Assortment Allocation to Distribution Centers to Minimize Split Customer Orders

Andrés Catalán [1*] and Marshall L. Fisher [2*]

[*] Operations and Informations Management Department, The Wharton School, University of Pennsylvania.

3730 Walnut Street, Suite 500 Jon M. Huntsman Hall, Philadelphia, PA 19104.

[1] Telephone: (215) 253-5533 Fax: (215) 898-3664 Email: andresc@wharton.upenn.edu
[2] Telephone: (215) 898-7721 Fax: (215) 898-3664 Email: fisher@wharton.upenn.edu.

This version: November 4, 2012. This research was supported in part by a grant of the Baker Retailing Center of The Wharton School of the University of Pennsylvania. The authors gratefully acknowledge the support of Gang Yu, Junling Liu, Chrissy Minyard, Jonathan Wang, Jason Han, Arthur Huang and Jessie Lai at Yihaodian, as well as Vishal Gaur, Ernesto Avendaño, Hessam Bavafa, Jun Li and other participants at the 2012 Meeting of the Consortium of Operational Excellence in Retail and the Wharton OPIM Doctoral Seminar Series.

Assortment Allocation to Distribution Centers to Minimize Split Customer Orders

Abstract

A split order in retail occurs whenever an order is fulfilled in more than one shipment. This entails additional costs to the retailer. In the context of electronic commerce retail, we study the problem of assigning stock-keeping units to distribution centers (DCs) to minimize split orders. We show that this problem is NP-hard, propose several heuristics and test them against other benchmark algorithms using actual transaction data from a Chinese online retailer. We evaluate the results of the test along three dimensions: the average number of shipments per order, workload balance across the DCs, and the number of DCs that are capable of fulfilling a particular order.

Keywords

Split orders, online retail, heuristics, assortment

1 Introduction

Consider the problem faced by a retailer that fulfills customer orders directly from its distribution centers (DCs). If none of the DCs contain all of the stock-keeping units (SKUs) in a particular order, the order must be fulfilled in multiple shipments.

This so-called *split order* adds considerably to fulfillment cost, including longer handling and picking times, and increased shipping cost, costs which the retailer cannot typically charge the consumer. Amazon recognizes this problem in the 10K filing for fiscal year 2011:

> "We may experience *an increase in our net shipping cost* due to complimentary upgrades, *split-shipments*, and additional long-zone shipments necessary to ensure timely delivery for the holiday season." (Amazon.com, Inc. 2011, item 1A, p. 8, emphasis added)

Amazon reported $4 billion in outbound shipping costs in 2001, which is 87.2% of total fulfillment costs, including warehousing, and 6.3 times the reported $631 million of net profit (Amazon.com, Inc. 2011). Although there are no public reports of the incidence of split orders among retailers, we can extrapolate data from previous research to quantify the impact of this phenomenon. According to Xu, Allgor, and Graves (2009), in 2004 a large online retailer had to split 10 to 15% of its orders containing multiple SKUs, which made up 3.9 to 6.4% of the total orders for that year. In other words, at least 3.75 to 6% of the shipments were additional shipments due to split orders. If these percentages were applied to Amazon in 2011 and all shipments are assumed to cost the same, split orders would be responsible for 150 to 240 million dollars in costs, or about 24 to 38% of stated profit.

Split orders arise from two main sources. The first source is how SKUs are allocated to DCs: if an order includes two SKUs that are not stored together in the same DC, that order will have to be split. This can happen when none of the DCs is large enough to accommodate all of the SKUs in their product line. The second source is the stock level of each SKU in each DC: when one of the SKUs in an order is not in stock, part of that order will have to be shipped from another DC or backordered until more stock arrives, even if all of them are originally allocated to the same DC.

In this paper we examine the first of these sources, the allocation of SKUs to DCs, and propose a set of heuristics to minimize the number of split orders. Besides the cost of split orders, the way SKUs are assigned to DCs can impact other metrics of interest, such as the balance of workload between DCs or the number of DCs that are capable of fulfilling a particular order. Being able to fulfill orders from multiple DCs would allow the retailer to satisfy other objectives, such as selecting a closer DC for fulfillment, or to hedge against split orders caused by stockouts. In our analysis of these heuristics we also measure their impact along these other dimensions.

Our research was conducted in collaboration with Yihaodian, an online retailer based in Shanghai, China. The company started operations in 2008 in the Shanghai area, initially focusing on fast-moving consumer goods, but rapidly expanding to other key cities and many other product segments (Fisher, 2012). In 2011 it was distinguished as the fastest growing technology company in Asia (Deloitte, 2011). In China, customer preferences vary significantly between regions. Thus, Yihaodian must provide a highly localized assortment to each region, which therefore can only be served from local DCs. In addition to that, the company's assortment has expanded at a remarkably fast pace to remain competitive. At the end of 2011 Yihaodian's product database comprised 6 times more SKUs than at the beginning of that year (see Figure 1). As the assortment broadens, Yihaodian increases the number of DCs as needed, assigning the available capacity to new SKUs are they are brought into the assortment. Due to the breadth of the company's product offering and the dynamic but fragmentary nature of its warehousing capacity, split orders are a frequent, costly problem for Yihaodian.

