Research Gallery: Ecommerce Order Fulfillment Optimization at IBM Research

Xuan Liu, Ajay Deshpande, Ali Koc, Brian Quanz, Dahai Xing, Lei Cao, Yingjie Li

1. Research Overview

Dr. Xuan and her team at IBM Research strive to solve challenging ecommerce analytics and optimization problems. The team have been working closely with external business customers, and have deployed multiple production solutions for ecommerce retailers.

2. InBalance Order Fulfillment Optimization

InBalance offers two solutions: 1) The Fulfillment Event Planner provides rapid what-if analysis to plan fulfillment network, labor and sourcing rules; 2) The Order Fulfillment Optimizer optimizes the sourcing of live orders while balancing shipping and labor costs, delivery times, and markdown savings.

Omni-channel retailers have a huge potential advantage -- their store network. Winners in the market will figure out how to turn their network into a huge advantage. However, effectively using the store network in an omni-channel world is a huge technological and analytical challenge - since typical supply chain modeling tools fail when the network grows from 10's of nodes to 1000's of nodes. Questions like "How many stores should I use to get to my customer faster? Which stores? What inventory should I have in my stores to meet omni-channel demand? How should I use my store network for peak?" -- are all deep analytical questions which need the use of big-data and advanced modeling capabilities -- not only in an off-line planning mode, but also in an online optimal execution mode. InBalance solution uses break-through "Continuous Big Data Simulation Analytics and Optimization Solutions" which applies advanced analytics/optimization and simulation to data from e-commerce, supply chain, stores and marketing. InBalance -- a play on Inventory and Balance (e.g. balancing cost vs speed)-- both terms that are very often used by e-commerce, supply chain and store professions -- are the keys to omni-channel success.

InBalance Fulfillment Event Planner: This is a big-data simulator which uses real POS data, Inventory data and e-Com data, to provide “what if simulation capabilities” in support of omni-channel fulfillment network design by setting up the big Knobs - targeting questions like

•How many stores and DCs should I use for network fulfillment for peak season?
•What products should be eligible for fulfillment from which nodes?
•What rules should be used for node selection (closest store, DC)?
•What is the impact of network fulfilment on transportation cost, markdown?

InBalance Order Fulfillment Optimizer: This is an order level sourcing optimization

solution which uses predictive analysis and math programming technologies for multi-objective optimization using end-to-end cost to serve models. This optimizes both batch and single orders to enable retailers to balance across many different conflicting needs. This is an in-line execution engine to support real-time order fulfillment.
•Optimize transportation cost vs speed to customer
•Optimize transportation cost vs markdown reduction
•Optimize transportation cost vs node capacity utilization

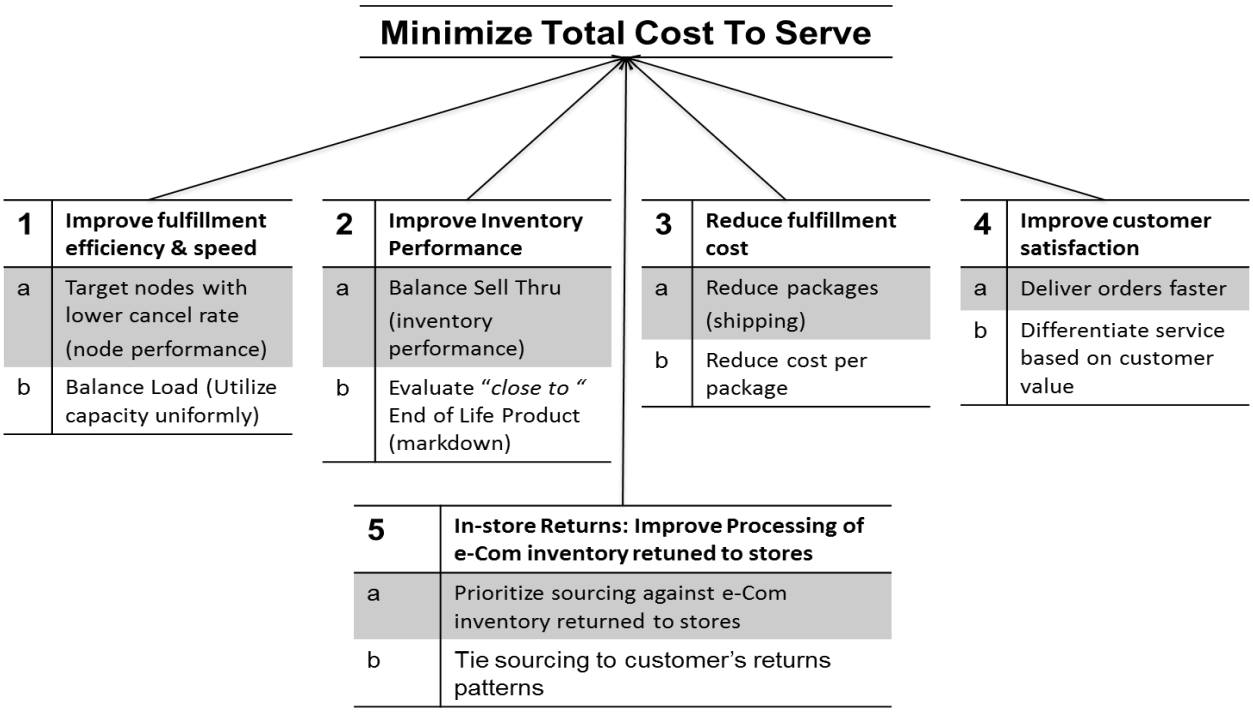

Figure 2: The total-cost-to-serve order fulfillment optimization objective

In our optimization model, we address several dimensions of the omni-channel order fulfillment optimization problem, including the ones mentioned above. We assign a set of items in an e-com order to (sourced from) the nodes (brick-and-mortar store, e-fulfilment centers, etc) of a retail channel. We optimize for various dimensions: cheapest-cost carrier availability at the nodes, package weight (total weight of the items assigned to the nodes), capacity utilization of the nodes, item inventories at the nodes, markdown and order cancellation risks at the nodes.

