

The Benefits of Re-Evaluating Real-Time Order Fulfillment Decisions

Ping Josephine Xu • Russell Allgor • Stephen Graves

Amazon.com, Seattle, WA

Sloan School of Management, MIT, Cambridge, MA

pingx@alum.mit.edu • rallgor@amazon.com • sgraves@mit.edu

January 2006, revised February 2007, October 2007; January 2008

When a customer orders online, an online retailer assigns the order to one or more of its warehouses and/or drop-shippers, so as to minimize procurement and transportation costs, based on the available current information. However, this assignment is necessarily myopic as it cannot account for any subsequent customer orders or future inventory replenishment. We examine the benefits of periodically re-evaluating these real-time assignments. We construct near-optimal heuristics for the re-assignment for a large set of customer orders to minimize the total number of shipments. Finally, we present evidence of significant saving opportunities by testing the heuristics on order data from a major online retailer.

1. Introduction

In the decade since the Dotcom boom, many online retailers have come of age. The existence and growth of these companies pose new challenges to efficient supply chain management. While their market segments, operational scales, and supply chain structures might differ, they share some common characteristics. Xu (2005) summarizes the salient characteristics of online retailing:

Large scale: Online retailers generally have a very large scale operation and catalog.

Logistics as a matter of trust: Online customers consider the timely delivery of products to be a significant component of trust. Thus, the reliability and efficiency of the supply chain are even more crucial than for traditional brick-and-mortar retailers.

High visibility: The availability of a vast amount of information raises questions about how online retailers should share real-time information with their customers or suppliers.

Assemble-to-order system: A major element of online retailing operations is an assemble-to-order system. The components of the assembly are the items in a customer order, and the final product is an individual customer order. The assembly process is very simple, but the number of final products is exponential in the size of the catalog.

Delay in demand fulfillment: In online retailing, there is typically a time delay between when a demand occurs and when inventory is physically deployed to meet a customer order. By delaying the decisions on how to fulfill customer orders, online retailers can utilize more resources and information to make better decisions.

Retailer-directed demand allocation: In online retailing, customers have limited control on how their demand will be served. An online retailer can utilize all of its warehouses or fulfillment centers to serve the customer demand. This centralized demand allocation provides new opportunities to minimize operating costs.

In this paper, we examine a new order fulfillment problem that is shaped by these characteristics. As will be seen, we can improve order fulfillment by taking advantage of the delay in demand fulfillment and the retailer-directed demand allocation. Our paper seems to be the first model of order fulfillment in online retailing. Survey papers that focus on modeling the general E-Business setting include Simchi-Levi, Wu and Shen (2004), Swaminathan and Tayur (2003), and Johnson and Whang (2002).

When a customer places an order on a website, the online retailer attempts to find the most cost-effective fulfillment plan subject to any shipping preferences given by the customer (e.g., two-day delivery). The fulfillment plan specifies for each item in the order, the order fulfillment center or drop-shipper (for simplicity, we refer to both as warehouses) from which the item will be shipped. The fulfillment plan also determines for each item in the order the time at which it can be shipped; this time availability depends on whether there is inventory at the warehouse and if not, when the next replenishment order will arrive at the warehouse. The cost of a fulfillment plan is primarily the transportation costs of shipping the order from the warehouse(s) to the customer location. For orders with multiple items, the online retailer might not be able to ship the order from one location or at the same time. As a result, the online retailer will split the order into multiple shipments that are made from multiple warehouses and/or at different times.

We refer to this search for a cost-effective fulfillment plan as the order-warehouse assignment decision. An online retailer makes these order-warehouse assignment decisions in real

time when each customer places an order. There are several reasons why these decisions are made at the instant the customer orders. First, the online retailer wants to provide in real time a service expectation to the customer. It quotes an estimated-to-ship date as well as the expected number of shipments to the customer; these service expectations depend on the fulfillment plan. This instant feedback allows for the customer to change her shipping preferences accordingly in real time. Second, the online retailer wants to set this service expectation so that it can be met with a very high probability. Once the customer commits to the purchase, the online retailer virtually reserves the inventory in the fulfillment plan for the order so as to deliver reliably on the service expectation. Third, the online retailer needs to be able to reliably make these order-assignment decisions in a 24/7 operating environment that processes up to millions of orders per day. Due to variability in order arrivals over a day, the order execution system needs to complete hundreds of order transactions per second at the daily peak. As a consequence, to reliably handle this volume of transactions the online retailer resorts to a myopic solution approach to the order-warehouse assignment decision; that is, at the time each order is placed, the online retailer quickly searches for a feasible fulfillment plan for the order and then virtually reserves inventory in this fulfillment plan.

We observe that there is always some time delay between when a customer order is received and when it gets picked and assembled for shipping. For some orders this delay is due to some items not being in inventory, and hence the order must wait, usually on the order of a few days, for the inventory replenishment of the missing items. When the customer opts for delayed or free shipping, then the online retailer has a longer time window in which to pick and ship the order. In other cases, even when all of the items are available in inventory, an order will typically wait a minimum of *several hours* before being picked and assembled into a shipment. There are two reasons for this. First, the online retailer maintains a picking backlog or queue of work at each warehouse so as to smooth the workload over a day. Second, some orders will be delayed to allow orders with higher shipping priority to be picked and shipped first.

This delay in demand fulfillment provides an opportunity to revisit the order-warehouse assignment decisions. In this paper, we pose the question of whether or not there is value in re-evaluating these real-time decisions for an online retailer; that is, might we use the time delay between when an order is placed and when it gets fulfilled to improve the decisions of assigning orders to warehouses. Furthermore, if there is a benefit from re-evaluation, we wish to determine how to do the re-evaluation given the operating constraints and conditions of

the online retailer. We report in this paper the research we did to address these questions.

We present Example 1.1 to illustrate the assignment decisions as well as the opportunities from re-evaluation.

Example 1.1. We have four customer orders, labeled as O1, O2, O3, O4 in the sequence of arrival, and three warehouses, labeled as W1, W2, W3. The warehouses carry five SKUs: a CD, a book, a toy, a camera, and a DVD. We depict the real-time assignment in Figure 1. The first customer order is for the book; the online retailer assigns it to W3, possibly because

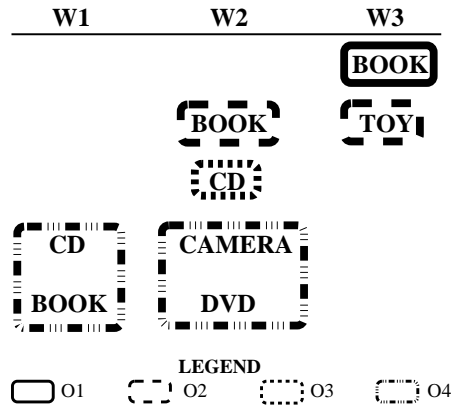


Figure 1: Read-Time Assignments, Six Shipments – Example 1.1.

the first customer is nearest to W3 or W3 is the only warehouse with the book in stock at the time. The second order O2 consists of the book and the toy. Suppose that at the time of O2, W3 has zero uncommitted book inventory: O1 took the last unit of book in W3. Hence, the online retailer assigns the book to W2 and the toy to W3. The online retailer assigns O3 to W2. Finally, there are two shipments for customer O4: the CD and book from W1 and the camera and DVD from W2. Thus, there are six shipments for the four orders.

