{D}_j) \\ \text{subject to} \quad & x_{nr}^i = \sum_{k \in \mathcal{K}_j^+} \bar{x}_{nr}^j(k) \lambda_k^i, & \forall n \in \mathcal{N}, \forall r \in \mathcal{R}, \forall i \in \mathcal{I}_j, \\ & \sum_{i \in \mathcal{I}_j} \sum_{r \in \mathcal{R}} d_r^i x_{nr}^i \leq \bar{u}_n^j, & \forall n \in \mathcal{N}, \\ & \sum_{r \in \mathcal{R}} d_r^i x_{nr}^i \leq d^i y_n^i, & \forall n \in \mathcal{N}, \forall i \in \mathcal{I}_j, \\ & \sum_{k \in \mathcal{K}_j^+} \lambda_k^i = 1, & \forall i \in \mathcal{I}_j, \\ & \lambda_k^i \geq 0, & \forall i \in \mathcal{I}_j, \forall k \in \mathcal{K}_j^+, \\ & y_n^i \in \{0, 1\}, & \forall n \in \mathcal{N}, \forall i \in \mathcal{I}_j. \end{aligned}$$

The decision variables λ_k^i are the weights in the convex combination of columns, and they dictate the values of the other decision variables x_{nr}^i, y_n^i through the first constraint.

Although **(D_j)** resembles **(P₁)**, it is much more computationally tractable, as it only concerns the subset of items \mathcal{I}_j , and its solution space is limited to that of the convex combination of columns

in \mathcal{K}_j^+ . The problem is easy to solve when $|\mathcal{K}_j^+|$ is small. Indeed, when $|\mathcal{K}_j^+| = 1$ (recall that this is the case for most clusters when $|\mathcal{J}| \gg |\mathcal{N}|$), then the disaggregation problem is trivial—the one column used by the cluster simply becomes the placement plan of all the items.

The disaggregation problem is always feasible because we can take $\lambda_k^i = \bar{\lambda}_k^j$ for all $i \in \mathcal{I}_j$ and $k \in \mathcal{K}_j^+$ as the initial feasible solution. This solution has the same shipping cost as the solution of (\mathbf{P}_3) by definitions of \bar{c}_k^j and \bar{c}_{nr}^j , a useful property when deriving theoretical bounds in Section 4.

When (\mathbf{D}_j) is solved to optimality, most items use only one column. (A formal proof is beyond the scope of this paper, but loosely speaking, by a common argument for optimality, if two items have positive weights on the same set of columns, then a better solution can be constructed by trading off their weights in favor of the item-column pair that has the lowest cost, as long as the demand and fulfillment center capacities permit.) Thus, (\mathbf{D}_j) can also be interpreted as an optimal assignment problem that assigns each item in the cluster to a column in \mathcal{K}_j^+ , except for a few items that must be split up across columns to satisfy capacity constraints.

That said, solving (\mathbf{D}_j) to optimality may be time-consuming in practice because it is a mixed integer program. In our numerical example in Section 5.3, we find that a small a posteriori optimality gap of less than 1% can already be achieved by terminating the solver as soon as it produces a solution that outperforms the initial feasible solution described above.

3.4. Computational advantages

The integration of item aggregation and column generation has a two-fold effect on computational efficiency. First, the number of decision variables is reduced as we go from the item-level problem (\mathbf{P}_1) to the cluster-level problem (\mathbf{P}_2) . Secondly, the problem becomes easier to solve as we go from the mixed integer program (\mathbf{P}_2) to the linear program (\mathbf{P}_3) that is solvable by column generation.

In addition, the framework naturally lends itself to parallelization. The most computationally intensive parts are the column generation subproblems (\mathbf{CG}_j) and disaggregation problems (\mathbf{D}_j) because they involve integer variables (aggregation can usually be done efficiently with heuristics (such as k-means), and (\mathbf{RMP}) is a relatively easy linear program). Fortunately, (\mathbf{CG}_j) and (\mathbf{D}_j) can be computed independently for each cluster. Therefore, with p parallel processors, we can speed up the computation roughly by a factor of p by distributing clusters across processors.

Early termination before reaching optimality is also possible when solving (\mathbf{RMP}) and (\mathbf{D}_j) . This can be a useful feature when computing time and resources are limited.

4. A priori bound on optimality gap

This section provides an upper bound on the optimality gap achieved by our large-scale optimization framework. It is an *a priori* bound in the sense that it can be computed from input data

without actually carrying out the optimization. An important application of this bound is to guide aggregation by highlighting certain trade-offs, as demonstrated numerically in Section 5.

We adopt the following mathematical notation. Given an optimization problem (\mathbf{P}) , let $v(\mathbf{P})|_{(\mathbf{x}, \mathbf{y})}$ (or $v(\mathbf{P})|_{\bar{\lambda}}$ if (\mathbf{P}) is in column-based form) denote the cost of a feasible solution (\mathbf{x}, \mathbf{y}) (or $\bar{\lambda} = (\lambda_k^j)_{k \in \mathcal{K}_j, j \in \mathcal{J}}$), and let $v^*(\mathbf{P})$ denote its optimal cost.

THEOREM 1. *Given a set of clusters \mathcal{J} that form a partition of the items $\mathcal{I} = \bigcup_{j \in \mathcal{J}} \mathcal{I}_j$, the objective cost difference between the final solution $(\mathbf{x}^i, \mathbf{y}^i)$ obtained by the large-scale optimization framework and the optimal cost $v^*(\mathbf{P}_1)$ of the original placement problem is bounded by*

$$\sum_{j \in \mathcal{J}} v(\mathbf{D}_j)|_{(\mathbf{x}^i, \mathbf{y}^i)} - v^*(\mathbf{P}_1) \leq \frac{1}{2} \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}_j} d^i \left(\sum_{r \in \mathcal{R}} |\bar{\delta}_r^j - \delta_r^i| \right) \left(\max_{n \in \mathcal{N}, r \in \mathcal{R}} c_{nr}^i - \min_{n \in \mathcal{N}, r \in \mathcal{R}} c_{nr}^i \right) \quad (11)$$

$$+ \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}_j} d^i \sum_{r \in \mathcal{R}} \bar{\delta}_r^j \max_{n \in \mathcal{N}} (\bar{c}_{nr}^j - c_{nr}^i) \quad (12)$$

$$+ \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}_j} \sum_{n \in \mathcal{N}} \bar{f}_n^j \left| \frac{d^i}{\bar{d}^j} - \frac{f_n^i}{\bar{f}_n^j} \right| \quad (13)$$

$$+ |\mathcal{N}| \max_{j \in \mathcal{J}} \left(\sum_{n \in \mathcal{N}} \bar{f}_n^j \right), \quad (14)$$

where $\delta_r^i = d_r^i/d^i$ and $\bar{\delta}_r^j = \bar{d}_r^j/\bar{d}^j$ are normalized demand distributions for items and clusters, respectively, and the other parameters are defined as in (\mathbf{P}_1) and (5).

Proof. We decompose the objective cost difference into several terms:

$$\begin{aligned} \sum_{j \in \mathcal{J}} v(\mathbf{D}_j)|_{(\mathbf{x}^i, \mathbf{y}^i)} - v^*(\mathbf{P}_1) &= [v(\mathbf{Q})|_{(\mathbf{x}', \mathbf{y}')} - v^*(\mathbf{P}_1)] + [v(\mathbf{P}_3)|_{\bar{\lambda}'} - v(\mathbf{Q})|_{(\mathbf{x}', \mathbf{y}')}] + [v^*(\mathbf{P}_3) - v(\mathbf{P}_3)|_{\bar{\lambda}'}] \\ &\quad + \left[\sum_{j \in \mathcal{J}} v(\mathbf{D}_j)|_{(\mathbf{x}_0^i, \mathbf{y}_0^i)} - v^*(\mathbf{P}_3) \right] + \sum_{j \in \mathcal{J}} \left[v(\mathbf{D}_j)|_{(\mathbf{x}^i, \mathbf{y}^i)} - v(\mathbf{D}_j)|_{(\mathbf{x}_0^i, \mathbf{y}_0^i)} \right]. \end{aligned} \quad (15)$$

In this decomposition, (\mathbf{Q}) is an auxiliary problem, defined below; $(\mathbf{x}', \mathbf{y}')$ is a feasible solution of (\mathbf{Q}) ; $\bar{\lambda}'$ is the column-based solution that is equivalent to $(\mathbf{x}', \mathbf{y}')$ and is feasible for (\mathbf{P}_3) , as shown in Lemma 2 below; and $(\mathbf{x}_0^i, \mathbf{y}_0^i)$ is the initial feasible solution of (\mathbf{D}_j) described in Section 3.3.

The auxiliary problem (\mathbf{Q}) in the first two terms of (15) has the following definition:

$$(\mathbf{Q}) \text{ is identical to } (\mathbf{P}_1) \text{ except that } d_r^i \text{ is replaced with } d^i \cdot \bar{\delta}_r^j, \text{ for every } i \in \mathcal{I}_j, j \in \mathcal{J}. \quad (\mathbf{Q})$$

This problem (\mathbf{Q}) acts as a bridge between the item-level problem (\mathbf{P}_1) and the cluster-level problem (\mathbf{P}_3) . It is an item-level problem with the same *total* demand per item as in (\mathbf{P}_1) , but its *regional* demand has a normalized distribution of $(\bar{\delta}_r^j)_{r \in \mathcal{R}}$ instead of $(\delta_r^i)_{r \in \mathcal{R}}$, i.e., each item has the same normalized demand distribution as that of its cluster in (\mathbf{P}_3) .

To prove Theorem 1, we develop upper bounds on each of the five terms on the right hand side of (15). Since we are minimizing, the third and fifth terms on the right-hand side of (15) are non-positive and hence have an upper bound of zero. The remainder of the proof focuses on the first, second and fourth terms and is divided into Lemmas 1 to 3 below, summarized as follows:

- We bound $v(\mathbf{Q})|_{(\mathbf{x}', \mathbf{y}')} - v^*(\mathbf{P}_1)$ using a feasible solution $(\mathbf{x}', \mathbf{y}')$ of (\mathbf{Q}) that utilizes fulfillment centers in the same way as does the optimal solution of (\mathbf{P}_1) , but serves the normalized demand distribution of (\mathbf{Q}) rather than that of (\mathbf{P}_1) . This bound is derived in Lemma 1 and results in the term (11) on the right-hand side of the inequality.

