

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

Ping Josephine Xu • Russell Allgor • Stephen Graves

Operations Research Center, MIT, Cambridge, MA

Amazon.com, Seattle, WA

Sloan School of Management, MIT, Cambridge, MA

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

January 2006

When a customer orders online, an e-tailer 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 replenishments. We examine the benefits from periodically re-evaluating these real-time assignments. We construct near-optimal heuristics for the re-assignment for a large set of customer orders by minimizing the total number of shipments. Finally, we present evidence of significant saving opportunities by testing the heuristics on order data from a major e-tailer.

1. Introduction

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

Large scale: E-tailers have a very large scale operation and catalog.

Logistics as a matter of trust: E-tailer customers consider the timely delivery of products to be a significant component of trust. Thus, the reliability and efficiency of the supply chain is even more crucial.

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

Assemble-to-order system: A major element of the 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 consumed or deployed to meet a customer order. By delaying the decisions on how to fulfill customer orders, e-tailers 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 e-tailer can utilize all of its warehouses or fulfillment centers to serve the customer demand. This centralized demand allocation poses new challenges and opportunities to minimize operating costs.

In this paper, we examine a new order fulfillment problem that is shaped by these characteristics.

When a customer places an order on an e-tailer’s website, the e-tailer, in real time, searches for available fulfillment options from its order fulfillment centers (warehouses) or drop-shippers. The e-tailer assigns the order to one or more warehouses virtually, based on the transportation cost of shipping the order from the warehouse(s) to the customer location, the inventory availability at these warehouses and the customer shipping preferences. The e-tailer then quotes a *estimate-to-ship date* to the customer. If the order has multiple items, then the e-tailer might not be able to ship the order from one location. As a result, it assigns the order to multiple warehouses or drop-shippers and the order is split into multiple shipments. After the order assignment, each item in the order enters the picking queue at its designated warehouse where it often waits for *several hours* before being picked and assembled into a shipment. Each shipment is then given to a third party carrier to deliver the package(s) to the customer location.

We present Example 1.1 to illustrate the real-time assignment decision.

Example 1.1. Suppose a customer located at Chicago orders one unit of a particular CD, as indicated in the dash box in Figure 1. In real time, the e-tailer searches for its available inventory in its warehouses: Warehouse 1 near New York and Warehouse 2 by San Francisco.

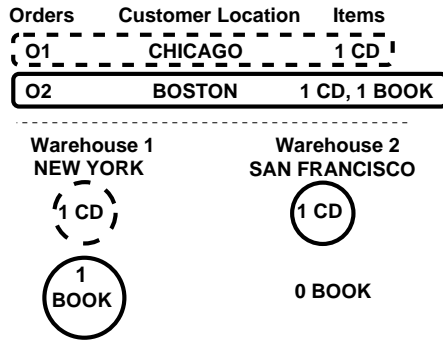


Figure 1: Real-Time Assignments, Three Shipments – Example 1.1.

Both warehouses have one unit of the CD available; the e-tailer chooses the cheaper option to ship the CD from New York, and assigns the CD inventory in Warehouse 1 to O1. Seconds later, a customer from Boston orders a unit of the same CD and a particular book, as in the solid box in Figure 1. When O2 arrives, there is only one fulfillment option and the e-tailer fulfills the second order with two shipments: Warehouse 2 can ship the CD and Warehouse 1 can ship the book to the second customer. We have a total of three shipments for the two orders.

In the transportation cost of shipping a package, the fixed cost component is very significant. As an illustration, we display in Figure 2 the UPS 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,

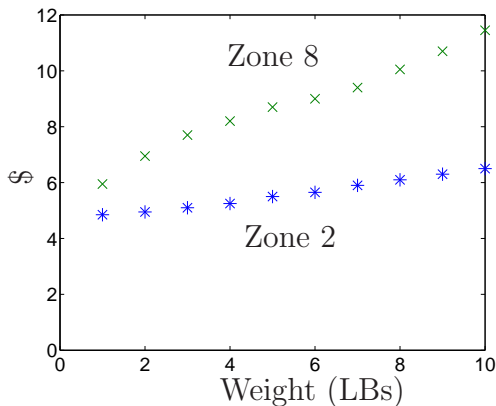


Figure 2: UPS Ground Commercial Rates Within the US Continent

plus a variable cost that is approximately linear in the weight of the package. Furthermore, 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 in the e-tailing setting. 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. Therefore, minimizing the number of shipments is a prime objective for e-tailers.

Revisiting Example 1.1, we see that we can reduce the number of shipments to two, as illustrated in Figure 3 by changing the order-warehouse assignments. We assign the first customer order O1 to Warehouse 2 and the second O2 to Warehouse 1. Example 1.1 is

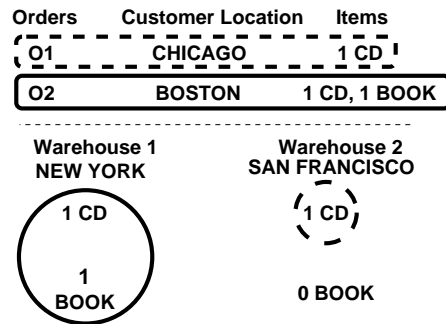


Figure 3: Re-Evaluation Reduces Number of Shipments to Two – Example 1.1.

a bit extreme and the modification to the initial assignments is very straightforward. To appreciate the difficulty and the subtlety of the problem, we discuss Example 1.2 next.

Example 1.2. 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 4. The first customer order is for the book; the e-tailer assigns it to W3, possibly because the

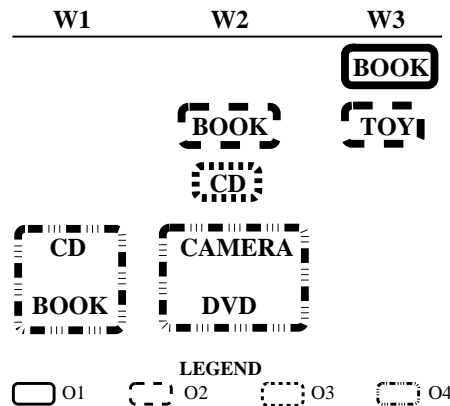


Figure 4: Read-Time Assignments, Six Shipments – Example 1.2.