For this example we now ask is it possible to reduce the transportation costs by adjusting the assignments. As an approximation we focus on reducing the number of shipments. In the transportation cost to ship a package, the fixed cost component is very significant. As an illustration, we display in Figure 2 the UPS US Ground Commercial rates (UPS 2004) for shipping to Zone 1 from Zone 2 (the closest zone) and from Zone 8 (the farthest zone). In both instances the shipping cost consists of a fixed cost of about \$5 per shipment, plus a variable cost that is approximately linear in the weight of the package. For small shipments the fixed cost represents the majority of the shipping costs. As a consequence, reducing the number of shipments is a very good proxy for minimizing the transportation costs for an

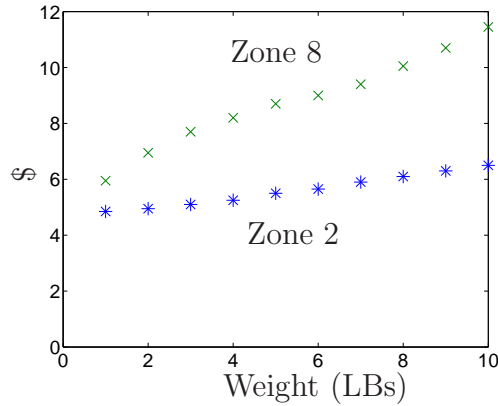


Figure 2: UPS Ground Commercial Rates Within the US Continent

online retailer that primarily ships packages weighing less than ten pounds. For example, consider an order that weighs about eight pounds. It is cheaper to ship a single package of eight pounds from Zone 8 than to ship two four-pound packages from Zone 2. The difference is even more pronounced at smaller weights, as would be the case for small books, CDs, and DVDs. There is also a savings in packaging costs from using one versus two boxes. Therefore, minimizing the number of shipments is a prime objective for online retailers.

Revisiting Example 1.1, we see that we can reduce the number of shipments from six to four (illustrated in Figure 3), which is clearly the best we can do. We assign the first

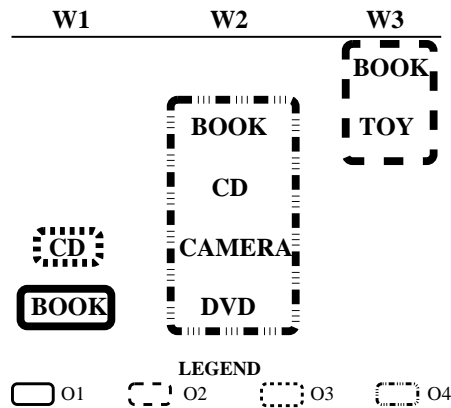


Figure 3: Re-Evaluation Reduces Number of Shipments from Six to Four – Example 1.1.

customer order O1 to W1, O2 to W3, O3 to W1, and O4 to W2.

In this example the initial assignment decisions are necessarily myopic because the online retailer does not anticipate any future customer orders or inventory replenishment. In practice, the real-time assignment is myopic because the online retailer needs to make instantaneous assignment decisions for the reasons described earlier. We conjecture that we

can reduce the total cost of shipping orders from warehouses by periodically re-evaluating the real-time assignment decisions, subject to not violating the estimated-to-ship date expectation for any customer order.

This shuffling of assignments is also practically feasible because of the delay in demand fulfillment. Thus, there is an opportunity to revisit and re-evaluate the real-time assignment decisions. We consider the queue of *not-yet-picked* customer orders at an instant in time and their real-time warehouse assignments. We re-evaluate these real-time decisions to reduce the number of shipments without violating the estimated-to-ship date expectation for any of these orders. We take inventory availability and the estimated-to-ship dates as given. We term this the re-evaluation problem.

We develop efficient and easy-to-implement heuristics to solve the re-evaluation problem. Given the real-time assignment decisions, we take the natural path to construct an improvement heuristic that starts with a feasible solution and iteratively finds better solutions.

In the following sections, we discuss the problem formulation and our heuristic solution approach. We also report our computational experiments on sets of real data from a global online retailer for the purposes to evaluate the heuristics and to assess the potential benefits from solving the re-evaluation problem.

2. Problem Formulation

We formulate the re-evaluation problem as a fixed-charge multi-commodity network flow problem with a time dimension. We refer the reader to Xu (2005) for other ways of formulating the problem. We start with the following notation.

k	index for warehouses, K is the number of warehouses
j	index for customer orders, J is the number of orders
i	index for SKUs, I is the number of SKUs
t	index for time, T is the length of the time horizon
d_{jt}^i	demand for SKU i in order j with estimated-to-ship date t
$D_{jt}^i = \sum_{\tau=0}^t d_{j\tau}^i$	cumulative demand for SKU i in order j as of time t
s_{kt}^i	supply for SKU i that is first available at warehouse k in time period t
$S_{kt}^i = \sum_{\tau=0}^t s_{k\tau}^i$	cumulative supply for SKU i at warehouse k as of time t

The decision variables are:

- x_{jkt}^i units of SKU i shipped at time t from warehouse k to fill customer order j
- y_{jkt} binary variable to indicate a shipment at time t from warehouse k to order j

We denote the following formulation as \mathcal{MIP} .

$$\begin{aligned}
\min \quad & \sum_{j=1}^J \sum_{k=1}^K \sum_{t=0}^T y_{jkt} \\
\text{s. t.} \quad & \sum_{\tau=0}^t \sum_{k=1}^K x_{jk\tau}^i \geq D_{jt}^i, \quad \forall i, j, t \\
& \sum_{\tau=0}^t \sum_{j=1}^J x_{jk\tau}^i \leq S_{kt}^i, \quad \forall i, k, t \\
& 0 \leq x_{jkt}^i \leq D_{jT}^i y_{jkt}, \quad \forall i, j, k, t \\
& y_{jkt} \in \{0, 1\}, \quad \forall j, k, t
\end{aligned}$$

The first constraint assures that the demand is met for each SKU in each customer order before or at the date needed. The second constraint assures that the amount of each SKU shipped from each warehouse does not exceed the available supply. The third constraint assures that we can send SKU i from warehouse k to satisfy order j at time t only if there is a shipment from warehouse k for order j at time t . Problem \mathcal{MIP} has JKT binary variables and $IJKT$ continuous variables. It has $IKT + IJT + IJKT$ number of constraints. The number of constraints and variables is linear in the input data. Xu (2005) shows that the general re-evaluation problem is NP-hard.

Our problem formulation is a network design problem that has a rich literature. Magnanti and Wong (1984) provide a survey of models and classic solution methods of the general network design problem up to 1984. Minoux (1989) surveys the models and solution methods of the variants of the general problems. Balakrishnan, Magnanti, and Mirchandani (1997) list annotated bibliographies on network design since 1985.

3. Exploratory Data Analysis

In this section, we summarize the important characteristics of the customer orders and the real-time assignments in the online retailing setting that motivate this research. Our empirical research reveals some typical customer online purchasing behaviors as well as the nature of a complex customer order-SKU network in the real world. By exploiting the results of our analysis, we are able to construct simple and effective solution procedures.

3.1 Summary Statistics

To facilitate the presentation, we introduce the following definitions.

Definition 3.1.

A *single order* is a customer order that consists of exactly one unit of one SKU.