- We bound $v(\mathbf{P}_3)|_{\bar{\lambda}'} - v(\mathbf{Q})|_{(\mathbf{x}', \mathbf{y}')} - v^*(\mathbf{P}_3)$ using a feasible solution $\bar{\lambda}'$ of (\mathbf{P}_3) that is the column-based equivalent of $(\mathbf{x}', \mathbf{y}')$. This is possible because (\mathbf{Q}) and (\mathbf{P}_3) have the same demand distribution and differ only in the objective. This bound is derived in Lemma 2 and results in (12) and (13).

- We bound $\sum_{j \in \mathcal{J}} v(\mathbf{D}_j)|_{(\mathbf{x}_0^i, \mathbf{y}_0^i)} - v^*(\mathbf{P}_3)$ by the fact that $(\mathbf{x}_0^i, \mathbf{y}_0^i)$ has the same shipping cost as that of final solution of (\mathbf{P}_3) , which allows us to simply bound their worst-case difference in fixed costs. This bound is derived in Lemma 3 and results in the term (14).

□

According to Theorem 1, the magnitude of the first three terms depends on the homogeneity of the items within each cluster, while the fourth term depends on the similarity of the size of the clusters. For example,

- The value of (11) is small when $\sum_{r \in \mathcal{R}} |\bar{\delta}_r^j - \delta_r^i|$ is small, which is the case when items in the same cluster have similar normalized demand distributions;

- The value of (12) is small when $\max_{n \in \mathcal{N}} (\bar{c}_{nr}^j - c_{nr}^i)$ is small, which is the case when items in the same cluster have similar shipping cost parameters;

- The value of (13) is small when $\left| \frac{d^i}{\bar{d}^j} - \frac{f_n^i}{\bar{f}_n^j} \right|$ is small, which is the case when items in the same cluster have a similar fixed-cost-to-demand ratio, f_n^i/d^i , for every fulfillment center;

- The value of (14) is small when $\max_{j \in \mathcal{J}} (\sum_{n \in \mathcal{N}} \bar{f}_n^j)$ is small, which is the case when fixed cost parameters are evenly distributed across clusters.

We complete the remainder of this section with the lemmas cited in in Theorem 1, as well as two lower bounds on $v^*(\mathbf{P}_1)$.

LEMMA 1. *Given a set of clusters \mathcal{J} , there exists a feasible solution $(\mathbf{x}', \mathbf{y}')$ of (\mathbf{Q}) such that*

$$v(\mathbf{Q})|_{(\mathbf{x}', \mathbf{y}')} - v^*(\mathbf{P}_1) \leq \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}_j} d^i v^*(\mathbf{MCF}_i) \quad (16)$$

$$\leq \frac{1}{2} \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}_j} d^i \left(\sum_{r \in \mathcal{R}} |\bar{\delta}_r^j - \delta_r^i| \right) \left(\max_{n \in \mathcal{N}, r \in \mathcal{R}} c_{nr}^i - \min_{n \in \mathcal{N}, r \in \mathcal{R}} c_{nr}^i \right), \quad (17)$$

where $v^*(\mathbf{MCF}_i)$ is the optimal objective value of the following Min-Cost Flow problem:

$$\begin{aligned} & \min_{\Delta_{rr'}} \sum_{(r, r') \in \mathcal{R} \times \mathcal{R}, r \neq r'} g_{rr'}^i \Delta_{rr'} & (\mathbf{MCF}_i) \\ \text{subject to} & \sum_{r' \in \mathcal{R}, r' \neq r} (\Delta_{r'r} - \Delta_{rr'}) = \bar{\delta}_r^j - \delta_r^i & \forall r \in \mathcal{R}, \\ & \Delta_{rr'} \geq 0 & \forall r, r' \in \mathcal{R}, r \neq r', \end{aligned}$$

with $g_{rr'}^i = \max_{n \in \mathcal{N}} (c_{nr'}^i - c_{nr}^i)$ being the maximum shipping cost difference from a fulfillment center for the pair of regions $(r, r') \in \mathcal{R} \times \mathcal{R}, r \neq r'$.

Proof. Our goal is to construct a new placement plan $(\mathbf{x}', \mathbf{y}')$ that maintains the same placement of inventory at each fulfillment center as the optimal solution of (\mathbf{P}_1) , but satisfies the demand of (\mathbf{Q}) instead. The new placement plan has the same fixed cost as the optimal solution of (\mathbf{P}_1) , so we focus on bounding their shipping cost difference.

For every item $i \in \mathcal{I}$, the per-unit shipping cost difference can be bounded by solving the minimum-cost network flow problem (\mathbf{MCF}_i) , in which there are $|\mathcal{R}|$ nodes representing the regions, and a complete graph of directed arcs between each pair of nodes for a total of $|\mathcal{R}| \times (|\mathcal{R}| - 1)$ arcs. The net flow into node $r \in \mathcal{R}$ is $\bar{\delta}_r^j - \delta_r^i$, the difference in normalized demand distribution in the region. The flow on arc $(r, r') \in \mathcal{R} \times \mathcal{R}, r \neq r'$ is denoted $\Delta_{rr'}$ and represents the amount of inventory that needs to be redirected and shipped to region r' instead of region r , in order to satisfy the demand difference between (\mathbf{Q}) and (\mathbf{P}_1) . The redirected inventory may come from multiple fulfillment centers depending on availability, but to obtain a bound, we set the unit flow cost on each arc at its worst value, independent of the placement decisions made for (\mathbf{P}_1) ; that is, the cost associated with the arc (r, r') is given by $g_{rr'}^i = \max_{n \in \mathcal{N}} (c_{nr'}^i - c_{nr}^i)$, the increase in shipping cost when redirecting the shipment of a unit inventory from region r to region r' from the worst possible fulfillment center. Multiplying $v^*(\mathbf{MCF}_i)$ by the item's total demand d^i , and summing up over $i \in \mathcal{I}$, yields the bound on $v(\mathbf{Q})|_{(\mathbf{x}', \mathbf{y}')} - v^*(\mathbf{P}_1)$ given in (16).

If there is a lot of items, solving a network flow problem for each of them may not be desirable. Therefore, we further relax the bound to obtain the expression (17), which only requires simple calculations from problem data. This latter bound is based on the observation that there exists an optimal solution of (\mathbf{MCF}_i) in which no flow passes through any node with $\bar{\delta}_r^j - \delta_r^i = 0$; a formal proof is given in Lemma 4 in Online Appendix C. This observation implies that there exists an optimal solution of (\mathbf{MCF}_i) such that the only positive flow is from the set $\mathcal{R}^- = \{r \in \mathcal{R} \mid \bar{\delta}_r^j - \delta_r^i < 0\}$ to the set $\mathcal{R}^+ = \{r \in \mathcal{R} \mid \bar{\delta}_r^j - \delta_r^i > 0\}$. Therefore, $v^*(\mathbf{MCF}_i)$ is bounded above by $\sum_{r \in \mathcal{R}^+} (\bar{\delta}_r^j - \delta_r^i) \cdot \max_{r' \in \mathcal{R}^-} g_{r'r}^i$, since for every node $r \in \mathcal{R}^+$, the total amount of incoming flow is $\bar{\delta}_r^j - \delta_r^i$, and the maximum cost of an arc with incoming flow to r is $\max_{r' \in \mathcal{R}^-} g_{r'r}^i$. Similarly, $v^*(\mathbf{MCF}_i)$ is also bounded above by $\sum_{r' \in \mathcal{R}^-} |\bar{\delta}_{r'}^j - \delta_{r'}^i| \cdot \max_{r \in \mathcal{R}^+} g_{r'r}^i$, the total maximum cost incurred by outgoing flow from nodes in \mathcal{R}^- . Combining the above, we have

$$\begin{aligned} v^*(\mathbf{MCF}_i) &\leq \min \left\{ \sum_{r \in \mathcal{R}^+} (\bar{\delta}_r^j - \delta_r^i) \cdot \max_{r' \in \mathcal{R}^-} g_{r'r}^i, \sum_{r' \in \mathcal{R}^-} |\bar{\delta}_{r'}^j - \delta_{r'}^i| \cdot \max_{r \in \mathcal{R}^+} g_{r'r}^i \right\} \\ &\leq \frac{1}{2} \left(\sum_{r \in \mathcal{R}} |\bar{\delta}_r^j - \delta_r^i| \right) \left(\max_{r \in \mathcal{R}^+} \max_{r' \in \mathcal{R}^-} g_{r'r}^i \right) = \frac{1}{2} \left(\sum_{r \in \mathcal{R}} |\bar{\delta}_r^j - \delta_r^i| \right) \left(\max_{r \in \mathcal{R}^+} \max_{r' \in \mathcal{R}^-} \max_{n \in \mathcal{N}} (c_{nr'}^i - c_{nr}^i) \right) \end{aligned}$$

$$\leq \frac{1}{2} \left(\sum_{r \in \mathcal{R}} |\bar{\delta}_r^j - \delta_r^i| \right) \left(\max_{n \in \mathcal{N}, r \in \mathcal{R}} c_{nr}^i - \min_{n \in \mathcal{N}, r \in \mathcal{R}} c_{nr}^i \right), \quad \forall i \in \mathcal{I},$$

where the second line uses the fact that $\sum_{r \in \mathcal{R}^+} \bar{\delta}_r^j - \delta_r^i = \sum_{r' \in \mathcal{R}^-} \delta_{r'}^i - \bar{\delta}_{r'}^j = \frac{1}{2} \sum_{r \in \mathcal{R}} |\bar{\delta}_r^j - \delta_r^i|$ and the definition of $g_{r'r}^i$, and the last line relaxes the need to know exactly which regions belong to \mathcal{R}^+ or \mathcal{R}^- . \square

Remark on Lemma 1. The looser bound (17) appears sufficient for our purpose of guiding item aggregation, and it has the advantage of being easier to compute than (16). However, it is worth pointing out that (16) is tighter when the demand distributions of (\mathbf{Q}) and (\mathbf{P}_1) differ only in regions that are located close to each other, so that the demand patterns “look” very similar if plotted geographically on a map. When this is the case, (\mathbf{MCF}_i) is able to produce a smaller bound by concentrating $\Delta_{rr'}$ on regions (r, r') that are close to each other, for which $g_{rr'}^i$ is small because r and r' have similar distances to every fulfillment center, i.e., $c_{nr}^i \approx c_{nr'}^i$ for every $n \in \mathcal{N}$. On the other hand, (17) does not capture such geographical information.