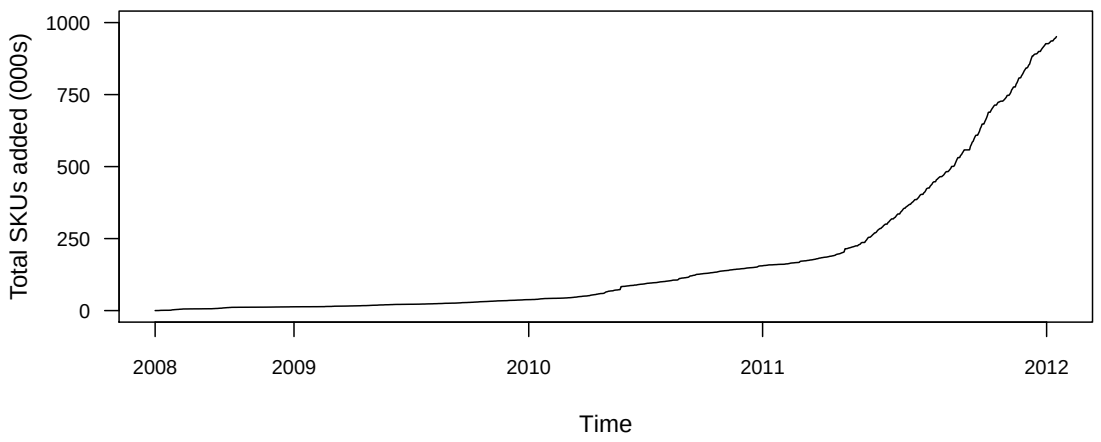

Figure 1: Cumulative number of SKUs added to the assortment over time.

According to the company, Yihaodian's typical profit per transaction usually lies between 0 and 20 RMB and the average shipment from a warehouse costs approximately 12 RMB per order. Thus, Yihaodian loses money on most split-shipped orders. The problem is exacerbated by the retailer's initial positioning as an online supermarket for dry groceries and its use of periodic promotions, which drive up the incidence of multi-SKU, multi-item orders. Yihaodian estimated the prevalence of split orders in Shanghai during the first half of 2012 to be up to 18.1% of all orders with multiple SKUs. However, after some initial efforts to curb split orders that figure went down to 13%. The company found that up to 77.32% of the split orders in a given week of the first semester of 2012 could be attributed to how the SKUs in the assortment had been allocated to DCs.

Although the impact of split orders in the bottom line of online retailers is significant, the operations management literature has barely addressed the subject. Xu et al. (2009),

the only other paper of which we are aware that directly addresses split orders, focuses on how to sequence orders to minimize stockouts that lead to split orders, proposing a dynamic optimization that occurs in real time.

This lack of emphasis on split orders is understandable, since, until recently, most retailers shipping their products directly to customers (including online, catalog and telephone sales) operated in an environment in which the following features were typical:

(a) supply chains were simple, with few DCs and almost no overlap between each DC's coverage areas;

(b) assortments were sufficiently narrow relative to DC capacity that DCs were capable of storing most, if not all, of the products on sale;

(c) the product categories they carried did not encourage broad multi-item orders;

(d) the retailers' product margins were high enough to make the extra shipping cost of a split order less relevant;

(e) delivery time standards were longer, allowing retailers enough time to pool orders.

However, in the early 21st century online retail has become mainstream and the above conditions no longer hold for many retailers. For instance, in 1991, catalog apparel retailer L. L. Bean carried 6,000 items in a single DC in Freeport, Maine (Schleifer, 1993, p. 2). By 2001 that count had increased to 16,000, still shipping from the same DC (Lal, Salmon, & Weber, 2004, p. 9). By contrast, in 2012, Amazon operates 31 fulfillment centers in the US, with up to 9 more to open towards the end of the year (Wulfraat, 2012). As of the writing

of this paper in 2012, a search on Amazon's US website yields over 140 million results. Of those search results, 17.5 million can be shipped under Amazon's Prime program and therefore fulfilled from the company's DCs. The product categories that L.L. Bean sold in the 1990s are, of course, a reduced subset of those sold by Amazon. Even though the relationship between those search results and actual SKUs might not be one-to-one, this still illustrates how direct-to-consumer retail has increased several orders of magnitude in size, reach and complexity, making the problem of split orders more salient.

Similar reasons could explain why most research contributions in online retailing come from literature streams related to marketing and information systems, rather than from operations and supply chain management. Lee (2001) explains that there are three stages or waves that appear successively and overlap as technology innovations evolve: substitution, scale and structural change. In a complex system, technology initially replaces a subset of the processes, leaving the rest of the system unchanged. When it leads to increased scale, sometimes due to a shift in former constraints, this triggers a structural change that can transform the whole system. From a supply chain standpoint, the emergence of online retailers is not different from the opening of another direct sales channel, as the substantive content of online retail operations is essentially the same as, say, those in the supply chain of catalog retailers. Thus, it would be logic that the first wave of innovation has been studied in more depth from those research fields concerned with the management of that sales channel. However, as scale ramps up and structural changes start to become noticeable, we would expect researchers and practitioners in the field of supply chain management to address the operational challenges

posed by online retail.

The rest of the paper is structured as follows. In section 2 we characterize the split order problem as an integer program and make use of an equivalent graph problem to show that it is NP hard. Section 3 describes four heuristics for allocating SKUs to DCs to minimize split orders. We also describe two other algorithms for SKU allocation, intended to represent typical current practice in the industry, that we use as a benchmark. Section 4 describes in detail the methodology and the actual transaction data that we used to test the performance of heuristics and benchmarks. In section 5 we compare the performance of heuristics and benchmarks along three different dimensions: cost, workload balance and flexibility. Finally, section 6 summarizes our findings and suggest additional research.

2 Integer programming formulation

Consider a retailer that has been operating for some time a network of DCs to directly fulfill customer orders. The retailer wants to use historic transaction data to minimize the number of split orders by placing in the same DCs those SKUs that are more likely to be ordered together.

When allocating SKUs to DCs to minimize the number of split orders, the retailer makes decisions before and after observing the actual content of the orders. In the first stage the retailer allocates SKUs to DCs without knowing what the orders will be. In the second stage, once an order is realized, the retailer ships from one or several DCs to fulfill the order in the least number of shipments possible. Both sets of decisions affect the number of shipments that it will take to fulfill all the orders. We could model this problem as a two-stage stochastic binary

optimization problem with recourse (Birge & Louveaux, 1997) in which order realizations are the stochastic component.