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 book inventory: O1 took the last unit of book in W3. Hence, the e-tailer assigns the book to W2 and the toy to W3. The e-tailer 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, and it may be unclear how we can shuffle the assignments to reduce the number of shipments. In Figure 5, we reduce the

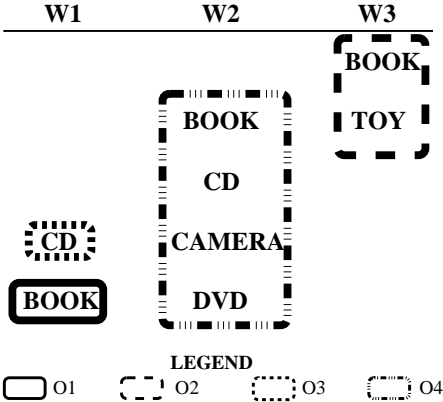


Figure 5: Re-Evaluation Reduces Number of Shipments from Six to Four – Example 1.2.

number of shipments from six to four, which is clearly the best we can do.

In these examples the initial assignment decisions are necessarily myopic because the e-tailer does not anticipate any future customer orders or inventory replenishment. In practice, the real-time assignment is myopic because the e-tailer wants to make a guaranteed service commitment to the customer at the purchase instant. That is, immediately after the customer clicks the purchase button, the e-tailer commits to a estimate-to-ship date, and reserves the necessary inventory so as to deliver on this commitment. The real-time assignment is myopic also because of additional implementation challenges. The large number of customer orders and the need to provide customers with a very quick response in real time make efficient assignment difficult. We conjecture that we can reduce the total transportation cost of shipping orders from warehouses by periodically re-evaluating the real-time assignment decisions, subject to the constraint that the estimate-to-ship date commitment for any customer order is guaranteed.

This shuffling of assignments is also practically feasible. Even when all items in an order are available at the warehouse, the order could wait at least several hours until the order

is released to be picked and sent for shipping. If one or more of the items in the order is not available, then the rest of the order is reserved and waits until the missing items arrive. If the customer opts for delayed or free shipping, then the e-tailer has a longer time window within which to pick and ship the order. Thus, there is an opportunity to revisit and re-evaluate the real-time assignment decisions. We pose a problem to re-evaluate the real-time decisions. We consider the queue of *not-yet-picked* customers orders at an instant in time and their real-time warehouse assignments. We re-evaluate these real-time decisions to reduce the shipping cost without violating the estimate-to-ship date commitments for any of these orders. We take inventory availability and the estimate-to-ship dates as given. We call this optimization of the myopic order assignments the re-evaluation problem.

The re-evaluation problem is difficult theoretically and in practice. Xu (2005) shows that even some of its simple special cases are NP-hard. Furthermore, exact methods cannot solve realistically dimensioned cases. For an off-season snapshot at a large e-tailer, there are 1 million orders with 2 to 3 million units waiting to be picked. There are up to 10 warehouses. The total number of SKUs in those orders ranges from 500,000 to 800,000. In the peak season, the number of orders can reach three or five times of the off-season.

We, therefore, develop efficient and easy-to-implement sub-optimal 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 summarize some computational experiments on sets of real data from a global e-tailer.

2. Problem Formulation

We present a network design formulation of the re-evaluation problem, specially, a fixed-charge multi-commodity flow problem (Allgor and Stratila, 2004). 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 warehouse set
j	index for customer orders, J is the order set
i	index for SKUs, I is the SKU set
d_j^i	units of SKU i in order j
s_k^i	inventory units of SKU i available at warehouse k

The decision variables are:

x_{jk}^i	units of SKU i shipped from warehouse k to customer j
y_{jk}	binary variable to indicate a shipment from k to j

We also denote set K_i as the set of warehouses carrying SKU i inventory, J_i as the set of customer orders containing nonzero units of SKU i .

We denote the following formulation as MIP .

$$\begin{aligned}
\min \quad & \sum_{j,k} y_{jk} \\
\text{s. t.} \quad & \sum_{k \in K_i} x_{jk}^i = d_j^i, \quad \forall i \in I, j \in J_i \\
& \sum_{j \in J_i} x_{jk}^i \leq s_k^i, \quad \forall i \in I, k \in K_i \\
& 0 \leq x_{jk}^i \leq d_j^i y_{jk}, \quad \forall i \in I, j \in J_i, k \in K_i \\
& y_{jk} \in \{0, 1\}, \quad \forall j, k
\end{aligned}$$

Variable x is a continuous variable here because for any given choice of y , we can decompose the problem into a transportation problem by SKU, and there exists an optimal integer solution for each transportation problem. The first constraint assures that the demand is met for each SKU in each order. The second constraint assures that the amount of each SKU shipped from each warehouse does not exceed the supply. Problem MIP has $|J||K|$ binary variables and $|I||J||K|$ continuous variables. It has $|I||K| + |I||J| + |I||J||K|$ 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 by proving that some simple special cases of the problem are NP-hard. The special case of K warehouses and each order having two units is NP-hard. The NP-Complete proof uses the transformation from the Minimum Edge Coloring problem. The special case of two warehouses and each order having three units is also NP-hard. The NP-Complete proof uses the transformation from the Exact Cover by 3SETS (X3C).

2.1 Literature Review

There are two clusters of literature that are most relevant to our problem: network design problems and local search algorithms, a wide class of improvement algorithms.

Magnanti and Wong (1984) provide a survey of models and classic solution methods of the general network design problem up to 1984. They show that the problem is very flexible and contains a number of well known network optimization problems as special cases: minimal spanning trees, shortest paths, vehicle routing problems, facility location problems, etc. Minoux (1989) surveys the models and solution methods of the variants of the general problems. He discusses the general models using minimum cost multicommodity flows, models of tree-like networks, models using nonsimultaneous single-commodity or multicommodity flows. Balakrishnan, Magnanti, and Mirchandani (1997) list annotated bibliographies on network design since 1985. The focus of the survey is on uncapacitated network design, capacitated network design, network loading, and network restoration problems.

