Title
Data-Driven Decision Making Algorithms For Internet Platforms

Author
Jehl, Titouan

Publication Date
2020

Peer reviewed|Thesis/dissertation

Data-Driven Decision Making Algorithms For Internet Platforms

by

Titouan Jehl

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Industrial Engineering and Operations Research

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Max Z. Shen, Chair
Professor Terry Taylor
Professor Philip M. Kaminsky
Associate Professor Paul Grigas

Spring 2020

Data-Driven Decision Making Algorithms For Internet Platforms

Abstract

Data-Driven Decision Making Algorithms For Internet Platforms

by

Titouan Jehl

Doctor of Philosophy in Engineering - Industrial Engineering and Operations Research

University of California, Berkeley

Professor Max Z. Shen, Chair

Internet platforms have been growing at a fast pace since the early 2000s and offer now a variety of products and services to their customers through digital devices. E-retailing platforms such as Ebay, Amazon.com, Alibaba and JD.com are changing the way people shop around the planet, while e-hailing platforms such as Lyft, Uber and Didi have reshaped how we commute. Moving the interaction with their customers online allowed internet platforms to gather valuable information for decision making. This data provides internet platforms with the necessary tools to make informed decisions on their key business strategies such as adapting their offer to the demand of regional markets.

An assortment of products is a subset of the products the internet platform offers to its customers in a regional market. To optimize their business metrics, e-retail platform design assortments of products to be held in warehouses while e-hailing platforms decide on which products to offer in what cities. The value of such an assortment may come from the synergies between different products. For instance, an e-retail platform holding products often purchased together in the same warehouse reduces its shipping costs and an e-hailing platform offering a premium service and a discounted one can cover several customer segments.

In this essay, we focus on selecting and studying assortments of products for e-retail and e-hailing platforms. We build data driven algorithms to guide such decisions. We provide thorough numerical analyses build on real world data sets to prove the performance of these methods.

E-retail platforms

E-retail platforms face many logistics challenges such as designing their delivery network, selecting assortments of products to be held in these warehouses and optimizing the replenishment policy. These problems are tied together but solved independently due to the intractability of the general problem. In this essay we focus on building the assortment of products for a given delivery network assuming the replenishment policy will minimize stock-outs of the assorted products.

Orders placed on an e-retail platform website may contain one or several products. If such an order can not be entirely fulfilled from a single warehouse, it is said to be split. Split orders typically result in a slower shipping speed and greater fulfillment costs. In Chapters 1 and 2, we develop algorithms to build assortments of products that minimize order split subject to capacity constraints. Acyclic delivery networks are studied in Chapter 1 and we extend these results to cyclic delivery networks in Chapter 2.

E-hailing platforms

E-hailing platforms cover multiple customer segments by offering different products such as X, Pool, Select, etc. for Uber; Lyft, Shared, Lux, etc. for Lyft; and Taxi, Express, Premier, etc. for DiDi. At a high level, these products can be partitioned between classic private rides and shared rides. Classic is a service that provides on demand transportation by dispatching drivers to passengers when a request is made. Shared is a service that matches riders travelling in the same direction with an available shared car. Classic is typically more expensive and Shared is often slower as sharing a vehicle may induce additional travel time. To keep Shared travel time reasonable, e-hailing companies set a detour constraint to place an upper bound on how much detour a rider can experience.

When two riders travelling in the same direction are matched, some supply cost is saved as a single car can serve both requests. The likelihood of such event increases with the demand density. Although a Shared request creates less revenue as the service is sold at a discount, a high density of demand coupled with an efficient matching algorithm, can result in a profitable market for the product. There is an economy of scale. To increase demand density, an e-hailing company can either offer greater discount thus decreasing the revenue per Shared riders or tighten the detour constraint, further constraining the matching algorithm and limiting the amount of feasible matches.

These two types of products are not offered together in every region where the e-hailing platform operates. Although offering Shared captures demand from other transportation solutions, it can also cannibalize demand from Classic. In Chapter 3, we study where and when it is profitable for an e-hailing company to offer both products.

Contents

List of Figures

Acknowledgments

I would like to express the deepest gratitude to my dissertation supervisor, Prof. Max Z. Shen for his support and guidance through the past few years. He has been a wonderful mentor and I thank him for the challenging and stimulating research experience. Above all, he has been genuinely supportive of my pursuits outside of my thesis work, both academic and otherwise, and I cannot thank him enough for this.