This approach would require estimating a probability distribution of orders. Without additional assumptions, the parameter space for such a distribution would be exponential in the number of SKUs, and the only available data on which to base this would be past actual orders.

We adopt an approach that applies heuristics directly to past transaction data (the same dataset that would have to be used to estimate the order distribution) to obtain an allocation of SKUs. By doing this, we assume that the immediate future will resemble the recent past. However, the useful life of the allocation will be limited by, among other factors, changes in the retailer's offering or shifts in consumer preferences.

This methodology is similar to that of predictive modelling (Blattberg, Kim, & Neslin, 2008, chapter 10), which is often used in quantitative marketing to mine information such as credit worthiness or customer persuadability (Linoff & Berry, 2011).

Within this framework, we formulate a deterministic binary program that corresponds to the problem of allocating SKUs to DCs to minimize the number of shipments over a history of orders. We specify the following parameters:

- D: number of DCs, indexed by d
- S: number of SKUs, indexed by s
- O: number of orders, indexed by o
- k_d: maximum number of SKUs that can be allocated to DC d

- M_o: set of SKUs in order o

By expressing capacity in terms of SKUs we are implicitly assuming that all SKUs have similar dimensions and warehousing specifications. It would be easy to generalize to non-homogeneous SKUs by introducing the appropriate coefficients in the constraints. We assume that the total capacity is enough to hold all the SKUs, $\sum_{d=1}^{D} k_d \geq S$, but no DC is big enough to hold all of them, $k_d < S$, $\forall d$.

The decision variables are defined as follows:

- $y_{sd} = 1$ if SKU s is allocated to DC d; 0 otherwise
- $x_{sdo} = 1$ if SKU s ships from DC d to fulfill order o; 0 otherwise
- $z_{do} = 1$ if order o is fulfilled with a shipment from DC d; 0 otherwise

The problem can be expressed as:

$$\min_{\mathbf{X},\mathbf{Y},\mathbf{Z}} \quad \sum_{d=1}^{D}\sum_{o=1}^{O} z_{do}$$

$$\text{subject to} \quad \sum_{d=1}^{D} y_{sd} \geq 1 \qquad \forall s \qquad (1)$$

$$\sum_{s=1}^{S} y_{sd} \leq k_d \qquad \forall d \qquad (2)$$

$$\sum_{d=1}^{D} x_{sdo} = 1 \qquad \forall o, \forall s \in M_o \qquad (3)$$

$$x_{sdo} \leq y_{sd} \qquad \forall d, \forall o, \forall s \in M_o \qquad (4)$$

$$x_{sdo} \leq z_{do} \qquad \forall d, \forall o, \forall s \in M_o \qquad (5)$$

The first two constraint sets impose that all SKUs must be allocated to at least one DC and no DC can contain more SKUs than it is specified in its capacity. The constraints (3) control that a SKU required by a given order is shipped from exactly one of the DCs, while those in (4) and (5) ensure that SKUs will be shipped from DCs where they have been allocated and that there will be a shipment from every DC where there is at least one SKU to ship, respectively.

The objective function assumes that the cost of a split order is the same for any order and for shipments originating in any DC. This was the case for Yihaodian, the retailer we have been collaborating with in our research. However, if needed, it would be straightforward to have order- and DC-specific coefficients in the objective function.

Proposition 1 below shows that the LP relaxation of this integer program provides only the trivial bound that the number of shipments is as great as the number of orders.

Proposition 1. *The LP relaxation of the integer program always yields a minimum number of total shipments equal to the number of orders.*

Proof. It suffices to show a feasible solution to the LP relaxation with objective value O. Take any $\mathbf{w} = (w_1, \ldots, w_D)$ such that $\sum_{d=1}^{D} w_d = 1$ and $w_d \leq \frac{k_d}{S}$. Note that a vector $\mathbf{w}$ satisfying these constraints must exist, or else total DC capacity would be insufficient to hold all SKUs. We show that the solution $x^*_{sdo} = w_d$ if $s \in M_o$, and 0 otherwise, $y^*_{sd} = z^*_{do} = w_d$ satisfies all constraints and has an objective value of O. By definition, for all o and for all $s \in M_o$, $\sum_{d=1}^{D} x_{sdo} = \sum_{d=1}^{D} w_d = 1$, which satisfies constraints (3). Constraint sets (4) and (5) can be rewritten as $y_{sd} = \max_o x_{sdo}$ and $z_{do} = \max_s x_{sdo}$. Given that $y^*_{sd} = w_d$ and $z^*_{do} = w_d$, both constraint sets are satisfied. Then, the number of shipments will be $\sum_{d=1}^{D} \sum_{o=1}^{O} z_{do} =$

$\sum_{o=1}^{O}\left(\sum_{d=1}^{D} w_d\right) = O.$ □

We can try to find a lower bound of the problem from a different approach, by using graph theory. The integer program can be represented as a hypergraph problem. Hypergraphs are a generalization of graphs in which edges, referred to as hyperedges, can connect any positive number of nodes. If every SKU is represented by a node, orders, as nonempty subsets of SKUs, can be represented by hyperedges. Thus, the split order problem becomes a hypergraph problem in which the nodes need to be split into D (overlapping) subsets of specific sizes $(k_1, \ldots, k_D)$ so that each hyperedge should be contained in the union of the least number of those D subsets.

Consider the special case that all the orders have exactly 2 SKUs. Therefore, hyperedges will have exactly 2 nodes and the hypergraph becomes a regular graph. Assume that there are 2 DCs and that both have the same capacity, $k_d = k$. Let every possible pair of SKUs occur in exactly one order, so that the graph is also complete. As a result, all S SKUs are identical, and it is optimal to use up all the available storage space and place any $B = 2k - S$ SKUs in both DCs. There will be $k - B = S - k$ SKUs allocated to one DC and not the other, and vice versa, so $(S-k)^2$ of the total number of orders will be split into two shipments. Therefore, $\binom{S}{2} + (S-k)^2$ shipments will be the optimal value, which is greater than the number of orders, $O = \binom{S}{2}$.