The general form of a network design problem is the multicommodity network design problem. Let n be the number of nodes and m be the number of edges in a graph, then we have the following (Balakrishnan et al. 1997):

$$\begin{aligned}
 \min \quad & \sum_{i \in \mathcal{I}} c^i x^i + f y \\
 \text{s.t.} \quad & \mathcal{N} x^i = b^i \quad \forall i \in \mathcal{I} \\
 & x^i \leq k^i y \quad \forall i \in \mathcal{I} \\
 & \sum_{i \in \mathcal{I}} x^i \leq K y \\
 & x^i \geq 0 \quad \forall i \in \mathcal{I} \\
 & y \in Y
 \end{aligned}$$

where \mathcal{I} is a set of commodities, $c^i \in \mathbb{R}^m$ is a vector of edge cost per unit of flow, $f \in \mathbb{R}^m$ is a vector of edge design or installation cost, \mathcal{N} is a node-edge incidence matrix, $b^i \in \mathbb{R}^n$ is a vector of node supplies or demand, k^i is a vector of edge capacity for commodity i , and K is the edge capacity for all commodities. There are two types of decision variables: each element in the vector $x^i \in \mathbb{R}^m$ models the continuous choice of routing commodity i flow on edges, and each element of $y \in \mathbb{R}^m$ models the discrete design choice of installing edges. The total cost is the sum of installation and routing costs.

The first constraint is a flow balance constraint. The second constraint ensures flow is only routed on the installed arcs and edge capacity on each commodity is satisfied. The third constraint imposes the total capacity on each edge for all commodities. Our problem

is a fixed charge uncapacitated multicommodity flow problem, a special case of the general network design problem. Specifically, our model has the following properties: it only has installation costs but no routing costs, it has no bundling capacity constraint on all commodities, the graph topology is bipartite, and each commodity has multiple origins and multiple destinations. The current literature has made the greatest progress in solving uncapacitated problems. Most progress, however, has been made on single-origin single-destination problem, which is the simplest case of the uncapacitated problem. Our model has a simple cost structure and topology and has no side constraints, but has multiple origin and multiple destination.

Many of the special cases of the fixed charge design problem, e.g., the Steiner Tree problem, are known to be difficult to solve or NP-hard in complexity. The fixed charge problem then is also NP-hard (Magnanti and Wong 1984). In addition to the theoretical arguments, substantial empirical evidence also confirms the difficulty of the problem on large-scale instances (Billheimer and Gray 1973, Wong 1985). Balakrishnan, Magnanti, and Wong (1989) develop a dual-ascent algorithm for the fixed-charge network design problem. They test problems with up to 500 integer and 1.98 million continuous variables and constraints. The procedure shows promising results of 1 to 4% of optimality. Holmberg and Hellstrand (1998) show a Lagrangian heuristic within a branch-and-bound framework to find the exact optimal solution. Judging by the size of the instance solved in the current literature, none are close to the scale of our problem.

Neighborhood search is the inspiration of our proposed heuristics, and the neighborhood of our heuristics is exponentially large. 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. Neighborhood search algorithms are a wide class of improvement algorithms that try to improve iteratively by searching the “neighborhood” of the current solution. We can view our heuristics as a type of local search procedure for partitioning problem, which involves transferring or exchanging elements between clusters. In our problem, the clusters are customer orders and the elements are inventory units. We refer readers to Thompson and Orlin (1989) for more discussion of the literature.

We can also view our heuristics as a network flow based improvement algorithm, which uses network flow techniques to identify improving neighborhoods. In our heuristics, by solving a transportation problem, we are able to identify cyclic exchanges of items among

customer orders. Some of these algorithms in the literature characterize improvement moves by negative cost disjoint cycles in a so called “improvement graph”. For example, Thompson and Psaraftis (1993) apply the technique to vehicle routing problems. They try to reduce the total cost of a set of routes by transferring demand among routes cyclically. Their results show the local search method as either comparable to or better than the vehicle routing heuristic algorithms. Ahuja, Orlin and Sharma (2001) apply the technique to the capacitated minimum spanning tree problem. Using variants of shortest path label-correcting algorithms, they are able to identify cycles that exchange nodes among multiple subtrees simultaneously. Their results can improve the best available solution for most of the benchmark instances by as much as 18%. Others effectively apply the search method to minimum makespan parallel machine scheduling problem (Frangioni et al. 2004), and capacitated facility location problem (Ahuja et al. 2004).

An application close in flavor to ours is the paper by Talluri (1996) on daily airline fleet assignment problem. The fleet assignment problem can be modeled as an integer multi-commodity flow problem subject to side constraints and each commodity is a fleet. The problem assigns fleet types to flight legs. After a fleet assignment problem is solved during the planning stage, often the airlines need to change the assignment to accommodate updated demand forecast, disruptions to the schedule, or breakdowns. He develops a swapping procedure to identify and change the fleet types on flights from a given solution. The exchanges are identified by finding negative cost cycles in a related network.

3. Motivation for Heuristics

Customized heuristics are more likely to outperform any general heuristics. To develop any solution procedure tailored to the problem structure, we must examine the problem data carefully. In this section, we summarize the important characteristics of the customer orders and the real-time assignments in the e-tailing setting that motivates this research.

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 e-tailer assigns each order to one or more warehouses and enters each item into picking queue at its designated warehouse. We take a snapshot of all the orders that are waiting to be picked at a time epoch. We present here the summary statistics for the order data sets in the off season from a large e-tailer and one data set from the peak season, as illustrated in Table 1. The off-season data is taken at random days during a period of five months in 2004 and the peak-season is a random day in December, 2004. The term "*Total orders*" represents the total number of

	Off-Peak Season				Peak Season
	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5
Total orders	869K	925K	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%	3.9%	3.7%	3.6%	6.4%

Table 1: Snapshot Data

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 differs 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 18% 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 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

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

single shipment; the row *"sgl or dbl shipment"* represents the percentage of split orders that have at least one single or double shipment. In the off-season data, there is at least one single shipment in more than 80% of the split orders. Over 90% of the split orders have at least one single shipment or one double shipment. These percentages are smaller but still significant in the peak season. This observation is one motivation for our heuristics; as will be seen, we exploit the abundance of the single and double shipments in our heuristics.