A *multi order* is a customer order that consists of more than one unit; it would typically entail multiple SKUs, but could also be an order for multiple units of one SKU.

A *split order* is a multi order that requires more than one shipment to complete.

A *single shipment* is a one-unit shipment of a split order, that is, a shipment of a single unit that is part of a multi order.

A *double shipment* is a two-unit shipment of a split order.

Recall that once customer orders are placed, the online retailer assigns each order to one or more warehouses and enters each item into the picking queue at its designated warehouse. We take a snapshot of all orders that are waiting to be picked at a time epoch. We present in Table 1 the summary statistics from five random days in 2004 for a large online retailer. The online retailer is subject to two demand seasons each year: the peak season which lasts for about 20% of the year and the off-season which lasts for the remaining 80% of the year. In Table 1 we report representative data from four random days during the off-season in 2004 and one random day from the peak season in December 2004. The term "*Total*

	Off-Peak Season				Peak Season
	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5
Total orders	869K	923K	918K	956K	1.55M
Total SKUs	411K	385K	388K	406K	526K
Single orders	64%	65%	66%	65%	56%
Multi orders	36%	35%	34%	35%	44%
Split orders	3.9%	3.7%	3.7%	3.6%	6.4%

Table 1: Snapshot Data

orders" represents the total number of customer orders that have not yet been released to be picked. The term "*Total SKUs*" is the number of unique SKUs among these orders. "*Single orders*", "*Multi order*", and "*Split orders*" are the percentages of single,

multi, and split orders among “Total orders”. Overall, the snapshot data is very consistent from day to day during the off-peak season. There are close to 1 million orders with 2 to 3 million units in the not-yet-picked queue for the snapshot data. The peak season data differ in that there are a greater percentage of multi orders and split orders.

The real-time assignments in the snapshot data split about 10% of the multi orders in the off-peak season and 15% in the peak season. Overall, the number of shipments in each split order is two or three shipments with few exceptions. In Table 2 row *"2-shipment"* represents

Split Orders	Off-Peak Season				Peak Season
	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5
2-shipment	92%	92%	93%	93%	94%
2 or 3-shipment	98%	99%	99%	99%	99%
sgl shipment	80%	86%	86%	86%	74%
sgl or dbl shipment	93%	93%	92%	93%	89%

Table 2: Snapshot Data – Split Orders

the percentage of split orders that have exactly two shipments; row *"2 or 3-shipment"* represents the percentage of split orders that have either two or three shipments. Moreover, the row *"sgl shipment"* represents the percentage of split orders that have at least one single shipment; the row *"sgl or dbl shipment"* represents the percentage of split orders that have at least one single or double shipment.

These summary statistics provide important insights into the nature of the split orders, as well as possible approaches for “fixing” these split orders. In particular, we observe the following:

- In the not-yet-picked queue, there is a high percentage of single orders; these orders are extremely flexible as they can be easily reassigned from one warehouse to another warehouse.
- Over 90% of split orders consist of two shipments; 99% of split orders consist of two or three shipments. Thus the split orders are not very fragmented as they usually have only two or three pieces to put back together.
- Most split orders have at least one single shipment. That is, one of the pieces is just a single unit that is being shipped from another warehouse.

These three observations are critical in guiding the construction of our solution approach. The problem objective is to reduce the number of split orders created by the real-time assignment. In effect, the challenge is to try to put the pieces of a split order back together so that the order can be shipped from a single warehouse. These observations suggest that there are usually just two pieces associated with each split order and at least one of them is a single shipment. Furthermore, there are many single orders, which are potential candidates for exchanges. As will be seen, our solution approach exploits these observations.

3.2 Decomposition

A natural question is whether the re-evaluation problem can be decomposed into a set of smaller problems. For instance, one might hypothesize that we could partition the customer orders according to product categories; that is, we might segment the orders into orders for books, hand tools, kitchen appliances, electronics, etc. If so, then possibly one could solve a smaller re-evaluation problem for each customer segment or product category.

To investigate this question, we examined the connectivity of the order-SKU graph constructed for the snapshot of the not-yet-picked queue. In this graph there is one node for each SKU and we place an arc between a pair of nodes only if there is a multi order that includes both SKUs; we set the arc weight to be the number of multi orders that contain both SKUs. We report our findings for one representative data set, taken from an off-peak season day (Data set 2 in Table 1) in 2004. Our analysis on the other data sets exhibits very similar behavior. In Table 3 we report the characteristics describing the connectivity of the order-SKU graph. Each component is a connected sub-graph; that is, there is an undirected path between each pair of nodes in the component and there are no paths between a node in one component and a node in another component. The first column "*Component Size*" in the table is the number of nodes in a component and the second column "*Component No.*" indicates the number of components with that size. The columns "*SKUs*", "*Orders*" and "*Splits*" indicate the percentage of each entity that is contained within the components of a given size. By examining the number and size of the components, we can assess the possibilities for decomposing the re-evaluation problem into smaller problems.

The Order-SKU graph has one enormous component with over 274K nodes, where each node is an SKU. This component accounts for 84% of all orders, 71% of all SKUs and 93% of all split orders.

Component Size	Component No.	SKUs	Orders	Split Orders
274K	1	71%	84%	93%
5 - 43	1.1K	1.9%	0.6%	0.9%
4	1.0K	1.0%	0.3%	0.6%
3	2.5K	2.0%	0.7%	1.5%
2	8.5K	4.4%	2.0%	3.7%
1	74K	19.3%	12.6%	0.0%
Total	87K			

Table 3: Components in an Order-SKU Graph

The next largest component has 43 nodes (SKU's)! There are over 1.1K components with sizes ranging from 5 to 43 nodes; these components include less than 1% of the split orders. There are 12K components with two, three or four nodes; these components represent 3% of the orders, 7% of the SKUs and 6% of the split orders.

Finally, there are over 74K components with a single node. Each of these components corresponds to a single SKU, where each of these SKUs is not part of any multi order with another SKU.

Thus, from Table 3, we do not find a simple way to decompose the problem. Although the order-SKU graph does split up into thousands of components, effectively all of the action remains in the one big component, which captures 93% of the split orders.

A second question is whether the largest component has any structure that might help simplify the problem. For instance, how would the removal of a small subset of SKUs affect the connectivity of the graph? To explore this question we did further analysis on the largest component. We report the distribution of the node degree in the largest component in Table 4. The degree of a node is the number of incident edges or connected nodes. In each row we report the number and percent of SKU's with node degree within a specified range. *The size of the range increases exponentially.* For instance, there are 59K SKUs with node degree between 4 and 7; this corresponds to 21% of the SKUs in the largest component. The node degree for an SKU is a measure of the popularity of the SKU as it indicates the number of other SKUs with which it shares at least one order. The degree distribution of the largest component appears to be highly right-skewed with a power-law tail, as indicated by an approximately straight-line on a doubly logarithmic scale graph. This seems consistent with some of the complex networks in the real-world. (Newman, 2003)

We focus on the SKUs in the tail of the distribution reported in Table 4, as possibly they are the glue that provides the connectivity of the big component. We incrementally

Node Degree	Number of SKU's	% of SKU's
1	21K	7.8%
2-3	50K	18%
4-7	59K	21%
8-15	54K	20%
16-36	47K	17%
37-92	31K	11%
93-241	105K	3.8%
242-645	17K	0.6%
646-1743	199	0.07%
1744-4725	26	0.01%
4726-	3	0.00%

Table 4: Degree Histogram of the Largest Component

removed the SKU's with the *highest* degree to see whether and how we might split up the big component. We report a summary of our findings in Table 5. If we remove the three SKUs

No. of Nodes Removed	No. of Components	Largest Component		
		SKU %	Order %	Split Order %
3	67	99.8%	97.9%	98.9%
250	625	98.5%	78.2%	91.6%
10000	2902	89.2%	55.2%	75.7%

Table 5: Decomposition of the Largest Component by Removing Nodes

with the highest degree, the big component splits into 67 components. However, one of these 67 components is still very large, retaining 99.8% of the SKUs from the large component, 97.9% of the orders and 98.9% of the split orders. Thus, removing these three highest-degree SKUs does not accomplish much. We see that this phenomenon persists as we remove more and more of the SKUs with high degree; the remaining large component retains 92% of the split orders after removing 250 SKU's and 76% after removing the most popular 10,000 SKUs.