LEMMA 2. *For every feasible solution $(\mathbf{x}', \mathbf{y}')$ of (\mathbf{Q}) , there exists a feasible solution $\bar{\lambda}'$ of (\mathbf{P}_3) that satisfies*

$$v(\mathbf{P}_3)|_{\bar{\lambda}'} - v(\mathbf{Q})|_{(\mathbf{x}', \mathbf{y}')} \leq \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}_j} \left[d^i \sum_{r \in \mathcal{R}} \bar{\delta}_r^j \max_{n \in \mathcal{N}} (\bar{c}_{nr}^j - c_{nr}^i) + \sum_{n \in \mathcal{N}} \bar{f}_n^j \left| \frac{d^i}{\bar{d}^j} - \frac{f_n^i}{f_n^j} \right| \right].$$

Proof. Recall that every item in (\mathbf{Q}) has the same normalized demand distribution as that of the cluster in (\mathbf{P}_3) to which the item belongs. Thus, to convert a placement plan $(\mathbf{x}', \mathbf{y}')$ of (\mathbf{Q}) into a column solution $\bar{\lambda}$ of (\mathbf{P}_3) , simply let $\bar{\lambda}_k^j = d^i / \bar{d}^j$ for the column $k \in \mathcal{K}_j$ that corresponds to the placement plan of item $i \in \mathcal{I}_j$. If the placement plan of i is not an integral solution that corresponds directly to a column, express it as a convex combination (which is always possible by Theorem 2 in Online Appendix A) and adjust the $\bar{\lambda}_k^j$ values accordingly. The column solution thus defined has a cost of

$$v(\mathbf{P}_3)|_{\bar{\lambda}'} = \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}_j} \frac{d^i}{\bar{d}^j} \left(\sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} \bar{c}_{nr}^j \bar{d}_r^j x'_{nr} + \sum_{n \in \mathcal{N}} \bar{f}_n^j y'_n \right)$$

The objective function of (\mathbf{P}_3) involves cluster-level cost parameters \bar{c}_{nr}^j and \bar{f}_n^j , while the objective function of (\mathbf{Q}) involves item-level parameters c_{nr}^i and f_n^i . Therefore, their difference is

$$\begin{aligned} v(\mathbf{P}_3)|_{\bar{\lambda}'} - v(\mathbf{Q})|_{(\mathbf{x}, \mathbf{y})} &= \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}_j} \left[\frac{d^i}{\bar{d}^j} \left(\sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} \bar{c}_{nr}^j \bar{d}_r^j x'_{nr} + \sum_{n \in \mathcal{N}} \bar{f}_n^j y'_n \right) - \left(\sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} c_{nr}^i (d^i \bar{\delta}_r^j) x'_{nr} + \sum_{n \in \mathcal{N}} f_n^i y'_n \right) \right] \\ &= \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}_j} \left[\sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} (\bar{c}_{nr}^j - c_{nr}^i) (d^i \bar{\delta}_r^j) x'_{nr} + \sum_{n \in \mathcal{N}} \left(\bar{f}_n^j \frac{d^i}{\bar{d}^j} - f_n^i \right) y'_n \right], \\ &\leq \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}_j} \left[d^i \sum_{r \in \mathcal{R}} \bar{\delta}_r^j \max_{n \in \mathcal{N}} (\bar{c}_{nr}^j - c_{nr}^i) + \sum_{n \in \mathcal{N}} \bar{f}_n^j \left| \frac{d^i}{\bar{d}^j} - \frac{f_n^i}{f_n^j} \right| \right], \end{aligned}$$

where the first line is by definition; the second line reorganizes the terms; and the inequality is due to the fact that $\sum_{n \in \mathcal{N}} x'_{nr} = 1$ for every $r \in \mathcal{R}$, and that $y'_n \leq 1$ for every $n \in \mathcal{N}$ and every $i \in \mathcal{I}$. \square

LEMMA 3. Let $(\mathbf{x}_0^i, \mathbf{y}_0^i), i \in \mathcal{I}_j$, be the initial feasible solution of (\mathbf{D}_j) given in Section 3.3. Then,

$$\sum_{j \in \mathcal{J}} v(\mathbf{D}_j)|_{(\mathbf{x}_0^i, \mathbf{y}_0^i)} - v^*(\mathbf{P}_3) \leq |\mathcal{N}| \max_{j \in \mathcal{J}} \left(\sum_{n \in \mathcal{N}} \bar{f}_n^j \right).$$

Proof. As noted in Section 3.3, $v(\mathbf{D}_j)|_{(\mathbf{x}_0^i, \mathbf{y}_0^i)}$ and $v^*(\mathbf{P}_3)$ have the same shipping cost. They also have the same fixed cost if the cluster j only uses one column, i.e., if $|\mathcal{K}_j^+| = 1$. There are at most $|\mathcal{N}|$ clusters with $|\mathcal{K}_j^+| > 1$, according to Section 3.2. For those clusters, the fixed cost of (\mathbf{P}_3) is nonnegative so can be omitted in the bound, while the fixed cost of (\mathbf{D}_j) is at most $\sum_{i \in \mathcal{I}_j} \sum_{n \in \mathcal{N}} f_n^i = \sum_{n \in \mathcal{N}} \bar{f}_n^j$, in the worst case that every item uses all fulfillment centers after disaggregation. (In fact, by the same logic, we could obtain a tighter bound by summing up the top $|\mathcal{N}|$ values of $\sum_{n \in \mathcal{N}} \bar{f}_n^j$ among all clusters $j \in \mathcal{J}$, but we use the bound stated in the lemma for simplicity.) \square

We close this section with two lower bounds on $v^*(\mathbf{P}_1)$, which can be combined with Theorem 1 to yield a percentage optimality gap. The first is obtained by solving (\mathbf{P}_1) without capacity constraints, so that it decomposes into the so-called *simple plant location problem (SPLP)* for every item. Such single-item problems are easy to optimize, but if a large number of items makes it computationally prohibitive, the following looser bound can be used:

$$v^*(\mathbf{P}_1) \geq \sum_{i \in \mathcal{I}} \left(\sum_{r \in \mathcal{R}} d_r^i \min_{n \in \mathcal{N}} c_{nr}^i + \min_{n \in \mathcal{N}} f_n^i \right). \quad (18)$$

The first term on the right-hand side reflects the lowest possible shipping cost (using one fulfillment center per item per region), and the second term the lowest possible fixed cost (using one fulfillment center per item), required to satisfy the demand, without capacity constraints. A numerical comparison of these two lower bounds is given in Section 5.

5. Numerical experiments

We apply our large-scale optimization framework to a placement problem with 1,000,000 items. Section 5.1 details the numerical example on which our method is tested. We then present four computational experiments:

- In Section 5.2, we study how aggregation affects the a priori optimality gap bound;
- In Section 5.3, we study how the number of clusters $|\mathcal{J}|$ affects the a posteriori optimality gap bound and computation time;
- In Section 5.4, we study how fixed cost parameters affect the sparsity of placement plans;
- In Section 5.5, we compare our framework to a simple sequential placement heuristic that approximates what is currently done in practice.

All experiments are implemented in Python 2.7.12 and its interface to Gurobi 7.0.1. Computation times are based on a single trial using one dual-core processor with a CPU of 2.10 GHz and run by the Linux Ubuntu 16.04 operating system. The input datasets, computer programs, and further implementation details are all available on the first author's website.

5.1. Numerical example

Although real datasets are difficult to come by in the highly competitive and confidential business of online retail, we are able to create a synthetic numerical example from publicly available data. The aim is to mimic a real online retail inventory system in terms of fulfillment center features, customer regions, shipping costs, and item attributes, as detailed below.

Fulfillment centers. We create a realistic network based on Amazon’s fulfillment centers in the United States, taken from an unofficial public dataset (MWPVL International Inc. 2015). There were $|\mathcal{N}| = 88$ fulfillment centers as of December 2015. The fulfillment center capacities $(b_n)_{n \in \mathcal{N}}$ are computed by setting the total capacity to be 1% higher than the total demand of all items; that is, we assume that $\sum_{n=1}^N b_n = 1.01 \cdot \sum_{i \in \mathcal{I}} d^i$. (See Chen 2017 for cases with more than 1% excess capacity.) The total capacity is distributed across the fulfillment centers in proportion to its square footage given in the dataset. Although the dataset notes that some fulfillment centers are dedicated to specific product categories, we make no such distinctions and simply allow every fulfillment center to carry all kinds of items.

Customer regions. The customer regions are created from 2010 US Census data (United States Census Bureau 2010). The raw data contains 3221 counties for continental US, which we group geographically into $|\mathcal{R}| = 98$ regions, each with a population and longitude-latitude coordinates.

Figure 1 maps the location of fulfillment centers (triangles) and customer regions (circles), with markers sized in proportion to the capacity and the population, respectively.