To investigate whether the problem can be decomposed into a number of smaller problems, we examined the connectivity of the order-SKU graph constructed for the snapshot of the not-yet-picked queue. 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. For these data sets, we find that there exists one very large component in the graph, containing the majority of the SKU’s. Furthermore, any removal of small subsets of SKU’s does not change the connectivity of the graph. Therefore, we do not see a clear way to decompose the problem by considering a limited number of orders or SKUs.

Our exploration of the order-SKU network in the e-tailing setting echoes the theory of random graphs. Solomonoff and Rapoport (1951) and independently Erdős and Rényi (1959) study a very simple model of network called Poisson random graphs. In a graph with n nodes, a random edge is independently chosen with probability of p to connect any two nodes in the graph. They show that in the limit of a large n , the degrees of the nodes in the graph is Poisson distributed. They also demonstrate that when the value of p is high, a large percentage of the nodes are joined together in a single *giant component*. Recently, Newman (2003) reviews the development in complex networks. The empirical studies of networks, such as the Internet, social networks, and biological networks, show some common properties among the different complex networks. In particular, as reproduced by the random graphs, the network exhibits the “small-world effect”, or most pairs of nodes in most real-world networks seem to be connected by a short path through the network. Our order-SKU

network is not as simple as the Poisson random graphs: our edges are correlated. Our network, however, seems to demonstrate the small-world effect.

4. Heuristic Approach

We start with an initial feasible solution, namely the real-time assignments and develop an improvement algorithm, by which we iteratively create better feasible solutions. The focus on an improvement algorithm is also driven by two practical concerns: **i)** Improvement algorithms generate a feasible solution at each iteration. After each iteration, we can implement the recommended changes to the incumbent assignments to get an improved order assignment. This facilitates implementation, since we always have a feasible solution, even if there were a sudden termination of the algorithm. **ii)** Retrieving data in the setting of a large e-tailer is time consuming, because of the quantity of retrieved data and that the data needs to reflect all changes to the state of orders and inventories. We consider a local search algorithm which searches 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. Therefore, we can improve overall running time by solving the problem and retrieving data in parallel.

One key idea of the heuristics is using the single orders to “fix” the split orders. The motivation is twofold. First, single orders consists of a single shipment and therefore are very flexible in their assignment. The special case of our network design problem in which all orders are single orders is an easy transportation problem. Second, the vast majority of split orders in the real-time assignment include a single shipment. By re-assigning a single order from warehouse A to warehouse B, we free up a unit of inventory at warehouse A that might be used to avoid a split order. Our Example 1.1 illustrates such an instance.

Our heuristics consist of two distinct parts. We name the first part as **Order Swap** as we consider split orders one at a time and examine possible swaps. The second part is **SKU Exchange** as we consider one SKU at a time and examine possible cyclic exchanges. In our implementation, we 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.

For 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 infinitely long estimate-to-ship date. This will simplify the presentation. We also assume that all on-order inventory arrives to the warehouse at the due date promised by the supplier.

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. Let j be the index for split orders, and let k be the index for warehouses. For each 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. Figure 6 specifies the general implementation. We illustrate the Order Swap heuristic with the following example.

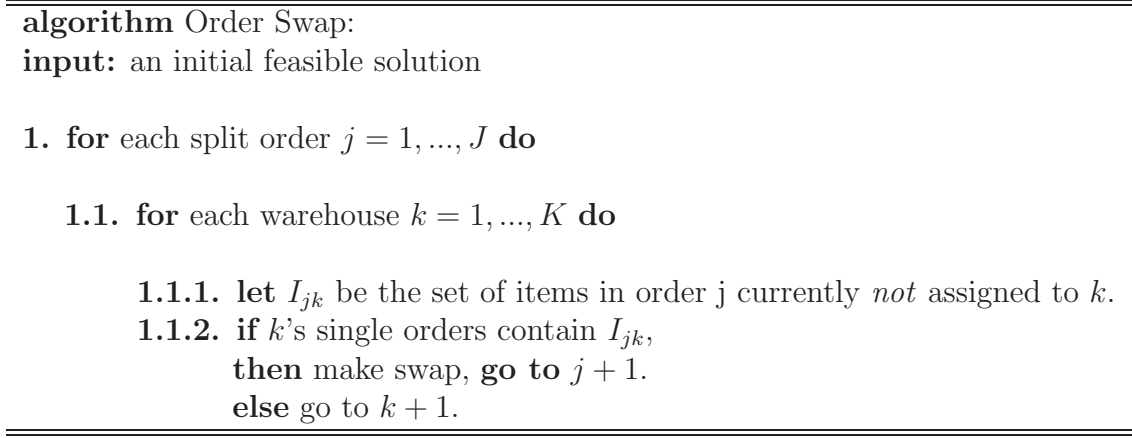


Figure 6: Order Swap Algorithm

Example 4.1. We perform the Order Swap procedure on O1 as illustrated in Figure 7. O1

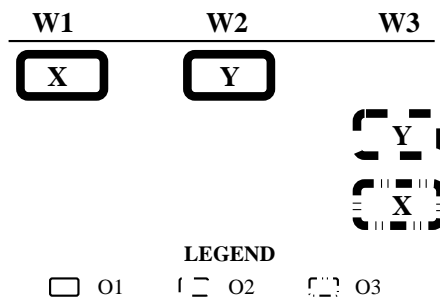


Figure 7: Order Swap Example 4.1.

requires one unit of both SKU X and Y and is currently a split order, with X assigned to warehouse 1 and Y to warehouse 2. O2 and O3 are the single orders at warehouse 3. There are no single orders at warehouse 1 or 2, so we consider warehouse 3 next. The entire order of O1 is currently not assigned to warehouse 3: $I_{13} = \{XY\}$. The set of single orders at warehouse 3 contains $\{XY\}$ and we can make the swap. The assignments after the swap are illustrated in Figure 8, resulting in a reduction of one shipment.