We did a similar analysis on the arcs, where we sequentially removed the arcs with the largest weights. Again this does not reduce the size of the largest component as illustrated in Table 6. By removing the top three arcs, we removed the associated 5255 orders. The resulting largest component still contains most of the SKUs and orders.

From this analysis of the order-SKU graph we conclude that there does not seem to be an easy way to reduce the size of the re-evaluation problem. It seems that the order-SKU graphs exhibits the "small-world effect" found in many complex networks (Newman, 2003),

No. of Arcs Removed	No. of Orders Removed	No. of Components	Largest Component		
			SKU %	Order %	Split Order %
3	5255	9	99.9%	98.0%	99.7%
203	51,174	303	97.9%	82.8%	84.7%
5850	125,167	1599	90.3%	57.6%	62.7%

Table 6: Decomposition of the Largest Component by Removing Arcs

whereby most pairs of nodes seem to be connected by a short path through the network. In this online retailing context, some multi-item orders can and do contain SKUs from quite disparate product categories. This results in a very high level of connectivity in that we can find an “order path” between most pairs of SKU’s; that is, we can find a series of SKU’s for which each successive pair is contained within an order. Furthermore, this connectivity does not depend upon a small set of high-degree nodes (or heavy-weight arcs); removing the most popular SKUs (or arcs) has a very limited impact on the connectivity of the order-SKU network.

4. Heuristic Approach

In this section, we develop a heuristic solution approach. We start with an initial feasible solution, namely the real-time assignments and develop an improvement algorithm, by which we iteratively create better solutions. The primary reason for considering this type of heuristic is our desire to develop an approach that has a chance of being implemented into a real-time order execution system for an online retailer. More specifically we provide the following arguments as rationale for this approach.

Whereas we can formulate the re-evaluation problem as a large-scale optimization problem, we cannot solve real-world problems with the currently available solution methods. Furthermore, as detailed in the prior section, the problem does not decompose into smaller problems and it is necessary to solve the re-evaluation problem considering all orders and all SKUs.

The implementation of the solution of the re-evaluation problem would entail periodically interfacing with a real-time order execution system. The order execution system handles hundreds of thousands of transactions each day and is the lifeblood of the online retailer. As a consequence, any changes to this system get extraordinary scrutiny to assure that there is *no risk* to what is already working. This raises a couple of considerations. We

need to always maintain a feasible solution, and we need to account for the dynamic nature of the order execution system. Our heuristic algorithm assures that we always maintain a feasible solution. After each iterative step we can implement the recommended changes to the incumbent assignments to get an improved order assignment. This facilitates implementation as we always have a feasible solution, even if there were a sudden termination of the algorithm.

Neighborhood search is the inspiration of our proposed heuristics. We start with a feasible solution, which is the assignment decision made at the time of customer order. At each iteration, we find a better solution from searching the “neighborhood” of the current solution. The neighborhood is limited either by the set of customer orders or inventory we consider at each iteration. We refer the readers to an extensive survey by Aarts and Lenstra (1997). Ahuja, Ergun, Orlin, and Punnen (2002) provide a comprehensive survey on very large-scale neighborhood search techniques.

Any solution for the re-evaluation problem depends on data from the order execution system on the current status of the inventory and the not-yet-picked queue at each warehouse in the system; however, the “current status” as maintained by the order execution system is just that, namely the status at the time epoch at which the data was retrieved. Thus, a solution approach that requires a lot of data concurrently and/or a lot of time to solve will be difficult to implement, as it will not be able to keep up with the changes to the status information. This motivates us to consider local search algorithms that can quickly search the neighborhood of the current solution at each iteration. Consequently we need to retrieve only a small amount of data pertaining to the neighborhood for each iteration.

As a final point, our analysis of the order data suggests that fixing the split orders might not require an extremely complex algorithm. Most orders are split into two shipments. Most split orders have at least one single shipment. And there is a large number of single orders in the system, which can be readily reassigned. Thus, this analysis indicates that we might find very good solutions with relatively simple heuristics. As we will see, this seems to be the case, as we find that our heuristic solution is near optimal on a set of test problems.

Our heuristic approach consists of two distinct parts. We name the first part **Order Swap** because we consider split orders one at a time and examine possible swaps. The second part is **SKU Exchange** because we consider one SKU at a time and examine possible exchanges. In our implementation, we find that it is best to start with Order Swap and then proceed to SKU Exchange on the remaining split orders. We view Order Swap as a fast and extremely simple greedy algorithm to exploit the abundance of single orders and unassigned

inventory. To extract additional benefits, we employ the efficient but more time consuming SKU Exchange heuristic.

To simplify the presentation of the heuristics, we will treat any unassigned inventory as if it were a single order. In particular, we refer to an unassigned unit as a single order for the SKU with an estimated-to-ship date set equal to the end of the planning horizon. We also assume that all on-order inventory arrives to the warehouse at the due date promised by the supplier; this assumption is consistent with current planning practices and is reasonable given the general reliability of suppliers.

4.1 Order Swap

Order Swap exploits the flexibility of single orders, including any unassigned inventory. We examine each split order in the initial assignment. We first define the following parameters for each item within the order:

- v_{ij} estimated-to-ship date of item i in the real-time assignment of order j ,
- u_{ij} time at which the inventory assigned to item i for order j arrives to the warehouse.

We require $u_{ij} \leq v_{ij}$ for the assignment of item i to be feasible for order j . Then we can swap the assignments for item i from order 1 with item i from order 2 if and only if $u_{i1} \leq v_{i2}$ and $u_{i2} \leq v_{i1}$. That is, we can swap the assignments for item i for two orders only if we can do so without violating their estimated-to-ship dates.

The items in a split order j may have different estimated-to-ship dates. For each order j , let t_j be the desired estimated-to-ship date for order j so that the split order may be shipped in one shipment. Then t_j is the earliest estimated-to-ship date among all items in the order. That is, if every item in the order is shipped at t_j , then no estimated-to-ship date is violated.

Let k be the index for warehouses. For each split order j , we examine each warehouse k . If k has sufficient single orders to swap with j such that the entire order of j can be fulfilled at k , then we complete the swap. We terminate consideration of order j if a swap involving j has occurred or when all warehouses have been checked. Algorithm 1 specifies the general implementation for each order j .