Suppose now that the regular graph is not complete, but, more generally, that its largest clique (or complete subgraph) has S' nodes. If none of the DCs can accommodate those S' SKUs ($S' > k$), the lower bound will be at least $O + (S' - k)^2$.

Another special case of the hypergraph formulation of the split order problem can help us determine the complexity of the split order problem.

Proposition 2. *The split order problem is NP-hard.*

Proof. Consider the hypergraph formulation of the split order problem. In the special case that S is even and all hyperedges have cardinality 2 and unit weights, when the number of DCs is $D = 2$ and their capacity is equal, $k_1 = k_2 = \frac{S}{2}$, the split order problem reduces to the graph bisection problem. A graph bisection is a partition of the node space into two equally sized, mutually exclusive subsets. The objective is to find a graph bisection that minimizes the number of edges straddling both subsets. It is known to be NP-hard (Karp, 1972; Bui & Jones, 1992; Feige, Krauthgamer, & Nissim, 2000; Andreev & Racke, 2006). □

3 Heuristics and benchmarks

We base our heuristics on the coappearance matrix, $\mathbf{C}$, where c_{ij} denotes the number of orders that include SKUs i and j and c_{ii} denotes the number of orders in which SKU i is present. Let $\mathbf{R}$ be the $O \times S$ binary matrix of orders, such that $r_{os} = 1$ if $s \in M_o$, and 0 otherwise. Then $\mathbf{C} = \mathbf{R}^T\mathbf{R}$.

The coappearance matrix is related to the weighted primal graph of the hypergraph that corresponds to the split order problem. A primal graph is a projection of the hypergraph into an ordinary graph: it has the same set of nodes, and its edges link a pair of nodes if they were part of the same hyperedge in the hypergraph. We define weights over the edges of the primal graph equal to the number of hyperedges that contain each pair of nodes. In the

split order problem, the sum of the adjacency matrix of the weighted primal graph plus the diagonalization of the vector of degrees of the hypergraph is equal to the coappearance matrix.

The coappearance matrix overweighs those pairs of SKUs that appear in orders with more than 2 SKUs, but, in exchange, we gain tractability and operate on a simpler but yet sound data structure that can be computed and accessed easily. In section 4 we show the descriptive statistics of the dataset against which we are applying the heuristics and we verify that most orders have 4 SKUs or less, so the loss in the approximation is minimal.

3.1 Heuristics

We developed four heuristics to allocate SKUs to distribution centers. All of them use the coappearance matrix, which can be calculated using past basket content data that can easily be extracted from the retailer's transaction records.

The `GREEDY ORDERS` heuristic (Algorithm 1 in the appendix) processes all the orders, sorted by decreasing number of SKUs per order. The algorithm assigns all the SKUs of every order to the largest DC that has unallocated space available, as long as these SKUs had not been allocated yet to that DC. Once all SKUs are allocated to one of the DCs, the remaining capacity is allocated using the matrix of coappearances: for every remaining empty slot in a DC, the algorithm selects the SKU among those not yet in that DC that has the highest coappearance with the contents of that DC. Once all the capacity of the DCs is allocated, the heuristic stops.

The `GREEDY PAIRS` heuristic (Algorithm 2 in the appendix) follows an approach similar

to GREEDY ORDERS, although, instead of reviewing all the orders, it processes all the pairs of SKUs, sorted by decreasing number of times that each pair has been ordered together. Like in GREEDY ORDERS, after all the SKUs have been allocated to one DC, the same process is used to fill the remaining DC capacity.

The GREEDY SEEDS heuristic (Algorithm 3 in the appendix) is slightly more sophisticated. It starts by choosing the best-selling SKU as the initial SKU, or "seed", for the largest DC. The algorithm iteratively designates seeds for each of the remaining DCs, proceeding in order of decreasing capacity, by choosing the SKU from the 10% best-selling SKUs with the lowest coappearance with respect to all the previously selected seeds. After allocating seeds to all DCs, the second stage of the algorithm consists on processing the rest of the SKUs, sorted by decreasing order of sales, to ensure that all of them are assigned to a DC. The algorithm calculates the average coappearance of each SKU with respect to the contents of each DC and allocates it to the DC with the highest average. At the end, the coappearance matrix is used to allocate the remaining DC capacity, as in the two algorithms described above.

Similarly to what we did in section 2 to find a lower bound of the optimal value of the objective function, the BESTSELLERS heuristic (Algorithm 4 in the appendix) assigns the B best-selling SKUs to all the DCs, where $B = \left\lfloor \frac{\sum_d k_d - S}{D-1} \right\rfloor$. Finally, each of the remaining SKUs, in order of decreasing sales, is allocated to the DC whose contents have the highest average coappearance with the SKU, like in the second stage of the GREEDY SEEDS heuristic.

3.2 Benchmarks

The following two benchmarks emulate how a retailer would allocate SKUs using information other than its full transaction records. The `CATEGORIES` benchmark (Algorithm 5 in the appendix), for instance, uses the retailer's product taxonomy. The algorithm reviews all product categories, sorted by decreasing number of SKUs per category. For each category, all its SKUs that had not been allocated to a DC are added to the largest one with available capacity. Finally, the remaining capacity in each DC is assigned at random among the SKUs that had not been assigned yet to that DC.

The `SALES` benchmark (Algorithm 6 in the appendix), on the other hand, uses regular sales data. This benchmark assigns each of the SKUs, in decreasing order of sales per SKU, to the largest DC with available capacity. Like in `CATEGORIES`, all additional capacity is allocated at random, verifying that every SKU is only assigned once to any given DC.