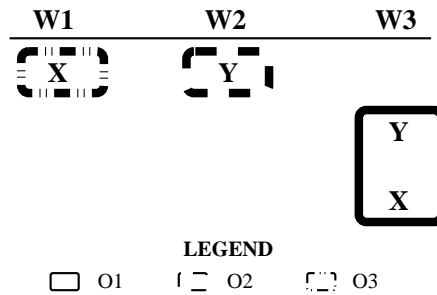


Figure 8: Order Swap Example 4.1– After a Swap.

4.1.1 Order Swap with time dimension

Different orders can have different estimate-to-ship dates; furthermore, the unassigned inventory can be either on hand or on order. Thus, the actual problem we solve has a time dimension. To explain how we incorporate this time dimension, we denote

- v_i estimate-to-ship date of item i in the real-time assignment,
- u_i time at which the inventory committed to item i arrives to the warehouse.

By assumption we have $u_i \leq v_i$ as this is required for feasibility. To ensure that we retain a feasible solution, we cannot violate the estimate-to-ship dates. Thus, we add the following constraint in Order Swap: swapping item i with j of the same SKU is feasible iff $u_i \leq v_j$ and $u_j \leq v_i$. That is, a swap is feasible if the e-tailer can still ship both items within their estimate-to-ship dates after the swap.

4.2 SKU Exchange

The second key idea of the heuristics is to consider one SKU at a time. This exploits the special case of the general re-evaluation problem, where all orders are single orders. This special case can be formulated as a transportation problem for which polynomial algorithms can solve those problems optimally. In SKU Exchange, we consider each SKU and solve a

transportation problem that attempts to reduce the number of split orders. After solving each transportation problem, we update 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). Figure 9 is a general implementation.

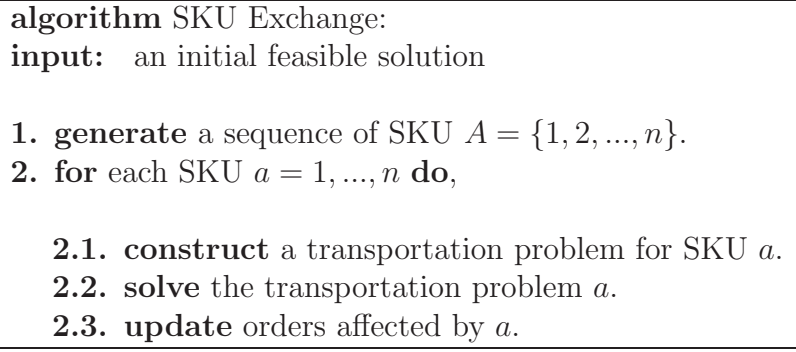


Figure 9: SKU Exchange Algorithm

4.2.1 SKU Exchange of single shipments

We only consider a SKU if it has single orders or uncommitted inventory, and split orders with single shipments of the SKU. We start with the following example to illustrate the transportation problem before we summarize the details.

Example 4.2. We consider the three orders with the real-time assignment listed in Figure 10. We consider SKU Y, as it has a single order O2 and has single shipments in two multi orders,

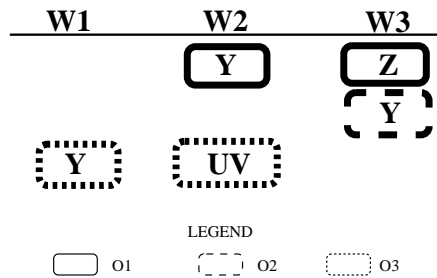


Figure 10: Real-Time Assignments – Example 4.2

O1 and O3. We construct the corresponding maximization transportation problem for SKU Y in Figure 11. Each warehouse represents a supply node, and each order with a single shipment of SKU Y represents a demand node. From Figure 11, we see that the supply at each supply node is the available units of SKU Y at the warehouse for re-assignment, and

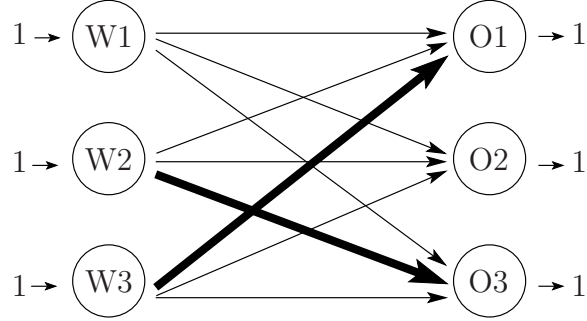


Figure 11: Transportation Problem for SKU Y – Example 4.2

the demand at each demand node is the number of units of SKU Y in the order. A unit of flow from supply node k to demand node j signifies that warehouse k ships a unit of SKU Y to fill order j 's requirement.

Definition 4.1. Let the set of *profitable warehouses* of SKU y in order j be $P_j(y)$ such that, $\forall k \in P_j(y)$, while maintaining other shipments, shipping the SKU y from warehouse k reduces a split in order j .

That is, there will be one less shipment if warehouse k supplies the SKU Y for order j . The arc profit for arc (k, j) , $\forall k \in P_j(Y)$ is 1, signifying that a unit flow on this arc results in one less shipment. The arc cost is zero for all other arcs. In Figure 11, $P_1(Y) = \{3\}$, $P_2(Y) = \emptyset$, $P_3(Y) = \{2\}$, and only arcs $(2, 3)$ and $(3, 1)$ (the dark arcs) have a profit of 1.

By inspection, we see that the optimal solution is to send one unit of flow along arcs, $(1, 2)$, $(2, 3)$ and $(3, 1)$. The optimal solution corresponds to the results in Figure 12. We

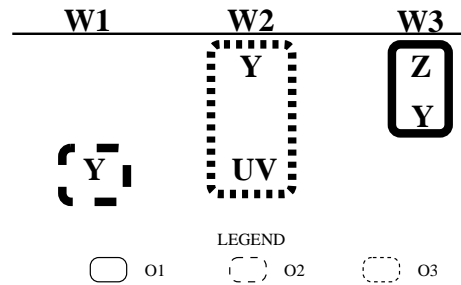


Figure 12: Re-Evaluation Reduces No. of Shipments from 5 to 3 – Example 4.2

reduce the number of shipments in the three orders from 5 to 3.