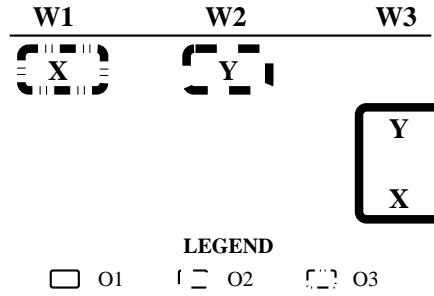


Figure 5: Order Swap Example 4.1– After a Swap.

the affected orders, and continue with the next SKU. We terminate at the end of the SKU sequence. The transportation problem allocates the supply of the SKU at the supply nodes (the warehouses) to the demand nodes (orders that include the SKU).

Input: an initial feasible solution
foreach $SKU\ i = 1, \dots, n$ **do**
 Construct a transportation problem for SKU i ;
 Solve the transportation problem i ;
 Update orders affected by i ;
end

Algorithm 2: SKU Exchange

We consider a SKU i only if two conditions are true. First, there needs to be at least one split shipment with a single shipment of i . Second, there must be at least one single order (or equivalently one unit of unassigned inventory) of item i somewhere in the system. We divide the time horizon into $T+1$ time buckets, $t = 0, \dots, T$. Each time bucket would typically represent a day with $t=0$ being the current day that the snapshot is taken and $t=T$ representing any commitments beyond the T day horizon.

Figure 6 is a representation of a transportation problem for one SKU.

Supply: We group the supply nodes into $T+1$ supply blocks. Supply block 0 represents the currently available inventory in the warehouses. Supply block $0 < t < T$ represents the on-order inventory that is scheduled to arrive in t days from the current date. Supply block T represents the currently on-order inventory that is scheduled to arrive on or after T days. For each supply block t and each warehouse k , we have a supply node, denoted by w_{kt} . We denote the supply for the supply node of warehouse k in time block t as s_{kt} .

Demand: We group the demand nodes into T demand blocks. Demand block t contains the

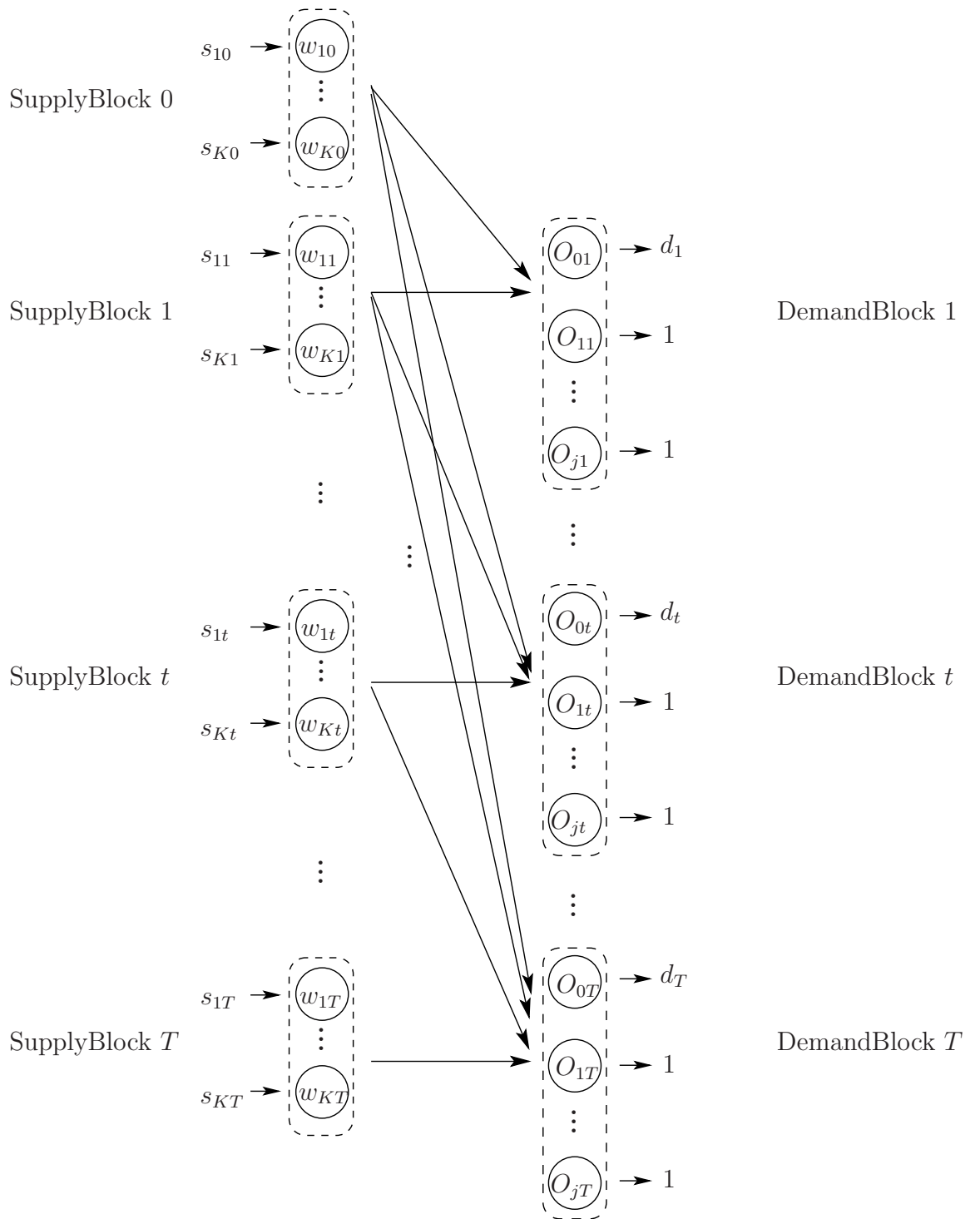


Figure 6: Transportation Problem for One SKU

nodes for the customer orders for SKU i that have estimated-to-ship dates in t days. There are two types of customer orders in each demand block: the single orders of SKU i and the multi orders that have a single-item shipment of SKU i . Demand block t has a single node O_{0t} with demand d_t representing all single orders with estimated-to-ship date of t days from now. Demand block t also has node O_{jt} with demand 1 to represent the single-item shipment for each multi order j .

Arcs: We permit arcs from each node in a supply block t_2 to each node in a demand block t_1 , $\forall t_1 \geq t_2$. A unit of flow from warehouse k in supply block t_2 to a demand node j in demand block t_1 represents that we ship a unit of SKU i that is available at warehouse k at time t_2 to satisfy order j 's requirement at t_1 . Thus, we ensure that we assign inventory to an order only if it can satisfy the estimated-to-ship date for the order.

Arc Costs: We solve the transportation problem by minimizing the total cost on the arc flows. The cost for each arc can either be 0 or -1. We set the cost to be -1 if a unit flow on the arc will reduce the number of shipments by one. Otherwise, we set the arc cost to be zero.

We illustrate the transportation problem using the following example.