Fulfillment sourcing logic: The crucial component is the ability to optimize order sourcing logic. Order sourcing can be optimized based on transportation mode, carrier and service level, store labor capacity and cost. There are many options (nodes of the network) for fulfillment, including brick-and-mortar stores, vendors, distribution centers, third-party logistics providers, or in-store pickup. At the order level, retailers must be able to manage multiple ship-to locations and select different carriers based on service level, origin and destination. Retailers must also be able to upgrade or downgrade packages based on promised dates.

Backlog-aware shipping: A good order fulfillment system should work properly at all different conditions – such as the peak days with huge amount of demand that potentially will introduce large number of backlogged orders and eventually cause massive order cancellation. To solve this problem, we design our backlog-aware shipping approach. It first predicates the potential backlogs happening on each node in the fulfill network by analyzing the historical data at the beginning of each day. Then such predicated backlog information will be feed into the optimizer at the real time and considered as one variable when solving the shipping cost minimization problem for each incoming order. This approach is shown to be able to deliver the packages on time whenever possible, while still minimizing the shipping cost.

Load Balancing: Balancing the work load across the nodes in the fulfillment network not only is able to mitigate backlogs, but potentially also avoid markdown. Our load balancing approach achieves this goal by assigning a load balancing cost to each node. Our load balancing cost model handles nodes at different status with different strategies and dynamically adapts to the best strategy for each node based on its real time snapshot of work load. Integrating the load balancing cost into the fulfillment optimizer this mechanism is able to not only balance work load, but also reduce the shipping cost.

3. On determining when and how to bypass the fulfillment engine: Learning fulfillment engine behavior.

Full description (for a shorter description you could take just the first 2 paragraphs):
In recent years, omni-channel retailers are utilizing larger quantities and varieties of nodes in their supply networks for fulfillment of e-commerce orders with the goal of reducing fulfillment cost, in particular their large store networks. This raises new challenges as there are many more such nodes e.g., hundreds vs. 3-4, and further these were not designed for fulfilling e-commerce orders. As a result many more factors and objectives must also be taken into account in addition to just total shipping cost, such as inventory availability, clearance related savings, node performance, etc.

With the increase of order fulfillment options and business objectives taken into consideration in the deciding process, order fulfillment deciding - i.e., deciding which nodes to assign which parts of an order to - is becoming more and more complex. With increasing complexity, efficiency of the deciding process becomes a real concern. Finding the optimal fulfillment assignments among all possible ones may be too costly to do for every order, especially during peak seasons and times. Furthermore retailers do not want to incur the cost for increased resources to adequately handle the peak load as it is only needed during the short peak seasons. With the much larger fulfillment networks enabled, retailers' existing fulfillment deciding engines may not be able to adequately handle the increased load during peak season times and are forced to either revert to sub-optimal fulfillment assignments that result in higher fulfillment cost, or delay and cancel orders, severely damaging customer relationships.

Figure 3: Left: general short-cutting process- the dotted line shows the potential short-cut path taken; Right: order split deciding case.

In our research, we are exploring and developing new approaches by which the load can be reduced on the fulfillment deciding engine, essentially by shortcutting the fulfillment deciding process to avoid the complex, time-consuming full optimization problem of determining the best of all possible assignments. We introduce a novel shortcutting component in the fulfillment deciding process that analyzes an order before it is passed to the deciding engine. The idea is that by utilizing and learning from the past order fulfillment decisions we can find cases where the fulfillment decisions for future orders will likely follow similar patterns, and thereby greatly simplify the deciding process in those cases – e.g., by having to consider only a much smaller set of possible assignments. Cases where the full fulfillment deciding process is likely unnecessary are accurately identified and routed to bypass the heavy-weight deciding process. With this approach, we aim to enable retailers to handle both peak and non-peak season loads with less resources, and also reduce unnecessary utilization of the resources they do have, freeing them to be used for other tasks. Additionally companies can also see what patterns there are in their order fulfillment, better understand how their orders are getting fulfilled, and make other supply chain changes and decisions based on this as well.

In general, various types of shortcutting are possible. There are three shortcutting use cases in particular that motivate our initial research. The first is determining / predicting when an order will be split. Examination of a sample of peak season data from one retailer revealed that the majority of orders were not split in the end by the fulfillment deciding engine - around 75%. Identifying even a portion of these orders fairly accurately would allow significantly reducing the burden on the fulfillment deciding engine, since if we can say in advance an order doesn't need to be split we can simply check the total cost of shipping all order lines at each node – massively reducing the number of assignment options under consideration and greatly simplifying the solving process (e.g., linear scan of the nodes). The second use case is repeated identical orders for ``hot'' items. If an order arrives that is very similar to one that was recently decided, we can in many cases safely assign the order directly to the same nodes with minimal checking, thus bypassing the full-blown deciding process. This case arises quite commonly especially during peak seasons where hot or sales items are ordered at a high frequency. The third use case is short-listing - i.e. in some cases it may be possible to predict with high probability a reduced set of nodes for which the fulfillment engine will end up assigning the order lines among, even for split orders. In this case we may find patterns of key features that let us greatly reduce the set of nodes under consideration.