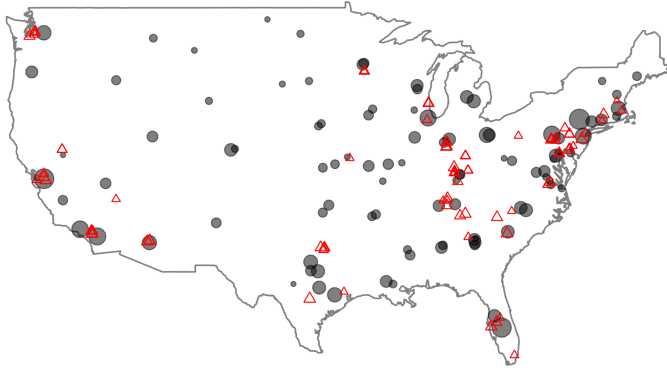
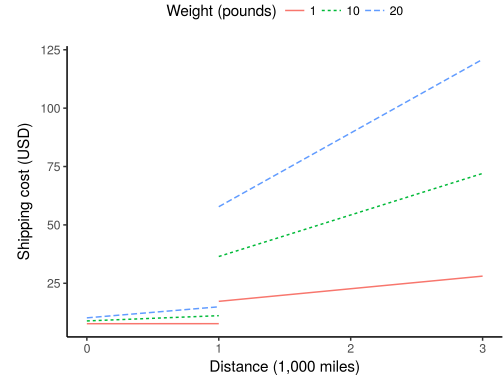
Shipping costs. The shipping cost parameters are given by

$$c_{nr}^i = \begin{cases} 0.247847l_{nr}w^i - 0.227878l_{nr} + 0.132063w^i + 7.553833, & l_{nr} \leq 1 \text{ (i.e., 1,000 miles)}, \\ 1.37674l_{nr}w^i + 4.02487l_{nr} + 0.75604w^i + 11.10890, & \text{otherwise,} \end{cases} \quad (19)$$

where l_{nr} is the spherical distance between fulfillment center n and region r (in thousands of miles, computed by the haversine formula), and w^i is the “shipping weight” of item i (which can depend on both the actual weight of the item and the dimensions of its packaging). To capture the nonlinearity of shipping costs as a function of distance, we assume that for items of a given weight, the per-unit shipping cost is a piecewise-linear function of shipping distance, with ground shipment being used for distances up to 1,000 miles, and second-day air shipment being used for greater distances. We then fit the function to the two-day shipping costs of UPS (United Parcel Service of America, Inc. 2016) to obtain the shipping cost function (19). See Figure 2 for examples.

Item attributes. We generate $|\mathcal{I}| = 1,000,000$ items by sampling four attributes independently and uniformly at random:

- $w^i \in [0, 35]$: the shipping weight (in pounds), which determines c_{nr}^i as mentioned above;
- $d^i \in [0, 100]$: the expected total demand in the planning period;
- $\alpha^i \in [0, 1]$: the geographical distribution parameter, described below;

Figure 1 Fulfillment center and customer region locations**Figure 2** Shipping cost function examples

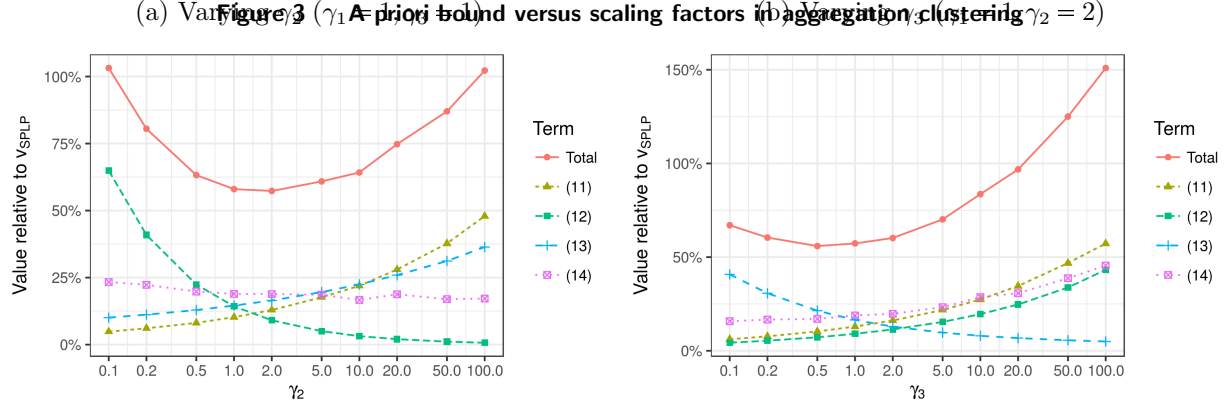
• $f_n^i = f^i \in [0, 10]$: the fixed cost parameter, assumed to be identical for all fulfillment centers for an item (see Section 5.4 for non-identical fulfillment center fixed costs).

The attributes d^i and α^i determine the regional demand d_r^i in the following manner. Let p_r denote the population of region $r \in \mathcal{R}$, and let m_r denote the longitude of region $r \in \mathcal{R}$. Let $\underline{m} = \min_{r \in \mathcal{R}} m_r$ and $\bar{m} = \max_{r \in \mathcal{R}} m_r$ be the longitude of the western-most and eastern-most regions, respectively. For an item $i \in \mathcal{I}$, let the demand in region $r \in \mathcal{R}$ be $d_r^i = d^i \cdot \frac{p_r a_r}{\sum_{r \in \mathcal{R}} p_r a_r}$, where $a_r = \alpha^i \cdot \frac{m_r - \underline{m}}{\bar{m} - \underline{m}} + (1 - \alpha^i) \cdot \left(1 - \frac{m_r - \underline{m}}{\bar{m} - \underline{m}}\right)$. In other words, the regional demand of an item is proportional to its population p_r and to the auxiliary parameter $a_r \in [0, 1]$, which is controlled by the uniform random variable α^i . If α^i is close to 1, then a_r is close to $\frac{m_r - \underline{m}}{\bar{m} - \underline{m}}$ for every $r \in \mathcal{R}$, which means that demand is higher in regions closer to the east coast. Similarly, for an item with α^i close to 0, demand is higher in regions closer to the west coast. When $\alpha^i = 0.5$, we have $a_r = 0.5$ for all $r \in \mathcal{R}$, and the regional demand depends solely on the population. (Note that $\alpha^i = 0.5$ does *not* correspond to the demand distribution being concentrated in the geographic center of the country.) Thus, by sampling α^i at random, we capture demand patterns biased at various levels toward either coast. This serves as a proof-of-concept for more complex demand patterns.

Lower bounds. The numerical example has an SPLP lower bound of $v_{SPLP} = 555,640,052$, which takes 32.62 hours to compute (or an average of 0.117 seconds per item). We report all optimality gaps relative to v_{SPLP} . The simpler and looser bound (18) has a value of 516,177,479 (7.1% lower than v_{SPLP}), and takes 7.78 minutes (or 0.000467 seconds per item) to compute.

5.2. Effects of aggregation on the a priori bound

In this experiment, we aggregate the 1,000,000 items into $|\mathcal{J}| = 1,000$ clusters by performing the well-known k-means clustering algorithm (Lloyd 1982) on $\left(\gamma_1 \alpha^i, \gamma_2 \frac{w^i}{35}, \gamma_3 \exp\left\{-\frac{f^i/10}{d^i/100}\right\}\right)$. This is the scaled vector of three normalized features that correspond to the respective terms of (11), (12) and (13) in the a priori bound of Theorem 1. The scaling factors $(\gamma_1, \gamma_2, \gamma_3)$ represent the



relative emphasis on each term, and we will choose these to achieve the best overall bound. Recall that emphasizing α^i produces clusters of items with similar normalized demand distributions, thus reducing (11). Emphasizing w^i produces clusters of items with similar shipping cost parameters, thus reducing (12). Emphasizing f^i/d^i (as a negative exponent, to control the range) reduces (13); it also indirectly affects the term (14), although ideally (14) should be minimized by not including fixed costs in aggregation, so that clusters have similar fixed costs.

Figure 3 depicts the ratio between the bound terms and the lower bound v_{SPLP} as we vary γ_2 or γ_3 . Due to the inherent randomness of k-means, we report the average of 10 trials. As expected,

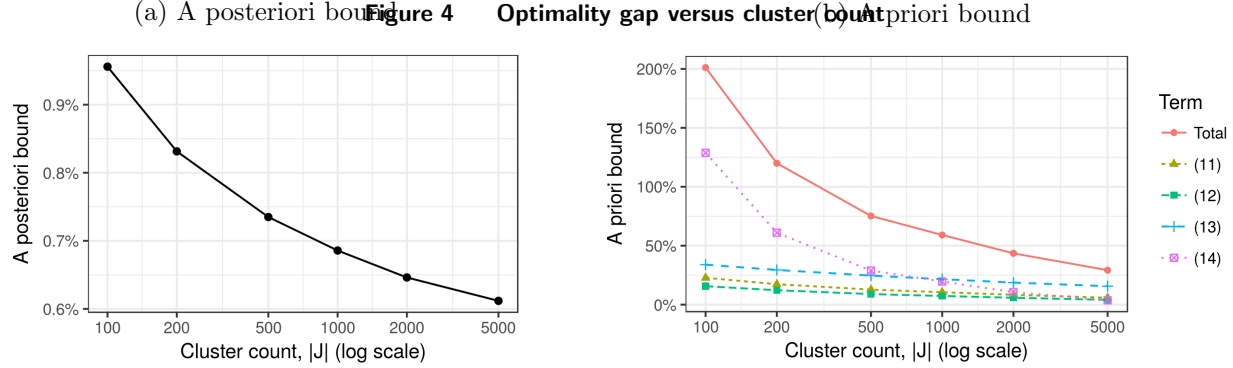
- Increasing γ_2 (while holding $\gamma_1 = \gamma_3 = 1$ constant) leads to a decrease in (12) and an increase in (11) and (13), while (14) remains relatively unaffected. The total is minimized at $\gamma_2 = 2$.
- Increasing γ_3 (while holding $\gamma_1 = 1$ and $\gamma_2 = 2$ constant) leads to a decrease in (13) and an increase in (11), (12) and (14). The total is minimized at $\gamma_3 = 0.5$.

Among the cases tested, $(\gamma_1, \gamma_2, \gamma_3) = (1, 2, 0.5)$ achieves the best a priori guarantee that the final cost exceeds v_{SPLP} by no more than 55.91%. We use these values in subsequent experiments. As we shall see next, the a posteriori bound is in fact much smaller and very close to optimal.

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

This experiment investigates how the number of clusters $|\mathcal{J}|$, also known as cluster count or the level of clustering, controls the trade-off in the following two performance measures:

- *Optimality gap*: Defined as the a posteriori bound of $\left(\sum_{j \in \mathcal{J}} v(\mathbf{D}_j)|_{(\mathbf{x}_j^*, \mathbf{y}_j^*)} - v_{SPLP}\right) / v_{SPLP} \times 100\%$, i.e., the percentage by which the cost of the final placement plan, $\sum_{j \in \mathcal{J}} v(\mathbf{D}_j)|_{(\mathbf{x}_j^*, \mathbf{y}_j^*)}$, exceeds the SPLP lower bound. It is *a posteriori* in the sense that it can only be obtained after completing the full optimization framework, in contrast with the a priori bound that can be calculated solely from input data. A high cluster count should correspond to a smaller optimality gap because it allows the column-based reformulation (\mathbf{P}_3) to be a better approximation of (\mathbf{P}_1) .