Example 4.2. We consider the three orders with the real-time assignment and the estimated-to-ship dates in Table 7. We consider SKU Y, as it has a single order O2 and has single

Order SKU	O1		O2	O3		
	Y	Z	Y	Y	U	V
estimated-to-ship date, v	9	9	9	9	9	9
inventory available date, u	9	0	0	9	0	0
assigned warehouse	W2	W3	W3	W1	W2	W2

Table 7: Real-Time Assignments – Example 4.2.

shipments in two multi orders, O1 and O3. We construct the corresponding transportation problem for SKU Y in Figure 7. We have two supply blocks where one unit of Y is currently available (in supply block 0) in warehouse 3 and two units will arrive in 9 days (supply block 9), one each for warehouse 1 and 2. We have one demand block since all orders are due in 9 days. Here each demand node represents an order with one unit of Y as a single-item shipment. The dark arcs have a cost of -1, since, for example, by assigning the on-hand unit

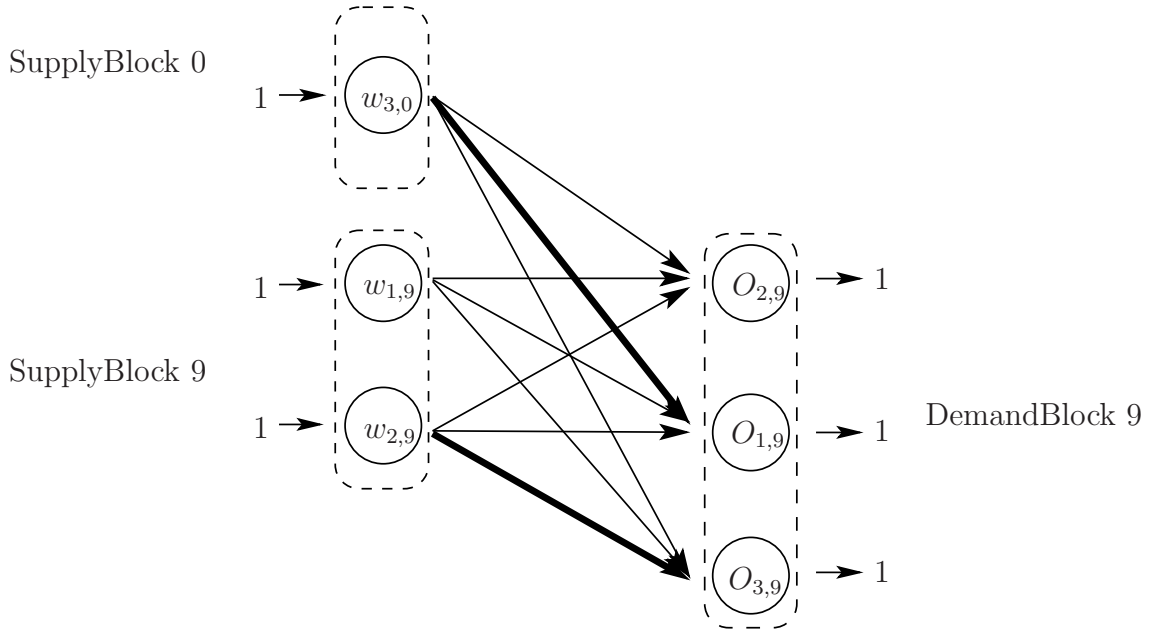


Figure 7: Transportation Problem for SKU Y – Example 4.2

of Y at warehouse 3 to $O1$ will reduce the number of shipments in Order 1 by one. Similarly, if we assign the arriving unit at warehouse 2 to Order 3, we will reduce the number of shipments for Order 2 by one. The arc cost is zero for all other arcs.

By inspection, we see that the optimal solution is to send one unit of flow along arcs, $(w_{3,0}, O_{1,9})$, $(w_{1,9}, O_{2,9})$ and $(w_{2,9}, O_{3,9})$. The optimal solution corresponds to the results in Table 8. We reduce the number of shipments in the three orders from 5 to 3.

Order SKU	O1		O2	O3		
	Y	Z	Y	Y	U	V
estimated-to-ship date, v	9	9	9	9	9	9
inventory available date, u	0	0	9	9	0	0
assigned warehouse	W3	W3	W1	W2	W2	W2

Table 8: Re-Evaluation Reduces Number of Shipments from 5 to 3 – Example 4.2.

One practical advantage of the SKU Exchange is the ability to solve and implement one subset of exchanges at a time while maintaining a feasible solution at each iteration. While the primary objective is to minimize the number of shipments, we also want to do so without implementing any unnecessary exchanges (which can occur due to the high likelihood of alternative optimal solutions). That is, we have a secondary objective to minimize the number of exchanges from the initial real-time solution. This objective is easy to add to the

current transportation problem by modifying some arc costs. Let j be a demand node and k be a supply node. Arc (k, j) has an additional cost of $-n\epsilon$ if n units of the supply at supply node k are currently assigned to the demand node j . The value of ϵ is specified as $0 < \epsilon < 1$. To ensure the objective of minimizing the number of exchanges is a secondary objective, we set the value of ϵ as $\epsilon < \frac{1}{m}$, where $m = \sum_{\forall k,t} s_{kt}$ is the total number of supply units in the transportation problem.

In some instances, an online retailer might want to minimize the transportation costs, instead of the number of shipments. Recall we approximate the transportation costs by the number of shipments, because the items we consider tend to have a small weight and thus have a significant fixed cost for shipment. We can easily extend the transportation problem to incorporate actual transportation costs. We illustrate the cost modification with the previous example, Example 4.2, on SKU Y. We set $c_j(k)$ to be the transportation cost to ship order j if its unit of SKU Y is shipped from warehouse k and the assignment of the rest of the order remains the same. For example, if $j = 1$, then $c_1(3)$ is the cost of shipping O1 whose one unit of SKU Y is shipped from warehouse 3. Given Z is also shipped from warehouse 3, $c_1(3)$ is the cost of shipping one package containing Y and Z from warehouse 3. Let warehouse k_j be the current assignment of SKU Y in order j . Then, we set the cost of arc (k, j) as $c_{kj} = c_j(k) - c_j(k_j)$, which is the additional transportation cost of shipping order j if its unit of Y is shipped from warehouse k instead of k_j .

For the real data sets that we have examined, a large percentage of the split orders in the real-time assignment (over 85%) have at least one single shipment. A larger percentage (94%) of the split orders have at least one single or double shipment. As one refinement to the SKU Exchange heuristic, we can add all orders with double shipments that include the SKU; this will provide more options for the SKU Exchange heuristic. The arc costs in the transportation problem are now harder to specify. For double shipments, we can reduce a split only if both units can be re-assigned. For a specific SKU, we can set the arc cost to be an estimate of the probability that the other SKU in the double shipment can also be re-assigned to the same warehouse. For example, if the other SKU has ample available inventory, we could set the current arc cost to a high probability. We conjecture that we do not need a precise estimate. Rather the intent is to set the cost parameter so that the transportation algorithm will make such exchanges as possible, but not at the expense of any exchanges that are certain to reduce the number of shipments.

Xu (2005) shows that both Order Swap and SKU Exchange heuristics can perform arbitrarily badly. For Order Swap, the worst cases occur when single orders are scarce and the multi orders are large in size. Since the majority of the split orders have two shipments in the real-time assignment and there is an ample amount of single orders and available inventory in practice, Order Swap performs well in practice. For SKU Exchange, the worst cases occur when no warehouse among the currently assigned warehouse of an order ever carries all the SKUs in the order. The two heuristics complement each other. Order Swap complements SKU Exchange’s shortcoming by considering shipping the order from warehouses that are not currently assigned to the order. On the other hand, SKU Exchange relies much less on the abundance of single orders or available inventory than Order Swap.