Figure 13 is an augmenting cycle with respect to the initial solution: $O3 \rightarrow W1 \rightarrow O2 \rightarrow W3 \rightarrow O1 \rightarrow W2 \rightarrow O3$. Starting from the initial solution, the augmentation increases one unit of flow on the forward arcs (warehouse to order) and decrease one unit of flow on the

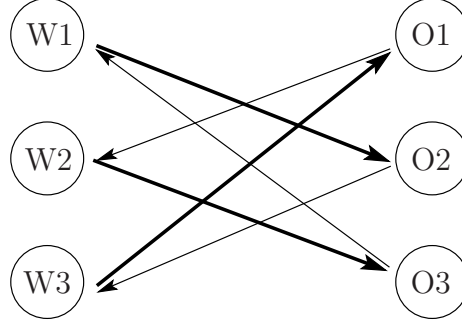


Figure 13: Augmenting Cycle – Example 4.2

backward (order to warehouse) arcs. After we augment the cycle from the initial solution, we reach the optimal solution of the transportation problem of Y . We can interpret the augmenting cycle in the context of the problem: a backward arc (j, k) represents un-assigning the inventory unit of Y in warehouse k that's currently assigned to order j ; a forward arc (k, j) represents assigning an inventory unit of Y in warehouse k to order j . We now can see the cyclic exchanges that are required to implement the solution: we un-assign the unit of Y at $W1$ to $O3$, and re-assign it to $O2$; we un-assign the inventory from $W3$ to $O2$, and re-assign it to $O1$; we un-assign the inventory from $W2$ to $O1$, and re-assign it to $O3$. That is, by implementing the cyclic exchange of SKU Y according to $O3 \rightarrow O2 \rightarrow O1 \rightarrow O3$, we arrive at the solution in Figure 12. The term *SKU Exchange* is named in view of the one or more cyclic exchanges in each transportation problem.

One practical advantage of the heuristic is the ability to solve and implement one cyclic exchange at a time while maintaining a feasible solution at each iteration. While the major objective is to minimize the number of shipments, we also want to do so without implementing any unnecessary exchanges. That is, we have a secondary objective of minimizing the number of exchanges from the initial real-time solution. This objective is easy to add on to the current transportation problem by modifying some arc costs. As denoted before, let j be a demand node and k be a supply node. Arc (k, j) has an additional profit of $n\epsilon$ if there are n committed inventory units currently in warehouse k and whose associated demand node is j . The value of ϵ is specified as $0 < \epsilon < 1$. To ensure the objective of minimizing number of exchanges is a secondary objective, we set the value of ϵ as $\epsilon < \frac{1}{m}$, where $m = \sum_{k=1}^K s_k$ is the total number of supply units in the transportation problem.

In some instances, we might want to minimize the transportation costs, instead of the number of shipments. Recall we approximate the transportation costs by the number of ship-

ments, because the items we consider tend to have small weight and thus have a significant fixed cost in the shipment. We can easily extend the transportation problem to incorporate actual transportation costs. Let's illustrate the cost modification with a previous example, Example 4.2. Consider arc (k, j) , let $c_j(k)$ be the transportation cost of shipping a unit of SKU Y from warehouse k in order j . For example, if $j = 1$, then $c_1(3)$ is the transportation of shipping one package containing one unit of SKU Y and Z from warehouse 3. Let warehouse k_j be the current location of SKU Y in order j . Then, we set the profit of arc (k, j) as $c_{kj} = c_j(k_j) - c_j(k)$.

4.2.2 SKU Exchange with double shipments

In the real data sets 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. To include more orders in the SKU Exchange heuristic, we can also incorporate all orders with double shipments. We illustrate this extension by means of the following example.

Example 4.3. We consider the orders with real-time assignment in Figure 14.

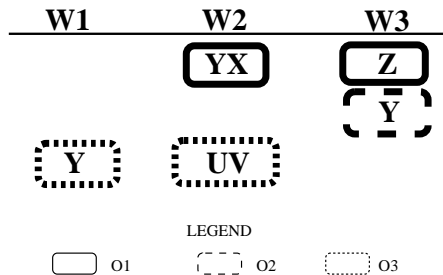


Figure 14: Real-Time Assignments – Example 4.3

Here the only change from Figure 10 is that the first order has a double shipment of YX instead of a single shipment of Y . Similarly, we implement a SKU Y transportation problem for the batch of orders, which all have a single or double shipment of Y . The resulting transportation problem, as shown in Figure 15, resembles Figure 11. Again, the darkness of the arcs indicates the relative amount of arc profit. The only difference is the arc cost of arc $(3, 1)$. In Figure 11, one unit of flow on arc $(3, 1)$ indicates that we supply one unit of SKU Y in warehouse 3 to order 1, which reduces one split. However, in the current example, a unit of flow on the arc does not reduce one split. The split can be reduced only if the unit of

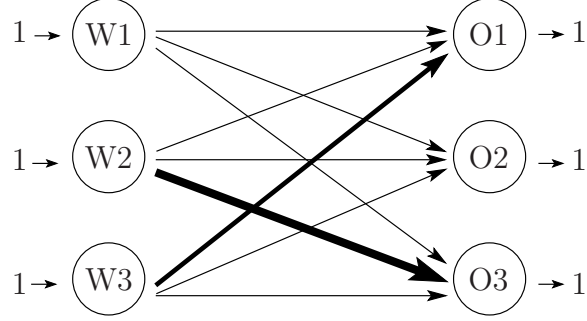


Figure 15: Transportation Problem for SKU Y – Example 4.3

SKU X can also be supplied from warehouse 3. Therefore, in our current example, we set the arc cost of $(3, 1)$ to be, p_{31} , an estimate of the probability that SKU X can also be shipped from warehouse 3. It is not clear how exactly to estimate this probability, but we conjecture that we do not need a precise estimate. Rather the intent is to set the 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. For our computational work, we set $p_{31} = 0.5$.