- *Computation time* serves as a proxy for computational complexity. Theoretical worst-case complexity analysis is difficult and not very meaningful for our framework, because the algorithms we rely on— k-means clustering, linear programming and mixed-integer programming— are often much faster in practice than their known theoretical guarantees. Besides, the implementations we use are highly optimized by software makers (Pedregosa et al. 2011, Gurobi Optimization, Inc. 2017), as is often the case for online retail companies in practice.

In all instances, column generation is performed until an optimal solution for (\mathbf{P}_3) is found, but we terminate (\mathbf{D}_j) as soon as the solver finds a better solution than the initial feasible solution.

Our findings are illustrated in Figures 4 and 5 and summarized as follows:

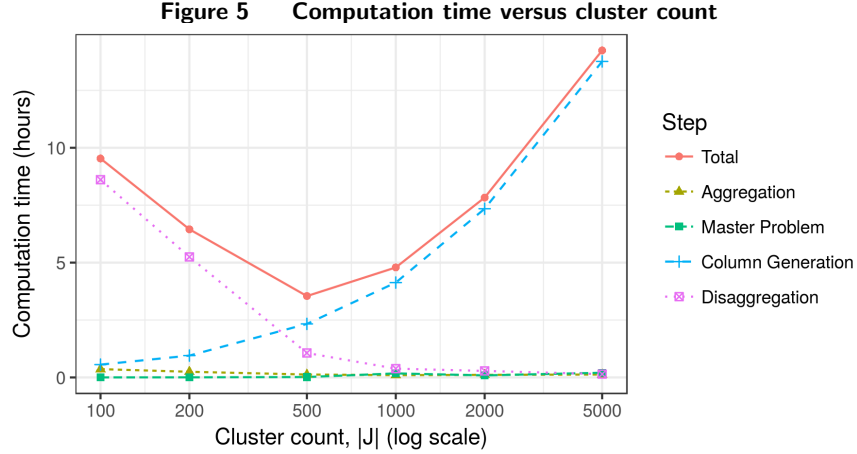
- The a posteriori optimality gap decreases as the cluster count increases (Figure 4a). This is because having smaller clusters allows (\mathbf{P}_2) and (\mathbf{P}_3) to be better approximations of (\mathbf{P}_1) . The bound is less than 1% in all cases, which means that the cost of our solution is very close to that of the optimal solution of the original problem (\mathbf{P}_1) .

- Every term of the a priori bound decrease with increasing cluster count (Figure 4b). This is because, in accordance with the discussion in Theorem 1, having more clusters allows for greater similarity of items within the same cluster. The most significant change is in the term (14), which is proportional to the maximum cluster fixed cost, and hence, decreases as the number of items per cluster decreases.

- The computation time is large when the cluster count is very small or very large (Figure 5). The best case of 3.54 hours is achieved with $|\mathcal{J}| = 500$. When $|\mathcal{J}|$ is small, computation time is dominated by disaggregation, because there is a lot of items in (\mathbf{D}_j) . When $|\mathcal{J}|$ is large, computation time is dominated by column generation, because there are many instances of (\mathbf{CG}_j) . Nevertheless, since both (\mathbf{CG}_j) and (\mathbf{D}_j) can be executed independently for each cluster $j \in \mathcal{J}$, the computation time could be reduced with parallel processors; see Chen (2017) for estimated time savings.

5.4. Effects of fixed costs on placement sparsity

We conduct two numerical studies on how fixed cost parameters control the *sparsity* of placement plans, including *fulfillment center count* per item and *item count* per fulfillment center.



First, for fulfillment center count per item, we consider the solution for $|\mathcal{J}| = 1,000$ from the previous experiment in Section 5.3, for which $f_n^i = f^i$ for every item $i \in \mathcal{I}$. We segment the items by f^i and analyze the fulfillment center count distribution. The average fulfillment center count per item is 6.84. Items with $f^i \in [0, 1)$ use a maximum of 31 and an average of 13.79 fulfillment centers. In contrast, items with $f^i \in [9, 10)$ use an average of 4.09 fulfillment centers, with most items using only 2 or 3 fulfillment centers. See Figure 6 of Online Appendix D for details.

Next, for item count per fulfillment center, we use the solution above as the base case, but remove the assumption that $f_n^i = f^i$ for the fulfillment center that has the highest item count, which is $n = 21$. (This fulfillment center is named BFI4 and is located in Kent, Washington, zip code 98030.) We multiply f_{21}^i by a constant factor for every item $i \in \mathcal{I}$. As expected, a larger constant factor leads to a smaller item count for BFI4, as shown in Figure 7 of Online Appendix D.

The key takeaway from these two studies is that sparsity appears to be a convex function of fixed cost parameters. In other words, doubling f_n^i would reduce the fulfillment center count or item count by more than half. This could inform the tuning of fixed cost parameters in practice.

5.5. Comparison with a sequential placement heuristic

We compare our optimization framework with a *sequential placement heuristic* that places one item at a time, according to some user-specified order. After an item is placed, the remaining fulfillment center capacities are given as input to the placement problem of the next item.

The main advantage of the sequential placement heuristic is its simplicity. By placing one item at a time, it avoids the complexity of jointly optimizing a large number of items. The placement problem for each item is easy to compute; for the size of our network, the solution for one item can typically be obtained in a fraction of a second. Informal interactions with industry professionals indicate that the sequential placement heuristic is a reasonable proxy for current practice.

The main drawback of this heuristic is that its myopic nature can result in an arbitrarily suboptimal cost. We give an example of this in Online Appendix E. Another disadvantage is that, unlike

Table 1 Sequential placement heuristic performance

Sequential placement order	d^i	w^i	$d^i \times w^i$
Optimality gap (%)	9.96	2.45	2.89
Computation time (hours)	71.22	66.45	74.00

our optimization framework, the heuristic as stated cannot be accelerated with parallel processors. (The computation time might be improved to some extent by placing several items simultaneously before updating the capacity.)

Table 1 reports the performance of three sample implementations of the heuristic that place items in decreasing order of (1) demand d^i , (2) weight w^i , (3) the product of demand and weight. The results indicate that sequential placement by weight leads to the smallest optimality gap, and is slightly better than sequential placement by the product of weight and demand. However, the optimality gap and computation time are both much worse than that of our framework. For a detailed comparison of solutions produced by the two methods, see Online Appendix E.

6. Conclusions

In this paper, we formulate the online-retail inventory placement problem as a mixed-integer program and solve it efficiently with a large-scale optimization framework that integrates item aggregation and column generation. We develop a theoretical optimality gap bound that can be calculated a priori from problem data, and use this bound to guide the aggregation process. For a numerical example of 1,000,000 items, the framework is able to find near-optimal solutions in a few hours. The computation time can be further reduced with parallel processors. The framework outperforms a sequential placement heuristic that represents the status quo.

An important extension of our framework is to cases where limits on fulfillment center count or item count are directly imposed as sparsity constraints, instead of being controlled indirectly by fixed cost parameters. Online Appendix F explains how to adapt our framework for sparsity constraints. The resulting optimization models are more difficult to solve, and there are no known a priori bounds. Therefore, fixed costs appear to be an easier approach for ensuring sparsity.

For future work, one promising direction is to incorporate the idea of *iterative aggregation*. Instead of only one level of aggregation (from items to clusters), allow multiple levels, and iteratively refine the placement plan at each level. This technique has been applied successfully to other aggregation-based optimization frameworks, e.g., Bärman et al. (2015) and Shetty and Taylor (1987), and has the potential of further enhancing scalability without severely impacting the optimality gap.

Another direction of future work is to consider multiple periods, such as the case in which demand changes from season to season. A good multi-period placement plan should avoid the overhead of unnecessarily “hopping” from one fulfillment center to another in successive periods, even if those

fulfillment centers are optimal for the individual periods. Our framework could serve as a building block for this more complex setting.

Acknowledgments

The authors would like to thank the Editor, Associate Editor, and two referees for helpful comments that encouraged the development of tighter bounds in Section 4 and more comprehensive numerical experiments in Section 5.

References

- Acimovic J, Graves SC (2015) Making better fulfillment decisions on the fly in an online retail environment. *Manufacturing & Service Operations Management* 17(1):34–51.
- Acimovic J, Graves SC (2017) Mitigating spillover in online retailing via replenishment. *Manufacturing & Service Operations Management* 19(3):419–436.
- Acimovic JA (2012) *Lowering outbound shipping costs in an online retail environment by making better fulfillment and replenishment decisions*. Ph.D. thesis, Massachusetts Institute of Technology.
- Agatz NA, Fleischmann M, Van Nunen JA (2008) E-fulfillment and multi-channel distribution—a review. *European journal of operational research* 187(2):339–356.
- Bärman 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.
- Chen AI (2017) *Large-scale optimization in online-retail inventory management*. Ph.D. thesis, Massachusetts Institute of Technology.
- Costa AM, Cordeau JF, Gendron B, Laporte G (2012) Accelerating benders decomposition with heuristic master problem solutions. *Pesquisa Operacional* 32(1):03–20.
- Francis VE (1985) *Aggregation of Network Flow Problems*. Ph.D. thesis, University of California, Los Angeles.
- Gendron B (2011) Decomposition methods for network design. *Procedia-Social and Behavioral Sciences* 20:31–37.
- Govindarajan A, Sinha A, Uichanco J (2017) Inventory optimization for fulfillment integration in omnichannel retailing. *Available at SSRN: <https://ssrn.com/abstract=2924850>* .
- Gurobi Optimization, Inc (2017) Gurobi optimizer reference manual version 7.0.
- Hallefjord Å, Storøy S (1990) Aggregation and disaggregation in integer programming problems. *Operations Research* 38(4):619–623.
- Hax AC, Meal HC (1975) Hierarchical integration of production planning and scheduling. Geisler MA, ed., *North-Holland/TIMS Studies in Management Sciences, Vol. 1: Logistics*, 53–69 (North-Holland, Amsterdam and American Elsevier, New York).
- Hewitt M, Nemhauser GL, Savelsbergh MW (2010) Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing* 22(2):314–325.