4 Data and methodology

We tested the effectiveness of the heuristics using a real dataset provided by Yihaodian. Our data contains orders with multiple SKUs and originated in Shanghai. The city was chosen because it is the company's largest and most mature market, in which Yihaodian offers its broadest assortment. We removed all SKUs that were part of a promotion, to avoid biases (promotional SKUs would be expected to be purchased with any other SKU), as well as SKUs sold by third party sellers, the shipment of which is not fulfilled from Yihaodian's DCs. Our dataset includes all the orders for the weeks of December 12 and December 19, 2011, for a

total of 105,745 and 123,416 orders, respectively. A total of 26,356 SKUs were part of multi-SKU orders from Shanghai during those two weeks. Of those, 4,242 SKUs were ordered only the first week and 4,793 SKUs the second week. A total of 17,321 SKUS were active in both weeks. Table 1 describes the transaction data in our sample, including SKUs per order, active SKUs and total number of orders and Figure 2 shows the distribution of the number of SKUs per order: in both weeks less than 15% of the orders had 8 SKUs or more, and those orders with more than 18 SKUs represented less than 1% of the total.

Going forward, we assume that Yihaodian has 3 DCs serving Shanghai, each of which has identical capacity, equal to 70% of the total number of SKUs offered to Shanghai customers during those two weeks: 18,449 SKUs. This assumption regarding capacity is consistent with the company's current plans in a 6- to 18-month horizon.

SKUs per order	Dec 12	Dec 19
Minimum	2	2
1st quartile	2	3
Median	4	4
3rd quartile	7	7
Maximum	93	90
Mean	5.17	5.18
Standard deviation	3.86	3.74
Active SKUs	21,563	22,114
Total orders	105,745	123,416

Table 1: Descriptive statistics of the transaction data for each of the weeks in the sample

From the transaction data we compute two coappearance matrices, one for each week. The off-diagonal elements of each coappearance matrix corresponds to the adjacency matrix of a 26,356-node graph. Both of these graphs show sizable largest connected components: 21,154 nodes (98.1% of total active SKUs) for the week of December 12 and 21,602 (97.7% of active SKUs) for the week of December 19.

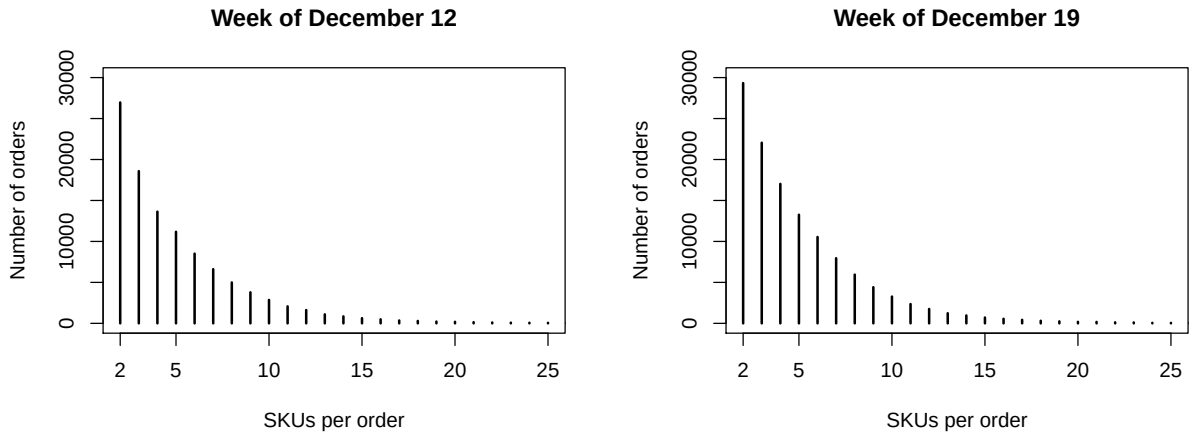

Figure 2: Histograms of SKUs per order (up to 25 SKUs) for each of the weeks in the sample

The neighborhood of SKU s given coappearance matrix $\mathbf{C}$ is defined as $\mathcal{N}_s = \{i : c_{si} > 0, i \neq s\}$. Then, the number of neighbors of s, $|\mathcal{N}_s|$, is the number of SKUs that have been ordered together with s at least once. Figure 3 shows the distribution of the number of neighbors of the active SKUs. This degree distribution shows a thick tail: in both weeks of our sample less than 23% of the SKUs had more than 120 neighbors and those with more than 1,000 neighbors were less than 1.2% of all SKUs. We can find further evidence of the thick tail in Table 2, which shows how the mean number of neighbors is greater than the third quartile for both weeks.

Neighbors per active SKU	Dec 12	Dec 19
Minimum	1	1
1^{st} quartile	8	8
Median	26	28
3^{rd} quartile	98	102
Maximum	7,104	8,813
Mean	115.2	120.3
Standard deviation	280.05	299.63

Table 2: Descriptive statistics of distribution of neighbors per active SKU for each of the weeks in the sample

We ran each of the algorithms against data from the week of December 12 to obtain six