5. Computational Tests

We implement the heuristics on several real data sets from a large online reatailer; these data sets are similar in structure to those presented in §3 but come from a different set of days in 2004. To examine the sub-optimality of the heuristics on the data, we would like to compare the heuristic solutions to the optimal solutions. However, the entire re-evaluation problem is too large to be solved optimally in CPLEX due to the lack of computer memory. Instead, we created from the real data a set of smaller test problems to evaluate the near-optimality of the heuristic solutions. In this section, we first discuss the results from the test data sets, and then present the benefits in the real data sets.

5.1 On the Reduced Data

The snapshot data of the not-yet-picked orders will typically include orders that were placed within the last several days of the snapshot time. To create the smaller test problems, we select only the orders that were placed on the day of the snapshot date. The resulting orders are the reduced test data. The reduced test data generally have 110-120K orders, 100K SKUs, and 7 warehouses. We were able to solve this re-evaluation problem with *no time dimension* in CPLEX within minutes (Allgor and Stratila 2004). That is, we assume all orders in the test data have the same estimated-to-ship date and all inventory is available by that date. The reduced data set with time dimension is again too large to solve.

Table 9 displays the summary statistics for the test data. We observe that overall the data sets are very similar. The columns "*Single orders*", "*Multi orders*", "*Split*

Data Set	Single orders	Multi orders	Split orders	Splits
a	56.7K	62.7K	10.5K	11.3K
b	55.2K	65.0K	10.2K	10.9K
c	52.4K	61.8K	9.6K	10.2K
d	45.8K	54.8K	8.1K	8.6K

Table 9: Test Data

orders" are as defined in §3. The column "*Splits*" represents the number of additional shipments due to split orders, which is equal to the number of shipments minus the number of orders.

We implemented the heuristics on the test data and report the results in Table 10. In Ta-

Data Set	Algorithm	Reduced Shipments	(%) of Opt.	(%) of Splits
a	0	6074	100	54.0
	1	5348	88.0	47.4
	2	5869	96.6	52.6
b	0	5836	100	53.3
	1	5184	88.8	47.3
	2	5678	97.3	51.9
c	0	5566	100	54.4
	1	4985	89.6	48.6
	2	5399	97.0	52.7
d	0	4435	100	51.3
	1	3773	85.1	43.6
	2	4370	98.5	50.5

Table 10: Heuristic Results on Test data

ble 10, the rows labeled "*Algorithm 0*" present the optimal solution obtained from CPLEX, the rows labeled "*Algorithm 1*" present the results obtained from the Order Swap procedure, and the rows labeled "*Algorithm 2*" present the results obtained from the combined Order Swap and SKU Exchange procedure. The SKU Exchange procedure considers the single and double shipments in the split orders. The column "*Reduced Shipments*" represents the number of shipments that have been eliminated or saved, relative to the real-time assignments. The column "*(%) of Opt.*" is the reduced shipments in the solution as a percentage of the reduced shipments in the optimal solution. The column "*(%) of Splits*" is the reduced shipments in the solution as a percentage of the additional shipments due to split orders in the real-time assignments.

The running time to find the optimal solution on these instances is approximately 300

seconds on a UNIX machine with 4GB of RAM. The running time of Algorithm 1 is approximately 10 seconds, and the running time of Algorithm 2 is approximately 150 seconds. We extract the necessary data using a text processor Perl. For the SKU exchange heuristic, we solve each transportation problem in CPLEX. After each iteration of both the order swap and SKU exchange heuristic, we update any affected order assignment in Perl. We did not attempt to optimize the running time of the heuristics.

From these tests we see that over 50% of the additional shipments due to splits can be eliminated by solving the re-evaluation problem. Furthermore, we find that the heuristic procedure Order Swap performs quite well by itself, achieving nearly 90% of the reductions in the optimal solution. This result seems to indicate that the online retailer had sufficient inventory when the snapshot was taken but it was not at the right place at the right time. However, we note that the projected benefits from Order Swap might be inflated since we considered the entire inventory data set for the reduced set of orders in the snapshot. We reap additional benefits from implementing the SKU Exchange heuristic procedure after the Order Swap procedure; on these four data sets, the combined heuristic achieves 97% of the optimal solution.

For the SKU Exchange, we sequence the SKUs randomly. To investigate the impact of how SKUs are sequenced, we re-ran the SKU Exchange heuristics with several alternative sequences, e.g., sorting SKUs by the size of their transportation problems (the number of demand nodes or the total amount of supply), sorting SKUs by the amount of uncommitted inventory. We found that the sequence of SKUs had indiscernible impact on the heuristic results.

In the Order Swap results, we sequence the (order, warehouse) pair randomly. We tested the Order Swap heuristic by using different sequences of orders and warehouses. Again, the sequence of orders or warehouses had insignificant effect on the heuristic results.

The Order Swap and SKU Exchange procedure need not reach a local optimum. To obtain a local optimum from the heuristic solutions, we re-ran the SKU Exchange heuristic for the entire list of SKUs repeatedly until there were no further reduction in the number of shipments. For these test problems we found that we reached a local optimum after three or four cycles through the SKU Exchange heuristic. The incremental improvements from the additional runs of the SKU Exchange were insignificant for each test problem; the largest total improvement was a reduction of 6 shipments. Thus, in the subsequent testing, we only consider one cycle each of both the Order Swap and SKU Exchange heuristics.

5.2 On the Entire Data

Having found that the heuristics perform well on the test data compared to the optimal solutions, we implement the procedures on the entire data sets. Table 11 presents the data summary. Each data set of A, B, C, D includes all the not-yet-picked orders from a snapshot,

Data Set	Single orders	Multi orders	Split orders	Splits
A	600K	314K	34K	38K
B	618K	338K	34K	38K
C	624K	338K	35K	38K
D	625K	329K	33K	36K
E	875K	680K	99K	112K

Table 11: Entire Data (Not-Yet-Picked Orders)

which is a randomly chosen day during a five month off-season period in 2004. Data set a in Table 9 is the reduced data set of A in Table 11. The data set E is from a randomly chosen day in the peak season of 2004. We use $T = 12$ time buckets.

We report the performance of the heuristic solution for the entire data sets in Table 12. The columns are defined as in Table 10. Since we do not have the optimal solution for the

Data Set	Algorithm	Shipments	(%) of Splits
A	1	11,028	29.0
	2	15,643	40.9
B	1	13,058	34.2
	2	19,579	51.3
C	1	13,795	35.9
	2	20,074	52.2
D	1	12,937	35.6
	2	19,055	52.4
E	1	37,862	34.0
	2	55,408	49.6

Table 12: Heuristic Results on Entire Data

entire data set, we eliminate the rows that correspond to "*Algorithm 0*". We note that the heuristic reduces the number of extra shipments by 15K to 20K consistently during the off season, which corresponds to 40% to 50% of the total number of extra shipments in the real-time assignments. These numbers are consistent with those in Table 10. Even though we have no optimal solution here to benchmark the heuristic solution, we observe a remarkable

consistency across the data sets and with the test-data solutions. Thus, we might expect these heuristic solutions to be near optimal, as we found with the test data.

The absolute number of reduced shipments is much greater for data set *E*, the peak season data. Again, we find that the heuristic eliminates 50% of the additional shipments due to split orders.