- Hübner A, Holzapfel A, Kuhn H (2016) Distribution systems in omni-channel retailing. *Business Research* 9(2):255–296.
- Klose A, Drexl A (2005) Lower bounds for the capacitated facility location problem based on column generation. *Management Science* 51(11):1689–1705.
- Krarup J, Pruzan PM (1983) The simple plant location problem: survey and synthesis. *European journal of operational research* 12(1):36–81.
- Lei Y, Jasin S, Sinha A (2016) Dynamic joint pricing and order fulfillment for e-commerce retailers. Available at SSRN: <https://ssrn.com/abstract=2753810> .
- Litvinchev I, Tsurkov V (2013) *Aggregation in large-scale optimization* (Springer Science & Business Media).
- Lloyd S (1982) Least squares quantization in pcm. *IEEE transactions on information theory* 28(2):129–137.
- MWPVL International Inc (2015) The amazon fulfillment center and distribution center network in the united states. URL http://www.mwpvl.com/html/amazon_com.html, accessed December 2015.
- Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830, see <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html> for k-means.
- Pirkul H, Jayaraman V (1998) A multi-commodity, multi-plant, capacitated facility location problem: formulation and efficient heuristic solution. *Computers & Operations Research* 25(10):869–878.
- Rogers DF, Plante RD, Wong RT, Evans JR (1991) Aggregation and disaggregation techniques and methodology in optimization. *Operations Research* 39(4):553–582.
- Shetty C, Taylor RW (1987) Solving large-scale linear programs by aggregation. *Computers & Operations Research* 14(5):385–393.
- Sridharan R (1995) The capacitated plant location problem. *European Journal of Operational Research* 87(2):203–213.
- Swaminathan JM, Tayur SR (2003) Models for supply chains in e-business. *Management Science* 49(10):1387–1406.
- Torabi S, Hassini E, Jaihoonian 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.
- United Parcel Service of America, Inc (2016) Daily rate and service guide. URL https://www.ups.com/content/us/en/shipping/cost/zones/daily_rates.html, accessed October 2016.
- United States Census Bureau (2010) 2010 census gazetteer files: Counties. URL <http://www.census.gov/geo/maps-data/data/gazetteer2010.html>, accessed December 2015.

- Xu P (2005) *Order fulfillment in online retailing: what goes where*. Ph.D. thesis, Massachusetts Institute of Technology.
- Zipkin PH (1980a) Bounds for row-aggregation in linear programming. *Operations Research* 28(4):903–916.
- Zipkin PH (1980b) Bounds on the effect of aggregating variables in linear programs. *Operations Research* 28(2):403–418.

Online Appendix

Item Aggregation and Column Generation for Online-Retail Inventory Placement

Appendix A: Feasible region equivalence of the column-based reformulation

THEOREM 2. *Let $(\mathbf{x}^j, \mathbf{y}^j)_{j \in \mathcal{J}}$ be a feasible solution of (\mathbf{P}_2) . Then there exists a feasible solution of (\mathbf{P}_3) , denoted $(\lambda_k^j)_{k \in \mathcal{K}_j, j \in \mathcal{J}}$, such that (6) and (7) hold.*

Proof. First, we establish that for every feasible solution of (\mathbf{P}_2) , the continuous variables can always be expressed as a convex combination of integral points in the set

$$\mathcal{X} = \left\{ (\mathbf{x}^j)_{j \in \mathcal{J}} \mid \sum_{n \in \mathcal{N}} x_{nr}^j = 1, x_{nr}^j \in \{0, 1\} \ \forall n, r, j \right\}.$$

We show this by induction on the number of continuous variables with fractional values, which we denote $\mathcal{F}(\mathbf{x})$; i.e.,

$$\mathcal{F}(\mathbf{x}) = \text{cardinality of the set } \{(n, r, j) \in \mathcal{N} \times \mathcal{R} \times \mathcal{J} \mid 0 < x_{nr}^j < 1\}.$$

We have $\mathcal{F}(\mathbf{x}) = 0$ for every $\mathbf{x} \in \mathcal{X}$. Now suppose $\mathbf{x} = (\mathbf{x}^j)_{j \in \mathcal{J}}$ is the vector of continuous variables in a given feasible solution of (\mathbf{P}_2) for which $\mathcal{F}(\mathbf{x}) > 0$. Let $(n', r', j') \in \mathcal{N} \times \mathcal{R} \times \mathcal{J}$ be such that $0 < x_{n'r'}^{j'} < 1$. Then, because $\sum_{n \in \mathcal{N}} x_{nr'}^{j'} = 1$, there exists another n'' such that $0 < x_{n''r'}^{j'} < 1$. Let $v = x_{n'r'}^{j'} + x_{n''r'}^{j'}$ and let $\mathbf{x}_1, \mathbf{x}_2$ be continuous variable values such that

$$x_{1,nr}^j = x_{2,nr}^j = x_{nr}^j \ \forall (n, r, j) \neq (n', r', j') \text{ and } (n, r, j) \neq (n'', r', j'),$$

$$x_{1,n'r'}^{j'} = v, \quad x_{1,n''r'}^{j'} = 0,$$

$$x_{2,n'r'}^{j'} = 0, \quad x_{2,n''r'}^{j'} = v.$$

It is straightforward to verify that

$$\mathbf{x} = \frac{x_{n''r'}^{j'}}{v} \mathbf{x}_1 + \frac{x_{n'r'}^{j'}}{v} \mathbf{x}_2,$$

and that \mathbf{x}_1 and \mathbf{x}_2 still constitute feasible solutions of (\mathbf{P}_2) . Moreover, we have $\mathcal{F}(\mathbf{x}_1) = \mathcal{F}(\mathbf{x}_2) = \mathcal{F}(\mathbf{x}) - 1$. In other words, \mathbf{x} can be expressed as a convex combination of continuous variables of which there are fewer fractional values. By induction, repeated application of this procedure results in an expression of \mathbf{x} that is a convex combination of integral points in \mathcal{X} .

Next, we construct a feasible solution of (\mathbf{P}_3) that satisfies (6) and (7). The convex combination obtained by the inductive procedure above already implies the existence of a feasible solution $(\lambda_k^j)_{k \in \mathcal{K}_j, j \in \mathcal{J}}$ that satisfies (6). It remains to show that (7) can be satisfied as well. This is achieved by taking $y_n^j(k) = y_n^j$ for every $j \in \mathcal{J}$ and every $k \in \mathcal{K}_j$ such that $\lambda_k^j > 0$. Then we have $(\mathbf{x}^j(k), \mathbf{y}^j(k)) \in X_j$ because if $x_{nr}^j(k) = 1$ for some n and r , then (6) implies $x_{nr}^j > 0$, which in turn implies $y_n^j = y_n^j(k) = 1$ by the feasibility requirement of (\mathbf{P}_2) .

Finally, we complete the proof with the remark that the capacity constraint of (\mathbf{P}_3) is satisfied due to the capacity constraint of (\mathbf{P}_2) and the definition of u_{kn}^j . \square

THEOREM 3. *Let $(\lambda_k^j)_{k \in \mathcal{K}_j, j \in \mathcal{J}}$ be a feasible solution of (\mathbf{P}_3) . Then the corresponding placement plan given by (6) and (7) is a feasible solution of (\mathbf{P}_2) .*

Proof. We proceed by substituting (6) and (7) into each set of constraints in (\mathbf{P}_2) and verifying that the constraints hold.

The demand satisfaction constraint is met for every $r \in \mathcal{R}, j \in \mathcal{J}$ because

$$\sum_{n \in \mathcal{N}} x_{nr}^j = \sum_{n \in \mathcal{N}} \sum_{k \in \mathcal{K}_j} x_{nr}^j(k) \lambda_k^j = \sum_{k \in \mathcal{K}_j} \lambda_k^j \left(\sum_{n \in \mathcal{N}} x_{nr}^j(k) \right) = \sum_{k \in \mathcal{K}_j} \lambda_k^j = 1,$$

where the second-to-last equality is by the definition of X_j , and the last equality is by the second constraint of (\mathbf{P}_3) .

The capacity constraint for every $n \in \mathcal{N}$ is met because

$$\sum_{j \in \mathcal{J}} \sum_{r \in \mathcal{R}} d_r^j x_{nr}^j = \sum_{j \in \mathcal{J}} \sum_{r \in \mathcal{R}} d_r^j \sum_{k \in \mathcal{K}_j} x_{nr}^j(k) \lambda_k^j = \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}_j} \lambda_k^j \left(\sum_{r \in \mathcal{R}} d_r^j x_{nr}^j(k) \right) = \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}_j} u_{kn}^j \lambda_k^j \leq b_n,$$

where we invoke the definition of u_{kn}^j and the capacity constraint of (\mathbf{P}_3) in the latter two equality and inequality, respectively.

For the relationship between the continuous and binary variables of every $n \in \mathcal{N}$ and $j \in \mathcal{J}$, we have

$$\sum_{r \in \mathcal{R}} d_r^j x_{nr}^j = \sum_{k \in \mathcal{K}_j} \lambda_k^j \sum_{r \in \mathcal{R}} d_r^j x_{nr}^j(k) \leq \sum_{k \in \mathcal{K}_j} \lambda_k^j d^j y_n^j(k) \leq d^j \max\{y_n^j(k) \mid k \in \mathcal{K}_j, \lambda_k^j > 0\} = d^j y_n^j,$$

where the first inequality is due to the definition of X_j , and the second inequality is due to the constraints of (\mathbf{P}_3) on $(\lambda_k^j)_{k \in \mathcal{K}_j}$.