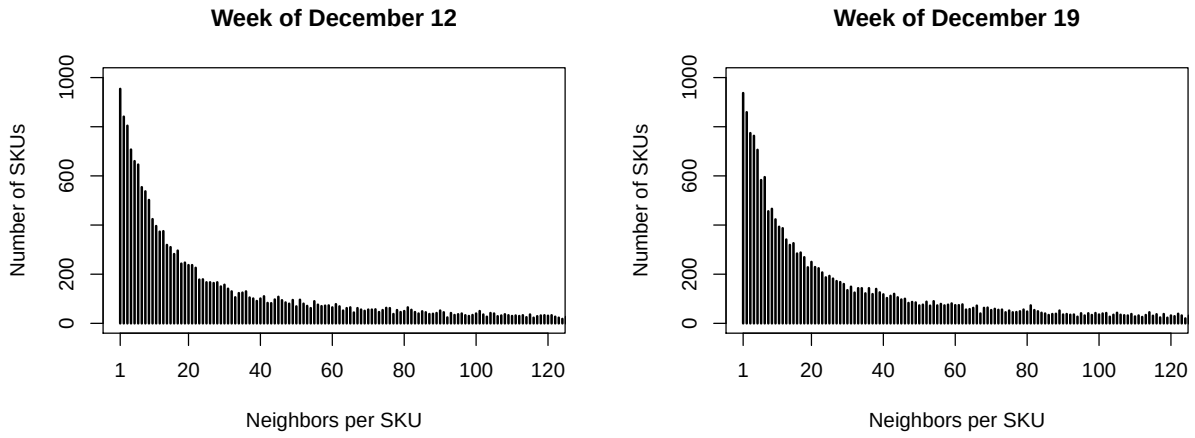

Figure 3: Histograms of neighbors per active SKU (up to 120 neighbors) for each of the weeks in the sample

allocations of the SKUs to the three DCs, one per algorithm. These allocations included all the 26,356 SKUs that appear in either week, although the algorithms only used transaction data from the first week. The 4,793 new SKUs (18.2% of the total) that were not ordered during the week of December 12 had a coappearance equal to zero and, therefore, their allocation was effectively made at random by all the algorithms. The allocations are, then, a worst-case result that could be improved by estimating the coappearance of SKUs that are new to the system by using additional information.

To test and compare the algorithms, we ran the orders from the week of December 19 through each allocation. In each case, we recorded the minimal number of shipments necessary to fulfill each order, as well as the DCs that originated each shipment and the number of combinations of DCs that can fulfill the order with the same minimal number of shipments. The results are described in the next section.

5 Results

We compare the performance of the six algorithms along three dimensions: the average number of shipments, the workload balance across DCs and the flexibility of the allocation.

Average number of shipments. The first block of columns of Table 3 keeps track, for each algorithm, of the number of orders that had to be fulfilled in 1, 2 or 3 shipments, as well as the average number of shipments per order. The last column, *% Split orders*, records the total percentage of orders that had to be split, regardless of the number of shipments. By all accounts, the heuristics we propose outperform the benchmarks considered, with `GREEDY PAIRS` performing better than the other heuristics and `GREEDY SEEDS` and `BESTSELLERS` trailing behind. It is worth noting that the performance of our benchmarks is remarkably similar to the 13-18% of split orders estimated by Yihaodian. In our simulation, using the heuristics leads to a reducion of split orders of up to one third to one sixth of those from the benchmarks.

Workload balance. A potential side effect of each of the algorithms that we propose would be to concentrate the best selling items in a small subset of the DCs. This could cause imbalances in the deployment and utilization of the labor force across DCs and lead to unintended indirect costs. In the second block of columns of Table 3 we record the number of shipments originating from each DC under each of the algorithms. We take the coefficient of variation (standard deviation divided over mean) of the shipments per DC as a measure of the balance of the workload: the higher the coefficient the more unbalanced the split of the load across DCs. All four heuristics outperform the benchmarks we considered, with two of them, `GREEDY`

	Average number of shipments					Workload balance				Allocation flexibility			
	Shipments per order				% Split	Shipped from				Available fulfillment alternatives			
Algorithm	One	Two	Three	Mean	orders	DC 1	DC 2	DC 3	Coeff Var	One	Two	Three	Average
CATEGORIES	88,354	35,062	0	1.284	28.4%	81,710	43,543	33,225	0.483	67,340	47,089	8,987	1.527
SALES	105,096	18,320	0	1.148	14.8%	93,075	23,951	24,710	0.840	74,748	41,019	7,649	1.456
GREEDY ORDERS	117,124	6,292	0	1.051	5.1%	36,361	56,203	37,144	0.260	24,309	22,317	76,790	2.425
GREEDY PAIRS	117,705	5,711	0	1.046	4.6%	37,791	55,380	35,956	0.249	26,335	19,370	77,711	2.416
GREEDY SEEDS	116,503	6,158	755	1.062	5.6%	44,486	45,224	41,374	0.047	34,104	3,884	85,428	2.416
BESTSELLERS	116,479	5,425	1,512	1.068	5.6%	42,195	44,421	45,249	0.036	34,586	0	88,830	2.440

Table 3: Algorithm comparison

SEEDS and BESTSELLERS leading to a particularly balanced outcome.

Flexibility of the allocation. Some orders may have more than one way to be fulfilled with a minimal number of shipments. Suppose, for instance, a retailer that sells 3 SKUs and has 3 DCs, each with capacity for 2 SKUs. Assume that DC 1 holds A and C, and DCs 2 and 3 both hold SKUs B and C. If an order for A and B arrives, the retailer will be able to fulfill it by shipping either from DCs 1 and 2 or from DCs 1 and 3. In this case, an order for A and B has two *fulfillment alternatives*, because both options entail the same number of shipments.

The count of the number of orders in our simulation that have 1, 2 and 3 fulfillment alternatives for each of the algorithms is shown in the third block of columns of Table 3. In general, all four heuristics perform better than the benchmarks in terms of flexibility, with a slight, marginal advantage of BESTSELLERS over the rest. Note that, in the case of BESTSELLERS only one- and three-option orders were possible: an order that comprised only bestselling SKUs (71.98% of the orders) could be fulfilled in one shipment from any of the DCs, while any other order can only be served in a minimal number of shipments by a unique combination of DCs.