It has been a wonderful learning experience working with Prof. Daniel Freund and Dr. Arash Asadpour on the shared ride sustainability project. The manner in which they maintain the balance between applied and fundamental research is something I will always aspire to.

Working with Dr. Di Wu and Dr Yuhui Shi on supply chain optimization has been an immensely insightful experience. I admire their dedication to promote academic research in an industry environment.

I would like to thank Prof. Phil Kaminsky who introduced to supply chain optimization and sparked the motivation to further study this field and take on the challenge to complete a Ph.D. His course on supply chain optimization is what got me excited about research in the first place, and I am grateful to him for that. I am also grateful to Prof. Dorit Hochbaum for introducing me to network flows optimization and for being very supportive of my decision to start a Ph.D.

I would like to thank Professors Gideon Weiss, Alper Atamturk, Ilan Adler, Lee Flemming, Alexandre Baleiev, Paul Grigas, Deepak Rajan, Xin Guo, Christopher Paciorek, Nike Sun, Anant Sahai, Stella Yu, Javad Lavaei, John Canny, Alexei Efros, Jitendra Malik and Sergei Levine for providing me a great learning experience through their courses at the University of California Berkeley.

I cannot thank my colleague and friend, Dr. Quico Spaen, enough for helping me out at several instances and playing the role of the lab senior to perfection. I have learnt a great deal by working on research projects with Mark Velednistky and Cristobal Pais. I am also thankful to Alfonso Lobos who helped me early on during my Master to select courses and research projects in preparation of a Ph.D. track.

Last but not the least, I would like to thank my family: my parents and my sister for supporting me spiritually throughout writing this dissertation and in my life in general.

Part I

E-retail Platforms

Chapter 1

Data-Driven Inventory Placement for E-Tailer

In the US alone, e-commerce sales have grown each year by over 13% for the past 8 years and now account for more than 9% of the total retail sales ([9]). A survey ([29]) shows that 63% of customers see delivery speed as an important feature when choosing a product and 27% will pay attention to the same day delivery option when choosing an e-retailer. Shipping faster at a reduced or no premium is a competitive advantage in capturing a greater share of the growing e-commerce market. E-retailers are thus looking at cutting their delivery time in order to be more attractive to customers. Our business partner, JD.com, relies on a two-level delivery network. Regional distribution centers (RDCs) are large warehouses holding every SKU vended online. Forward distribution centers (FDCs) are closer to customers distance-wise to provide fast shipping services. These warehouses are smaller than the RDCs. They have constrained inventory and rely on daily shipments for replenishment. Because of the fast increasing number of SKUs JD.com sells on its website, the company needs to restrict the number of SKUs stored at each FDCs. This business constraint comes from the organization of JD.com's delivery network. An RDC groups several storage facilities receiving replenishment directly from suppliers. They can accommodate the sorting of all JD.com products. An FDC is a smaller single storage facility that relies on daily replenishment from only one RDC. To achieve this dynamic replenishment, the company needs high quality forecasting and fast sorting and storing at the FDC level. Moreover, given the pace at which the number of products grows and the packaging changes, it is near to impossible to keep track of every single SKU volume or weight, making the traditional capacity constraint unrealistic. JD.com thus periodically builds whitelists of SKUs to store and replenish in an FDC. Because of these constraints, it is vital for e-retailers to identify a subset

of SKUs to be kept at the FDC level, whose cardinality verifies the above constraints.

JD.com covers over 99% of the Chinese population over enormous distances. When covering such a large area, e-retailer structure their delivery network meticulously. A very connected network has high in stock rate but also high shipping cost per package due to the low utilisation of some routes. On the contrary, a less connected network has low in stock rate and low shipping cost thanks to fully utilized routes. To simplify the replenishment decisions and optimize its operating costs, JD.com selected the latter, structuring its delivery network as a tree (see figure 1.1). It is known in the company that more complex structures would enable risk pooling and more effective deliveries, these structures are also being studied and the results are shared in Chapter 2. In JD.com's network, each customer area is handled by a single FDC which relies on one RDC for replenishment. When products are missing from an FDC, they can be shipped directly from the RDC to the customers. In this setting, the assortment of each FDC can be studied independently. The rest of the Chapter will focus on a single RDC-FDC pair, by replicating the method for each FDC, the optimal assortment for each node in the tree is found.