4.2.3 SKU Exchange with time dimension

The actual transportation problem that we need to solve is a bit more complex due to the need to account for the time dimension. We outline the general idea in this section. 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 and $t=T$ representing any commitments beyond the T day horizon.

Figure 16 is a full-scale representation of a transportation problem for one SKU.

Supply: We group the supply nodes into T supply blocks, and have a supply node for each warehouse for each time period. The supply node of warehouse k in time block t has a supply of s_{kt} . The supply available at all warehouses for the snapshot time block, $s_0 = \sum_{k=1}^K s_{k0}$, reflects the on-hand inventory, whereas the supply for future time blocks, $s_t = \sum_{k=1}^K s_{kt}$, $t > 0$, is the on-order inventory that will arrive during the time block.

Demand: We group the demand nodes into T demand blocks, one for each estimate-to-ship date category. Demand block t has a single-order node S_t with d_{st} number of single

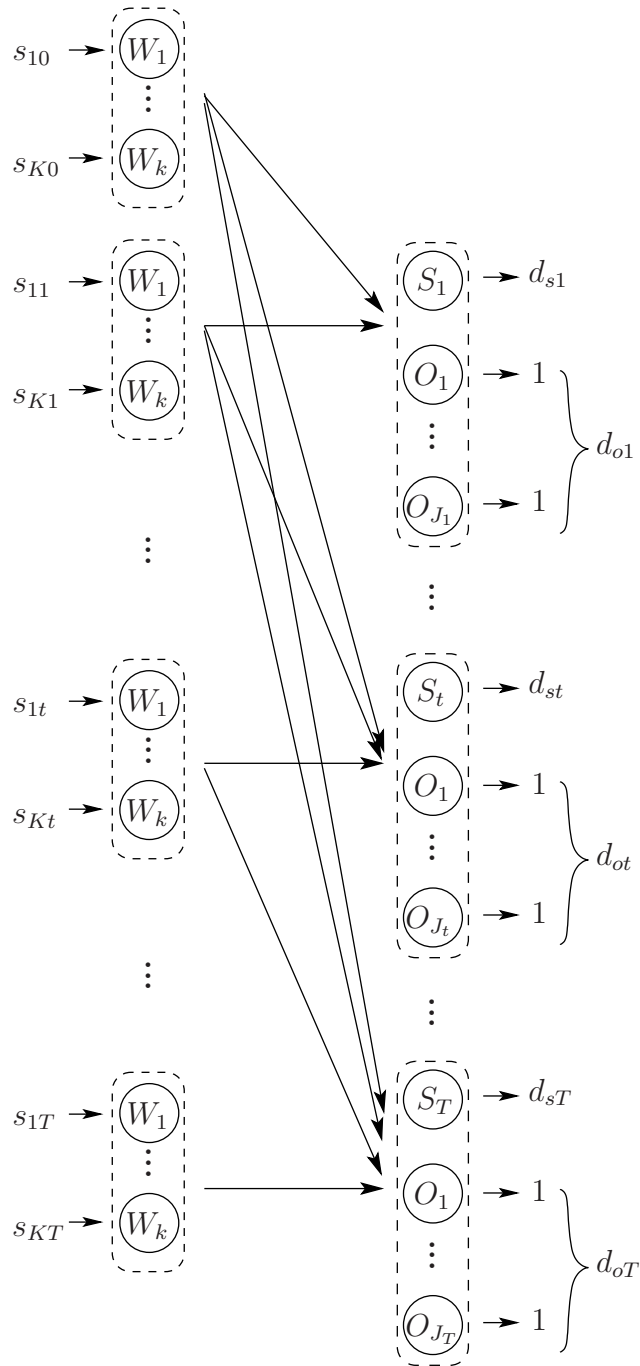


Figure 16: Transportation Problem of One SKU (with Simplified Arcs)

orders whose estimate-to-ship date is t . Each admissible shipment of the SKU with estimate-to-ship date t has a node in demand block t .

Arcs: We permit arcs from the nodes with nonzero supply in supply block t_2 to the demand nodes in demand block t_1 , $\forall t_1 \geq t_2$.

To implement the proposed heuristics, we need to extract the cyclic exchanges from the transportation problem. Xu (2005) shows with examples that solving a transportation problem in SKU Exchange is preferred than solving an assignment problem. She also demonstrates the procedure to extract the cyclic exchanges.

Xu (2005) shows that both Order Swap and SKU Exchange heuristics can perform arbitrarily bad. For Order Swap, the worst cases occur when single orders are scarce and the orders are large in size. Since the majority of the split orders have two shipments in the real-time assignment and there is ample amount of single orders and uncommitted inventory in practice, Order Swap performs well in practice. For SKU Exchange, the worst cases occur when each warehouse assigns to an order never carries all of 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 other than it currently occupies. On the other hand, SKU Exchange relies much less on the abundance of single orders or uncommitted inventory than Order Swap.

5. Computations

We implement the heuristics on several real data sets from a global e-tailer. To examine the sub-optimality of the heuristics on the data, we would like to benchmark the heuristic solutions with 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 extract a much smaller test data set to benchmark 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 all not-yet-picked orders will typically include several days worth of orders that were placed before the snapshot time. To create smaller data set (e.g., set a from

set A), we extract a subset of 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 estimate-to-ship date and all inventory is available by that date..

Table 3 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	Split
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 3: Test Data

orders" are as defined in §3. The column "*Split*" 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 4. In

Data Set	Algorithms	Shipments	(%) of Opt.	(%) of Splits	Time (sec)
a	0	6074	100	54.0	300
	1	5466	89.9	48.5	12
	2	5862	96.5	52.0	188
b	0	5836	100	53.3	300
	1	5175	88.6	47.3	12
	2	5669	97.1	51.8	150
d	0	5566	100	54.4	300
	1	4984	89.5	48.9	8
	2	5407	97.1	52.8	122
d	0	4435	100	51.3	300
	1	3974	89.6	46.0	8
	2	4371	98.6	50.6	162

Table 4: Heuristic Results on Test data

Table (4), *Algorithm 0* is the optimal solution from Cplex, *Algorithm 1* is the Order Swap procedure, and *Algorithm 2* is the combined Order Swap and SKU Exchange procedure. The column "*Shipments*" represents the number of shipments reduced from 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 of the optimal on these instances is approximately 300 seconds. We implement the heuristics on UNIX machines with 1.5 GHz processors and 1GB RAM. We extract the necessary data using a text processor Perl. For each SKU in the sequence, we solve each transportation problem in Cplex. We then update the affected orders back in Perl. Overall, the Order Swap runs within seconds, and the combined procedure of Order Swap and SKU Exchange terminates within a few minutes. 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. 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 reran the SKU Exchange heuristics with several alternative sequences, e.g., sorting SKUs by the size of their transportation problems (size of nodes or supplies), sorting SKUs by the amount of uncommitted inventory. We find that the sequence of SKUs has no discernable impact on the heuristic results.