Flexible allocations, with multiple fulfillment alternatives for many orders, stem from SKUs redundantly allocated to more than one DC and can mitigate the risk of split orders due to stock-outs. One example: suppose that a particular combination of SKUs is allocated to multiple DCs and one of those SKUs is stocked out in one of the DCs. When an order with that combination of SKUs arrives it can be assigned to one of the other DCs that carry that combination, and thus the retailer avoids having to split the order into multiple shipments.

Allocating SKUs to more than one DC can also help meeting other of the retailer's objectives, such as minimizing shipping time. Some of these additional benefits of flexible allocations are described in a newsletter sent by Amazon to its marketplace sellers in February 2012:

> "Effective February 25, 2012, inventory may be assigned to multiple fulfillment centers . . . to better enable product availability at the customer's preferred shipping speed. We anticipate that both you and customers who purchase your products will benefit from this change, including for the following reasons.
>
> Products located in multiple fulfillment centers can gain a competitive advantage over products located in a single fulfillment center. Order cut-off times . . . can be extended . . . with inventory in multiple fulfillment centers, so your offers will more likely be available for last-minute orders in more locations.
>
> . . . Events such as snowstorms, flooding, and earthquakes can disrupt transportation in a region or local fulfillment center. Having your inventory in multiple locations increases our ability to ship orders to customers on time even if this type of event occurs." (Amazon Services, 2012)

6 Conclusions

Throughout this paper we have highlighted the importance for direct-to-consumer retailers, especially online retailers, of avoiding split orders in order to rein up logistic costs. We have

explored different approaches to the problem and proposed and compared heuristics that can easily be applied and run with basic statistical software.

As a result of our research Yihaodian is set to implement the BESTSELLERS heuristic in its DCs in Shanghai and Beijing once their current DC expansion plans in both regions are finalized. The choice was based not only on the merits displayed by the heuristic in our simulation, but also on its fit within Yihaodian.

As we saw in Section 5, BESTSELLERS showed (Table 3) a potential to reduce the average number of shipments per order similar to that of the other three heuristics. Even though it could be argued that its performance is the worst of the four, the differences between them are relatively small and it also showed the best result both in terms of allocation flexibility and workload balance.

In addition to that, Yihaodian preferred BESTSELLERS for three compelling reasons. First, because its simplicity made it easy to communicate to everyone involved in warehouse and fulfillment management. Second, because the company has an ingrained culture geared at successfully running promotions and putting the spotlight on bestselling products. And third, because in its current growth stage Yihaodian generates most of its revenue from a relatively reduced fraction of its assortment. In this setting, the BESTSELLERS heuristic captures most of the key relationships between SKUs by focusing on the products assigned to all the DCs. However, if demand for niche SKUs grows and Yihaodian starts facing a "long-tailed" demand we would expect GREEDY SEEDS to become comparatively more competitive.

While research streams in information systems and marketing have addressed the chal-

lenges posed by online retail for over a decade, mainstream research in operations management has barely addressed the subject. A plausible explanation would be that, even though the Internet has disrupted the way goods and services are sold, fulfillment operations have, with few exceptions, remained the same as before the rise of electronic commerce, even though order fulfillment is now more complex, faster and carries more volume than before. As retail product offerings and demand expand rapidly, fulfillment capacity, which is inherently inelastic, grows fragmented, leading to significant frictions and costs that open opportunities for improvement throughout the retailer's internal and external supply chain. Strategic components of retail operations such as the topology of the logistics network, the role and processes of fulfillment centers or the flows of goods and information may have to be reassessed, adapted and leveraged to meet challenges of online retail such as cross-channel sales or the optimization of last mile delivery.

We hope that the research reported here will stimulate additional research on online retail operations.

Appendix

Algorithm 1: `GREEDY ORDERS`, greedy allocation by SKUs per order

Data: Matrix of orders **R**, Vector of DC capacities **k**

count the number of SKUs per order, $\mathbf{R} \cdot \mathbf{1}_S$;

sort the orders by decreasing SKUs per order;

sort the DCs by decreasing capacity;

foreach *order o in sorted order list* **do**

  foreach *SKU s in SKUs in order o* **do**

    find the DC d with the lowest index that has one unallocated space;

    if *SKU s is not allocated to DC d* **then** allocate SKU s to DC d;

compute the coappearance matrix **C**;

while *there is unallocated space in the DCs* **do**

  find a DC d that has unallocated space;

  retrieve the list of SKUs allocated to d so far;

  find the SKU s that has the most coappearances with the SKUs in the list;

  allocate SKU s to DC d;

Result: Correspondence between SKUs and DCs

Algorithm 2: `GREEDY PAIRS`, greedy allocation by coappearance of SKU pairs

Data: Matrix of orders **R**, Vector of DC capacities **k**

compute the coappearance matrix **C**;

sort the pairs of distinct SKUs by decreasing coappearance;

sort the DCs by decreasing capacity;

foreach *pair of SKUs p in sorted list of pairs* **do**

  foreach *SKU s in pair p* **do**

    find the DC d with the lowest index that has one unallocated space;

    if *SKU s is not allocated to DC d* **then** allocate SKU s to DC d;

while *there is unallocated space in the DCs* **do**

  find a DC d that has unallocated space;

  retrieve the list of SKUs allocated to d so far;

  find the SKU s that has the most coappearances with the SKUs in the list;

  allocate SKU s to DC d;

Result: Correspondence between SKUs and DCs

Algorithm 3: `GREEDY SEEDS`, greedy allocation by DC seeds and coappearance of SKUs