Delivery speed is a key feature when a customer chooses an e-retailer, thus one of the main strategies of JD.com is the so called 211 program. Under this program, JD.com promises same day delivery for orders placed before 11 a.m. and next day delivery before 3 p.m. for orders placed between 11 a.m. and 11 p.m. Because of the structure of JD.com's delivery network, an order can benefit from the 211 program only if every items purchased are located in the closer FDC. If a single item is missing from the FDC, the order won't meet the service level of 211 program. When this happens, the order is either routed to the parent RDC or split in multiple packages. In both cases, the delivery speed of the whole order is delayed and considered a miss in JD.com's operational metrics. In order to optimize the delivery speed metrics, a reasonable proxy is to assume that if a single item is missing from the FDC, the entire order is routed to the RDC.

Although the traditional constraint used for managing assortment is volume, limiting the number of SKUs is a practical choice adopted by JD.com, as packaging of SKUs changes frequently, making it difficult to keep track of the volume and weight. The FDC of JD.com have operation constraints on the maximum number of distinct SKUs per storage shelf. This can help to improve the stowing and picking efficiency in the warehouse based on current operation procedures. Furthermore, as the assortment decision is also commonly aided by manual adjustments, a constraint by distinct SKU number is much easier to operate than a similar constraint by expected volume or total unit count. The frontline operator may not have sufficient information on the expected units per SKU in the storage. The simpler SKU count constraint provides them with a much more straightforward process to work with.

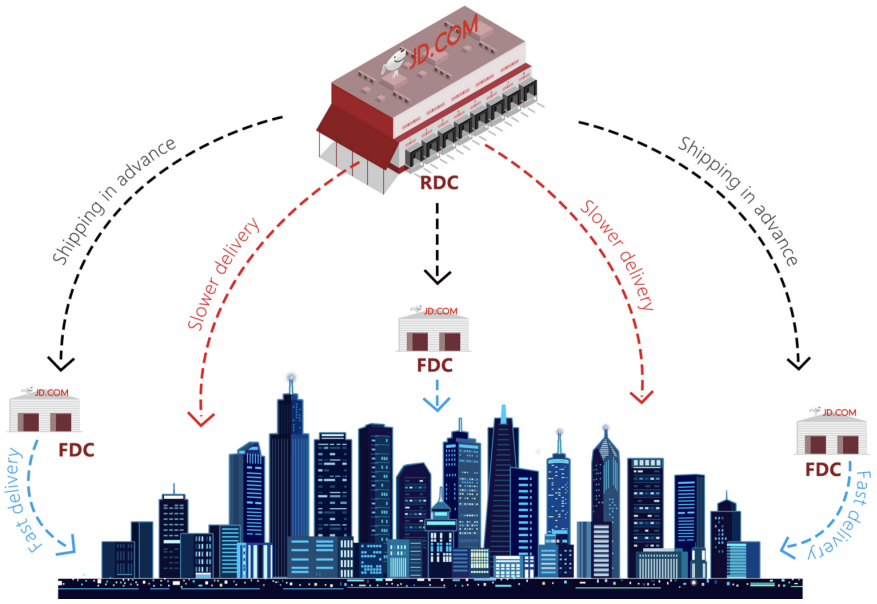


Figure 1.1: Simple RDC FDC network

A significant portion of orders placed on the online marketplace are composed of more than a single product. The subset of SKUs placed at FDC should maximize the number of orders that can benefit from fast shipping, i.e., that can be shipped from the closest FDC. Such orders are said to be *locally fulfilled*. In this setting, two distinct orders may share one or several products. The number of SKUs needed to fulfill both orders is smaller than the sum of their length. Consequently, unlike items of a knapsack problem, an order cannot be associated with an occupied volume in the warehouse.

As explained above, the delivery network of JD.com has lower in stock rate than a more connected network. It is thus primordial to optimize the selection of product in each DC. In this paper, we study the setting where an e-retailer needs to decide at the beginning of each cycle what subset of products to place in the FDC to maximize the number of orders that can be shipped from the FDC during the upcoming period. Because this problem does not have to be solved online, it is increasingly interesting to trade some of the computational efficiency of the industry standard algorithms for better performance. To achieve the trade-off, we propose a novel graph-based data-driven algorithm to output near-optimal solutions. We first relax the problem to a selection problem and show the equivalence between it and a minimum parametric cut problem that can be solved efficiently. To incorporate the stochasticity of demands, we introduce item-level demand forecasts into the problem and show how they can

be used in place of order level demand forecasts, which are nearly intractable given the combinatorial nature of possible order types.