The running time of Algorithm 1 on the off-peak data is approximately 30 seconds and 90 seconds on the peak data. The running time of Algorithm 2 on the off-peak data instances is approximately 800 seconds and 2900 seconds on the peak data. Note that we made no attempts to optimize the running times.

5.3 Summary

As the not-yet-picked queue corresponds to orders for one or two days, we expect that we can re-solve the re-evaluation problem at least every one or two days. From the off-peak season data of *A*, *B*, *C*, *D*, we estimate that we can reduce 15K to 20K shipments from the real-time assignments for each re-evaluation problem. The shipping rate structure for the online retailer is proprietary. To estimate the potential savings, we suppose that we have a net savings of approximately \$1 to \$2 for each shipment we eliminate; we base this approximation on the UPS domestic ground shipping rates for packages of less than five pounds (http://www.ups.com/media/en/2007_daily_rsg.pdf). Thus, the opportunity for cost savings by solving the re-evaluation problem can range from \$2.7 million to \$14.6 million per year.

Our heuristic is relatively easy to implement, as each iteration translates into a series of swaps or exchanges among a limited set of orders. We can feed these exchanges into the online retailer's existing order-management systems.

We conclude that there is an opportunity to reduce the transportation costs for an online retailer by means of a re-evaluation of its real-time fulfillment decisions. We have developed a heuristic to do this re-evaluation and have shown with preliminary testing that it results in better decisions by utilizing more resources and more information.

6. Future Research

One topic for further research is to develop bounds on the performance of the heuristics. One idea is to deploy dual-ascent methods on the network design problem. We can exploit

the special structure of the dual of the LP relaxation to generate lower bounds on the optimal solution. This Dual-ascent method has been proven effective for a number of difficult problems, see Erlenkotter (1978), Wong (1984), Balakrishnan, Magnanti, and Wong (1989), and Raghavan (1994).

By construction, the re-evaluation optimization problem is based on a snapshot of not-yet-picked orders at a random time. We employ effective heuristics to solve the problem. Naturally, we need to solve this problem on a rolling horizon basis. A second topic of future research should address how often to solve the problem. Considering the heuristics developed in this paper, we could solve the simple Order Swap procedure very frequently during a day. For SKU Exchange, we could solve the list of SKUs repeatedly. For each SKU, we could incorporate the new customer orders or inventory arrivals since the last time we solved the SKU's transportation problem. In practice, this frequency may depend on the time needed to retrieve data on the current customer order status.

Recall our heuristics are improvement algorithms based on the initial feasible solution, the real-time assignment. The effectiveness of our heuristics certainly depends on the quality of the real-time assignment. As a third topic for research, one might explore the impact of the real-time assignment on the sub-optimality of the heuristics.

In our re-evaluation problem, we assume that on-order inventory will arrive as planned, without any delay. In practice, the arrival time of on-order inventory is stochastic, and thus some orders may miss their estimated-to-ship dates because of the delay in the arrival of their on-order inventory. In such cases, we might reassign the order so as to avoid violating the estimated-to-ship date, or at least to minimize the extent of the violation. This would entail expanding the formulation of the re-evaluation problem to include the consideration of these "late" orders.

Finally, we take the estimated-to-ship as given in the model. Often an online retailer will quote a time window by which an order will be shipped; furthermore there might be some discretion in how these estimates are set. There would be value from future research that studies how to exploit this flexibility in quoting estimated-to-ship dates so as to further minimize costs.

As of the date of the paper submission, the online retailer has begun implementation of the heuristics as developed in this paper. The initial implementation only includes the SKU Exchange heuristic with only customer order data but no inventory data. The customer order data is imported from the execution system and fulfillment changes are then fed back

to the execution system.

Currently, the heuristic is run regularly with a ten minute interval between consecutive runs. Each run takes more than ten minutes, where the majority of the time is spent on retrieving data and confirming data accuracy. The actual solution time for the SKU Exchange is about 2 to 5 minutes.

Any suggested changes from the heuristic must be implemented by the order fulfillment execution system. In effect for each order to be changed, the execution system must first release the inventory that has been assigned to it, and then make the new assignment, as recommended by the heuristic. But the current execution system can only process one order at a time. As a result, we cannot implement a simple exchange of an SKU between two orders, unless there is a free unit of this SKU at one of the warehouses.

The online retailer is still able to implement between 10% to 50% of the recommended fulfillment changes; this translates into removing several hundred splits per day during the off-peak season. The online retailer intends to address the current restriction that limits its replanning to a single order at a time; this should allow it to then capture a much higher percentage of the recommended changes from the SKU Exchange heuristic. It plans to incorporate inventory data in the future, which will both enhance the SKU Exchange performance and permit the implementation of the Order Swap heuristic.

Acknowledgments

The authors wish to thank the editor, the associate editor and the referees for their very helpful and constructive feedback on earlier versions of the paper. This research has been supported in part by the MIT Leaders for Manufacturing Program, a partnership between MIT and major manufacturing firms; and by the Singapore-MIT Alliance, an engineering education and research collaboration among the National University of Singapore, Nanyang Technological University, and MIT.

References

- Aarts, E., J.K. Lenstra. 1997. *Local Search in Combinatorial Optimization* Wiley, New York.
- Ahuja, R.K., O. Ergun, J.B. Orlin, A.P. Punnen. 2002. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics* **123** 75–102.

- Ahuja R.K., T. Magnanti, J.B. Orlin. 1993. *Network Flows: Theory, algorithms, and applications*. Prentice Hall, Inc.
- Allgor, R., D. Stratila. 2004. Personal Communications.
- Balakrishnan A, T. Magnanti, P. Mirchandani. 1997. Network Design. M. Dell’Amico, F. Maffioli, and S. Martello, eds. *Annotated Bibliographies in Combinatorial Optimization*. John Wiley & Sons.
- Balakrishnan, A., T.L. Magnanti, R.T. Wong. 1989. A dual-ascent procedure for large-scale uncapacitated network design. *Operations Research* **37** 5 716–740.
- Erlenkotter, D. 1978. A dual-based procedure for uncapacitated facility location. *Operations Research* **26** 992–1009.
- Johnson, M. E., S. Whang. 2002. E-Business and Supply Chain Management: An Overview and Framework. *Production and Operations Management* **11** 4, 413-423.
- Magnanti, T.L, R.T. Wong. 1984. Network design and transportation planning: models and algorithms. *Transportation Science* **18** 1–55.
- Minoux, M. 1989. Network synthesis and optimum network design problems: model, solution methods and applications. *Networks* **19** 313–360.
- Newman, M.E.J. 2003. The structure and function of complex networks. *SIAM Review* **45** 2, 167–256.
- Raghavan, S. 1994. Formulations and algorithms for network design problems with connectivity requirements. *PhD Thesis*. MIT.
- Simchi-Levi, D., D. Wu, Z.J. Shen. eds. 2004. *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*. Springer, Inc.
- Swaminathan J., S. Tayur. 2003. Models for Supply Chains in E-Business. *Management Science* **49** 10, 1387-1406.
- Talluri, K.T. 1996. Swapping applications in a daily airline fleet assignment. *Transportation Science* **30** 237–248.
- UPS. 2005. www.ups.com.
- Wong, R. 1984. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming* **84** 271–287.
- Xu, P.J. 2005. Order Fulfillment in Online Retailing: What Goes Where. *PhD Thesis*. MIT.