Finally, the constraints that $x_{nr}^j \geq 0$ and $y_n^j \in \{0, 1\}$ are trivially true. \square

Appendix B: Column generation algorithm

Algorithm B.1 Stage C: Solving the column-based reformulation with column generation

```

1: Initialize  $\mathcal{K}_j^0$  for every  $j \in \mathcal{J}$  according to (10)
2:  $t \leftarrow 0$ 
3: while termination criteria not met do
4:   Solve the restricted master problem (RMP)
5:    $(p_n)_{n \in \mathcal{N}} \leftarrow$  dual variables of (8)
6:    $(q_j)_{j \in \mathcal{J}} \leftarrow$  dual variables of (9)
7:    $added \leftarrow \text{False}$ 
8:   for all  $j \in \mathcal{J}$  do
9:     Solve the column generation subproblem (CGj)
10:     $v \leftarrow$  objective value of (CGj)
11:     $k^* \leftarrow$  optimal solution of (CGj) as column index
12:    if  $v < 0$  then ▷ Found column with negative reduced cost
13:       $\mathcal{K}_j^{t+1} = \mathcal{K}_j^t \cup \{k^*\}$ 
14:       $added \leftarrow \text{True}$ 
15:    end if
16:  end for
17:  if  $added = \text{False}$  then ▷ All columns have non-negative reduced costs
18:    Go to Line 22
19:  end if
20:   $t \leftarrow t + 1$ 
21: end while
22: return Optimal column solution of (P3)

```

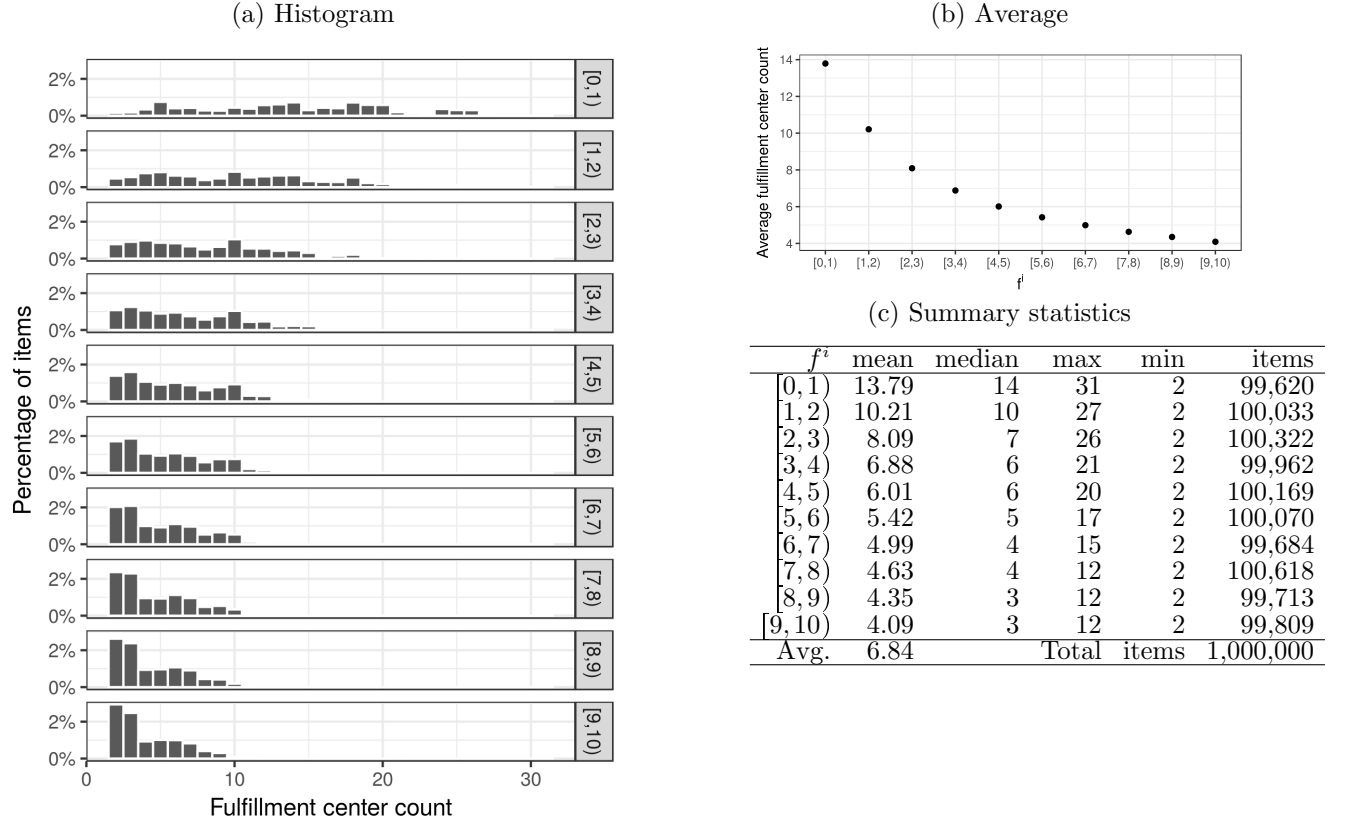
Appendix C: Additional proof for theoretical bound

LEMMA 4. For a given item $i \in \mathcal{I}$, there exists an optimal solution of (**MCF_i**) such that for every node $r \in \mathcal{R}$, if $\Delta_{rr'} > 0$ or $\Delta_{r'r} > 0$ for any $r' \neq r$ (i.e., there is flow passing through the node r), then exactly one of the following conditions is true about r :

- $\delta_r^i - d_r^i > 0$ and $\Delta_{rr'} = 0$ for all $r' \neq r$ (i.e., there is no outgoing flow, but only incoming flow);
- $\delta_r^i - d_r^i < 0$ and $\Delta_{r'r} = 0$ for all $r' \neq r$ (i.e., there is no incoming flow, but only outgoing flow).

Proof. It suffices to show that if there is a solution with nodes that do not satisfy either condition, then there exists another solution that is no worse in cost for which there is one fewer of such nodes. Let $r, r', r'' \in \mathcal{R}$ be distinct nodes such that $\Delta_{r'r} = a > 0$ and $\Delta_{rr''} = b > 0$. The node r does not satisfy either condition, as it has both incoming and outgoing flows. Consider another solution such that $\Delta_{r'r} = a - \Delta$ and $\Delta_{rr''} = b - \Delta$, where $\Delta = \min(a, b)$, and let there be an additional flow of Δ along the arc (r', r'') , so that all flow is balanced. The change in cost from the former solution to the latter solution is $\Delta \cdot (\bar{c}_{r'r''} - \bar{c}_{r'r} - \bar{c}_{rr''})$. By definition, we have

$$\begin{aligned}
\bar{c}_{r'r''} - \bar{c}_{r'r} - \bar{c}_{rr''} &= \max_{n \in \mathcal{N}} (c_{nr''}^i - c_{nr'}^i) - \max_{n \in \mathcal{N}} (c_{nr}^i - c_{nr'}^i) - \max_{n \in \mathcal{N}} (c_{nr''}^i - c_{nr}^i) \\
&\leq \max_{n \in \mathcal{N}} (c_{nr''}^i - c_{nr'}^i) - \max_{n \in \mathcal{N}} [(c_{nr}^i - c_{nr'}^i) + (c_{nr''}^i - c_{nr}^i)] \\
&= \max_{n \in \mathcal{N}} (c_{nr''}^i - c_{nr'}^i) - \max_{n \in \mathcal{N}} (c_{nr''}^i - c_{nr'}^i) = 0,
\end{aligned}$$

Figure 6 Distribution of fulfillment center count

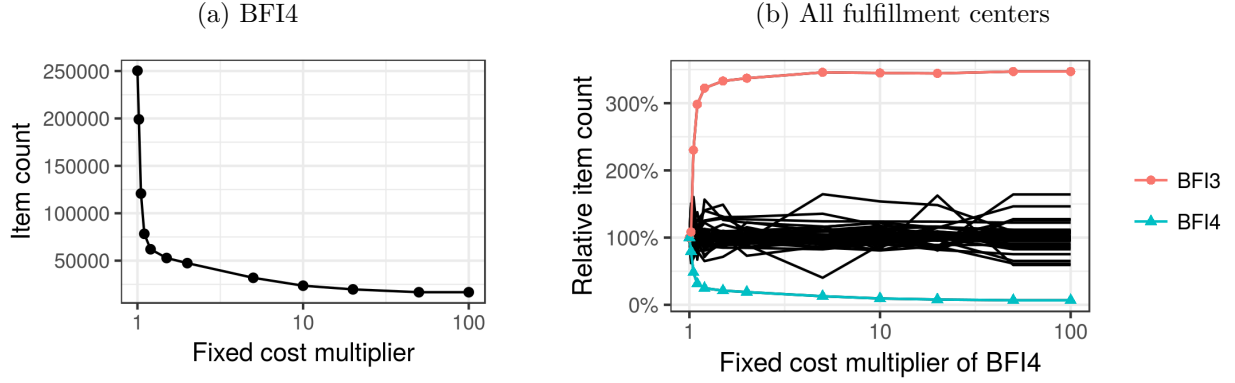
where the inequality is due to the fact that $\max_{n \in \mathcal{N}} v_n + \max_{n \in \mathcal{N}} w_n \geq \max_{n \in \mathcal{N}} (v_n + w_n)$ for any set $\{(v_n, w_n) \mid n \in \mathcal{N}\}$. Thus, the latter solution is no worse in cost, and r now satisfies exactly one of the two conditions. Moreover, the conditions of r' and r'' are not affected, since (r', r) and (r', r'') are both outgoing arcs from r' , and likewise, (r, r'') and (r', r'') are both incoming arcs to r'' . By repeating this procedure, the node r can be made into a pure source or sink, without affecting the status of any other nodes. \square

Appendix D: Additional figures for fixed cost experiment

See Figures 6 and 7. Note: One might wonder how changing the fixed cost parameters of BFI4 affects other fulfillment centers. It turns out that as the item count of BFI4 decreases, the item count of BFI3 (located in DuPont, Washington, zip code 98327) increases, while other fulfillment centers appear relatively unaffected. This is illustrated in Figure 7b, which shows the item count for all fulfillment centers, relative to each of its own item count in the base case. A reasonable explanation is that, since BFI4 and BFI3 are located close to each other and happen to have the same square footage, the latter absorbs most of the impact of BFI4's item count being decreased.

Appendix E: Suboptimality of the sequential placement heuristic

The sequential placement heuristic can result in an arbitrarily suboptimal cost. To see this, consider a small example with $N = 2$ fulfillment centers, $R = 1$ region, and two items $\mathcal{I} = \{1, 2\}$. Suppose both fulfillment centers have capacity $1 + \epsilon$, where $\epsilon > 0$, and that the fixed costs are all zero. Suppose also that the demand of

Figure 7 Item count

items are $d_1 = 1 + \epsilon, d_2 = 1$, and that the shipping costs are $(c_{11}^1, c_{21}^1) = (1, 2)$ for Item 1, and $(c_{11}^2, c_{21}^2) = (1, C)$ for Item 2, where $C > 2$ is a large number. In other words, it is better to ship both items from the first fulfillment center, and in the case that they need to be shipped from the second fulfillment center, the additional cost for Item 2 is greater than that of Item 1.