In the Order Swap results, we sequence the (order, warehouse) pair randomly. We tested the heuristics by using different sequences of orders and warehouses. Again, the sequence of orders or warehouses did not affect the heuristic results significantly.

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 5 presents the data summary. Each data set of A, B, C, D includes all the not-yet-picked orders from a snapshot, which is a randomly chosen day during a five month off-season period in 2004. The data set E is from a randomly chosen day in the peak season of 2004. We use $T = 12$ time buckets.

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 5: Entire Data (Not-Yet-Picked Orders)

We report the performance of the heuristic solution for the entire data sets in Table 6. The columns are defined as in Table 4. Since we do not have the optimal solution for the

Data Set	Algorithms	Shipments	(%) of Splits	Time (sec)
A	1	11,028	29.0	43
	2	15,643	40.9	732
B	1	13,058	34.2	32
	2	19,579	51.3	893
C	1	13,795	35.9	28
	2	20,074	52.2	820
D	1	12,937	35.6	26
	2	19,055	52.4	862
E	1	37,862	34.0	89
	2	55,408	49.6	2931

Table 6: 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 4. 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.

We report the running times for the heuristics in the table, but 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. Suppose that we save approximately \$1 to \$2 for each shipment we eliminate. Thus, the opportunity for cost reduction by solving the reduced problems can range from \$1.4 million to \$4.2 million per year, and solving the re-evaluation problems 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 cyclic exchanges among a limited set of orders. We can feed these exchanges into the e-tailer’s existing order-management systems, and as such, are optimistic that implementation is possible.

We conclude that there is an opportunity to reduce the transportation costs for an e-tailer 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

In this paper, we generate near-optimal heuristics to reduce the number of shipments required to satisfy a set of multi orders from a set of warehouses. We show that there is a significant cost saving by exploiting the order-to-delivery window.

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 methods have been proven effective for a number of difficult problems. Erlenkotter (1978) uses a dual-ascent procedure for uncapacitated facility location problem. Wong (1984) uses the method for the Steiner tree problem. Balakrishnan, Magnanti, and Wong (1989) use it for a large-scale uncapacitated network design problem where each commodity has a single origin and destination. Their results are guaranteed to be within 1 to 4% of optimality. Raghavan (1994) studies the procedure on network design problem with connectivity requirements. His algorithm also solves the special cases of the k -edge-disjoint path and k -node-disjoint path problems optimally. The solutions

from the dual-ascent procedure are within 4% of the optimal for typical telecommunication applications and within 1% of the optimality for Steiner tree problems. Future research needs to explore the dual-ascent method on our model as well.

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 chapter, we could solve the less time consuming and simple Order Swap procedure very frequently during a day, and solve the more time consuming SKU Exchange procedure daily. In practice, this frequency may depend on the data retrieving time.

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, we should 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 that's assigned to a customer order would arrive before the order's estimate-to-ship date. In practice, the arrival time of on-order inventory is stochastic, and thus some orders may miss their estimate-to-ship dates because of the delay in their on-order inventory arrival. We should address those "late" orders in solving the re-evaluation problem.

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

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.

- Ahuja, R.K., J.B. Orlin, S. Pallottino, M.P. Scaparra and M.G. Scutella. 2004. A multi-exchange heuristic for the single-source capacitated facility location problem. *Management Science* **50** 6, 749–760.
- Ahuja, R.K., J.B. Orlin, D. Sharma. 2001. New neighborhood search structures for the capacitated minimum spanning tree problem. *Mathematical Programming* **91** 71–97.
- 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.
- Billheimer, J., P. Gray. 1973. Network design with fixed and variable cost elements. *Transportation Science* **7** 49–74.
- Erdős P., A. Rényi. 1959. On random graphs. *Publicationes Mathematicae* **6** 290–297.
- Erlenkotter, D. 1978. A dual-based procedure for uncapacitated facility location. *Operations Research* **26** 992–1009.
- Frangioni A, E. Necciari, M.G. Scutella. 2004. A multi-exchange neighborhood for minimum makespan parallel machine scheduling problems. *Journal of Combinatorial Optimization* **8** 195–220.
- Garey, M.R., D.S. Johnson. 1979. *Computers and Intractability, a guide to the theory of NP-Completeness*. W. H. Freeman and Company.
- Holmberg, K., J. Hellstrand. 1998. Solving the Uncapacitated network design problem by a Lagrangian heuristic and branch-and-Bound. *Operations Research* **46** 2, 247–259.
- Holyer, I. 1981. The NP-Completeness of edge-colouring. *SIAM J. Comput* **10** 4, 718–720.
- 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*. Massachusetts Institute of Technology.
- Solomonoff, R, A. Rapoport. 1951. Connectivity of random nets. *Bulletin and Mathematical Biophysics* **13** 107–117.
- Talluri, K.T. 1996. Swapping applications in a daily airline fleet assignment. *Transportation Science* **30** 237–248.
- Thompson P.M., J.B. Orlin. 1989. The theory of cyclic transfers. *Working Paper*. Operations Research Center, MIT, Cambridge MA.
- Thompson, P.M., H.N. Psaraftis. 1993. Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations Research* **41** 5, 935–946.
- UPS. 2005. www.ups.com.
- Wong, R.T. 1985. Probabilistic analysis of an optimal network problem heuristic. *Networks* **15** 347–363.
- 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*. Massachusetts Institute of Technology.