Data: Matrix of orders **R**, Vector of DC capacities **k**

compute the coappearance matrix **C**;

sort the SKUs by decreasing sales (diagonal of **C**);

sort the DCs by decreasing capacity;

assign the top (best selling) SKU to the top (largest) DC;

for $d \leftarrow 2$ **to** D **do**

    list all the SKUs that have not been allocated yet;

    discard from the list those SKUs that are not in the top decile in terms of SKU sales;

    find the SKU s from the list with the least coappearances with all allocated SKUs;

    allocate SKU s to DC d;

foreach *SKUs s sorted by decreasing sales that has not been allocated yet* **do**

    foreach *DC d* **do**

        retrieve the list of SKUs allocated to d so far;

        calculate the average of coappearances of s with the SKUs in the list;

    find the DC d for which the average of coappearances is maximal;

    allocate SKU s to DC d;

while *there is unallocated space in the DCs* **do**

    find a DC d that has unallocated space;

    retrieve the list of SKUs allocated to d so far;

    find the SKU s that has the most coappearances with the SKUs in the list;

    allocate SKU s to DC d;

Result: Correspondence between SKUs and DCs

Algorithm 4: BESTSELLERS, first bestselling SKUs, then maximum coappearance

Data: Matrix of order types **M**, Vector of order type counts **n**, Vector of DC capacities **k**

$B = \left\lfloor \frac{\sum_d k_d - S}{D-1} \right\rfloor$;

while $\exists d$ *such that* $k_d < B$ **do**

  foreach *DC* d *such that* $k_d < B$ **do**

    assign the k_d top (best selling) SKUs to DC d;

    leave DC d outside of the remaining calculations;

  $B = \left\lfloor \frac{\sum_d k_d - S}{D-1} \right\rfloor$;

if $B = 0$ **then** apply GREEDY SEEDS heuristic;

else if $B > 0$ **then**

  compute the sales per SKU;

  sort the SKUs by decreasing sales;

  sort the DCs by decreasing capacity;

  assign the B top (best selling) SKUs to all the DCs;

  compute the coappearance matrix **C**;

  foreach *SKUs s sorted by decreasing sales that has not been allocated yet* **do**

    foreach *DC* d **do**

      retrieve the list of SKUs allocated to d so far;

      calculate the average of coappearances of s with the SKUs in the list;

    find the DC d for which the average of coappearances is maximal;

    allocate SKU s to DC d;

Result: Correspondence between SKUs and DCs

Algorithm 5: CATEGORIES, allocation by category blocks

Data: Correspondence between SKUs and categories, Vector of DC capacities

count the SKUs in each category;

sort the categories by decreasing number of SKUs;

sort the DCs by decreasing capacity;

foreach *category c in sorted category list* **do**

    find the DC d with the lowest index that has capacity $\geq$ count of c;

    allocate all the SKUs in c to DC d;

while *there is unallocated space in the DCs* **do**

    find a DC d that has unallocated space;

    draw at random a SKU s among those that have not been allocated yet to d;

    allocate SKU s to DC d;

Result: Correspondence between SKUs and DCs

Algorithm 6: SALES, allocation by SKU sales

Data: List of SKUs with their sales, Vector of DC capacities

sort the SKUs by decreasing sales;

sort the DCs by decreasing capacity;

foreach *SKU s in sorted SKU list* **do**

    find the DC d with the lowest index that has unallocated space;

    allocate SKU s to DC d;

while *there is unallocated space in the DCs* **do**

    find a DC d that has unallocated space;

    draw at random a SKU s among those that have not been allocated yet to d;

    allocate SKU s to DC d;

Result: Correspondence between SKUs and DCs

References

Amazon Services. (2012, February). Fulfillment by Amazon Newsletter. Retrieved November 1, 2012, from http://goo.gl/juNB4

Amazon.com, Inc. (2011). *SEC Form 10-K for the fiscal year ended December 31, 2011*. Retrieved November 1, 2012, from http://goo.gl/ifctT

Andreev, K., & Racke, H. (2006, October). Balanced Graph Partitioning. *Theory of Computing Systems*, *39*(6), 929–939.

Birge, J. R., & Louveaux, F. (1997). *Introduction to Stochastic Programming* (First edition). Springer Series in Operations Research. Springer.

Blattberg, R. C., Kim, B.-D., & Neslin, S. A. (2008). *Database Marketing: Analyzing and Managing Customers* (First edition). International Series in Quantitative Marketing. Springer.

Bui, T. N., & Jones, C. (1992, May). Finding good approximate vertex and edge partitions is NP-hard. *Information Processing Letters*, *42*(3), 51–56.

Deloitte. (2011). Deloitte Technology Fast 500: 2011 Winners, Results and Fast Facts. Retrieved November 1, 2012, from http://goo.gl/PsNkc

Feige, U., Krauthgamer, R., & Nissim, K. (2000). Approximating the minimum bisection size. In *32nd acm symposium on the theory of computing* (pp. 530–536). Portland, OR: ACM Press.

Fisher, M. L. (2012, January). *Yihaodian: The No. 1 Store*. The Wharton School, case study.

Karp, R. (1972). Reducibility among combinatorial problems. In R. E. Miller & J. W. Thatcher (Eds.), *Complexity of computer computations* (pp. 85–103). Plenum Press.

Lal, R., Salmon, W., & Weber, J. (2004, March). *L.L. Bean: A search for growth* (Case study No. 9-504-080). Harvard Business School.

Lee, H. L. (2001). *Supply Chain Management in the Internet Age*. Stanford Executive Briefings, taped live on May 23, 2001 at Stanford University. Mill Valley, CA: Stanford Video.

Linoff, G. S., & Berry, M. J. (2011). *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management* (Third edition). John Wiley and Sons.

Schleifer, A. (1993, September). *L.L. Bean, Inc.: Item forecasting and inventory management* (Case study No. 9-893-003). Harvard Business School.

Wulfraat, M. (2012, August). Amazon.com distribution network strategy. Retrieved November 1, 2012, from http://goo.gl/x18FD

Xu, P. J., Allgor, R., & Graves, S. C. (2009, April). Benefits of Reevaluating Real-Time Order Fulfillment Decisions. *Manufacturing and Service Operations Management*, *11*(2), 340–355.