Now consider a sequential placement heuristic that places items in the order of decreasing demand. It starts by placing Item 1 entirely in the first fulfillment center; then, it places Item 2 entirely in the second fulfillment center. This results in a total shipping cost of $d_1 c_{11}^1 + d_2 c_{21}^2 = 1 + \epsilon + C$.

A better solution would be to place Item 2 fully in fulfillment center 1, along with ϵ units of Item 1, and place the remaining 1 unit of Item 1 in fulfillment center 2, for a total cost of $3 + \epsilon$. Depending on the value of C , the former cost could be arbitrarily higher than the latter.

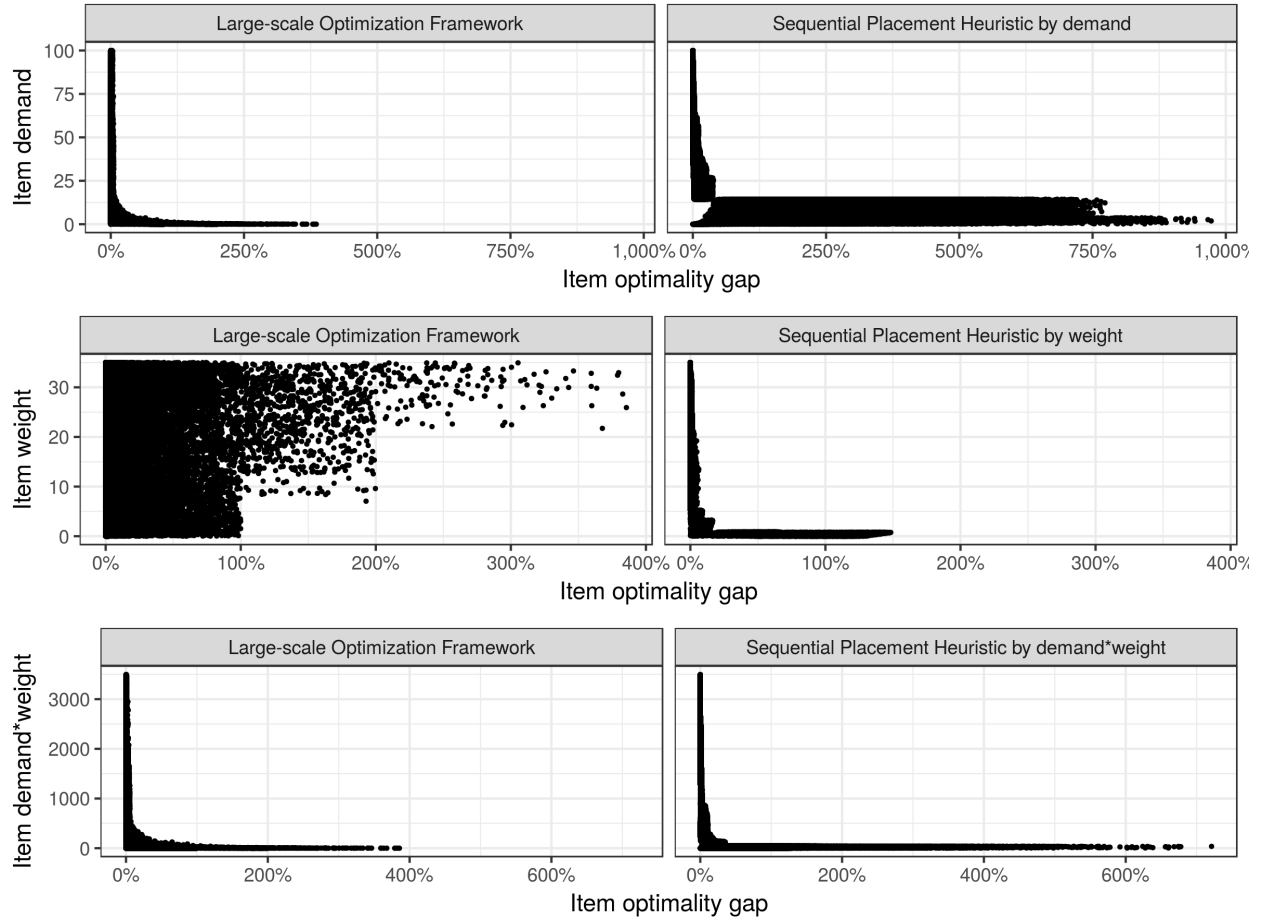
Figure 8 provides numerical evidence on the suboptimality of sequential placement. There are six subfigures, organized into three rows and two columns. The column on the left illustrates the solution of the large-scale solution framework, while the column on the right illustrates solutions of the three sequential placement heuristics that, respectively, places items in the order of decreasing demand, weight, and the product of demand and weight. In each subfigure, there are 1,000,000 dots, each representing an item. The x -axis is the “item-wise” optimality gap, defined as $\frac{v^i - v_{SPLP}^i}{v_{SPLP}^i} \times 100\%$, where v^i is the cost of item i ’s placement plan, and v_{SPLP}^i is the item’s SPLP lower bound. The y -axis is the item’s attribute. (Note that the left column is the same placement plan plotted against three different item attributes, while the right column are three different placement plans, produced by three different sequential placement orders.)

The figure reveals that, for the sequential placement heuristic, the item-wise optimality gap is small for the items that are placed first (i.e., high on the y -axis), but continues to increase as placement progresses. Moreover, there is a significant surge in optimality gap for items that are placed towards the end (i.e., low-demand or low-weight items). This phenomenon is consistent with the small example given above.

Appendix F: Extensions to sparsity-constrained placement

Instead of using fixed costs to induce sparsity in the placement plan, we can remove the fixed cost component in the objective of (P_1) and add the following *sparsity constraints* on the fulfillment center usage variables $\mathbf{y}^i = (y_n^i)_{n=1}^N$:

$$(\mathbf{y}^i)_{i \in \mathcal{I}} \in S, \quad (20)$$

Figure 8 Item-wise optimality gap for different solution methods

where the set S represents different types of sparsity constraints, as described below.

Using sparsity constraints instead of fixed cost parameters has the potential benefit that, in certain business contexts, the set S may be easier to characterize than the fixed cost parameters. However, it also has the drawback that the presence of both sparsity and capacity constraints may result in the problem being infeasible. There appears to be no simple way to check feasibility. One technique for ensuring feasibility is to add a virtual fulfillment center that is uncapacitated and does not count toward sparsity constraints, but has a high shipping cost to all regions. Then, any flow that would not fit in the actual fulfillment centers would be redirected to the virtual one, and the resulting solution could be used to diagnose the cause of infeasibility.

Below, we describe two types of sparsity constraints that give rise to different problem structures, and discuss how to adapt the large-scale solution scheme to each case.

Fulfillment center count per item. An online retailer may wish to limit the number of fulfillment centers carrying an item, due to considerations such as replenishment location restrictions imposed by suppliers. This can be modeled by specifying (20) as

$$\sum_{n=1}^N y_n^i \leq s_i \quad \forall i \in \mathcal{I},$$

where the sparsity limit s_i for each item $i \in \mathcal{I}$ is given as input.

When performing aggregation, the sparsity limit s_j of the cluster $j \in \mathcal{J}$ should be taken as the minimum sparsity limit among its items, $\min_{i \in \mathcal{I}_j} s_i$, so as to ensure the feasibility of the disaggregated placement plans for each item. For this reason, it may be a good idea to aggregate the items according to their sparsity limits, in addition to other attributes.

The column-based reformulation of this problem has the same reduced master problem (**RMP**), while the constraint $\sum_{n=1}^N y_n^j \leq s_j$ is added to the column-generation subproblem (**CG_j**). The resulting column solution is a convex combination of columns that satisfy the sparsity constraint. The disaggregation problem should be an integer program that selects exactly one column for each item, which is different from the previous formulation that selects a convex combination of columns with fixed cost adjustments.

Item count per fulfillment center. A fulfillment center may also have a limit on the number of items it can carry, for example, if the storage space is organized by “bins” that must each carry a unique item. In this case, we can specify (20) as

$$\sum_{i \in \mathcal{I}} y_n^i \leq t_n, \quad \forall n = 1, \dots, N,$$

where the sparsity limit t_n for each fulfillment center n is given as input.

Unlike the previous case, these sparsity constraints cannot be decomposed by item, but instead create coupling across items. Therefore, they must be included in the reduced master problem along with fulfillment center capacity constraints. One possible approach is to add the following constraints to (**RMP**):

$$\sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}'_j} |\mathcal{I}_j| y_n^j(k) \lambda_k^j \leq t_n, \quad \forall n = 1, \dots, N,$$

where $|\mathcal{I}_j| y_n^j(k)$ is equal to $|\mathcal{I}_j|$, the item count of cluster j , if column $k \in \mathcal{K}_j$ utilizes in fulfillment center n (i.e., if $y_n^j(k) = 1$), and 0 otherwise. Recall that λ_k^j represents the fraction of cluster j that uses column k . Thus, the left-hand side of the constraint is an estimate of the total number of items (across all clusters) placed in fulfillment center n .

When performing disaggregation, we need to ensure that the number of items allocated to each fulfillment center does not exceed the value dictated by the convex combination of columns. This can be achieved by adding the constraints

$$\sum_{i \in \mathcal{I}_j} y_n^i \leq t_n^j \quad \forall n = 1, \dots, N,$$

to the disaggregation problem (**D_j**), where $t_n^j := \left\lfloor \sum_{k \in \mathcal{K}'_j} |\mathcal{I}_j| y_n^j(k) \lambda_k^j \right\rfloor$ is the number of items in cluster j that can be placed in fulfillment center n , and $\lfloor \cdot \rfloor$ denotes rounding down to the nearest integer (although it is possible for some clusters to be rounded up as well, as long as $\sum_{j \in \mathcal{J}} t_n^j \leq t_n$).