1.1 Literature review

The problem of selecting subsets of products arises owing to two main objectives: maximizing the revenue under a capacity constraint and minimizing the order split given constrained containers or shelves. The first problem, also known as the assortment problem, has widely been studied for brick-and-mortar shops. The problem faced by e-retailers is determining how products may together constitute items sets of high values to optimize SKU assortment. Identifying item sets of high values is commonly tackled with association rules. [26] developed an efficient algorithm to mine association rules from large relational databases. An example of such a rule could be *"30% of the transactions that contain beers also contain diapers."* The algorithm starts with the generation of frequent item sets using an algorithm based on the Apriori algorithm; then it generates candidate rules to finally filter out the most relevant ones. The algorithm runs linearly with the number of entries in the database. [5] used data mining to identify frequently purchased item sets and then improved the assortment decision by solving an MIP that included this additional information. Each item set is associated with a positive weight in the objective such that selecting every item it is composed of increases the objective value. This algorithm has several orders of magnitude less variables than if every possible set were used. Solving this relaxed problem thus only provides a lower bound for the optimal decision. Other studies have focused on how to identify the relationship between products to build a better assortment. We point the reader to a survey of this topic [13].Association rules are more commonly used for maximizing revenue rather than for maximizing local fulfillment.

Machine learning techniques have been proposed to embed all products in a vector space such that the distance between two products becomes a proxy for how often they are ordered together. Assuming such an embedding is found, running clustering algorithms on the embedded vectors provides reasonable solutions for minimizing the order split. Google developed the famous word2vec architecture [8] that uses sentences as contexts to map words to a latent space. This algorithm can easily be extended such that orders are used as contexts to then map products to a latent space. Further architectures of neural networks have been explored by [30] to preserve the initial structure of the data to several levels. Most of the work mentioned before tends to keep the distance between two product vectors small if the products are often ordered together. Using the distance in the vector space as a proxy of how

often products are purchased together relies on the assumption that the last metric verifies the triangular inequality. Two products that are often ordered with a third one will lie in close range of each other no matter how often they are ordered together. A simple example is the case where two products A and A' that are substitutes of each other from the same category of products are frequently ordered with a third product B. In such a setting, the above algorithms will likely group the three products in the same cluster, though it is not trivial that they should be stored in the same place.

Several studies have tackled the organization of the warehouses to minimize shipping time. [20] reviewed different methods of organizing a warehouse to reduce the time needed to assemble an order. Among the techniques used, SKU grouping is relevant to our work. [21] designed a model that partitions the products into k subsets such that the minimum number of groups have to be accessed when retrieving a given order under a pick-by-order policy. However, the algorithm developed in this paper does not consider the stochasticity of demands and tries to solve the problem using ejection chain algorithms. We note that the problem we are studying here is much simpler as it needs only one subset that does overlap with the global set. However, the datasets considered in this Chapter are much larger and uncertainty cannot be ignored.

Our work uses parametric s-t cut minimization to output assortments in order to maximize the number of orders that can be fulfilled locally from FDC in the upcoming cycle. [3] shows that a selection problem can be solved faster by solving a minimum s-t cut on a bipartite graph than by solving a Linear Program (LP). [12] shows that the parametric minimum s-t cut is solved efficiently under some assumptions on the parametric capacity of the graph's arcs. Building on these two papers, [15] proposes an algorithm to decide what facility to open given a budget constraint. By solving a minimum parametric s-t cut problem, the paper computes an efficient frontier providing an upper bound on the performance of the optimal solution. [31] uses a similar algorithm to select a product assortment to maximize the revenue coverage. Both papers assume full knowledge of all variable. We solve a significantly different problem where the metric to maximize is the delivery speed of orders with an SKU count constraint. Furthermore, because e-retailers experience rapidly shifting demand, it is important to provide an algorithm that optimizes for the delivery speed of future orders. Therefore, our work focuses on solving a non-deterministic problem and cannot assume full knowledge of all variables. In this Chapter, simplifying assumptions are made to use parametric s-t cut minimization. Used jointly with ensemble methods and other heuristics, we provide insight and algorithms to robustly handle changing demand. Parametric cut minimization has also been used in many different fields such as image segmentation ([16], [18], [14]), but has not been very much used for supply chain problems or assortment problems.

There is to our knowledge no paper using this technique in a non-deterministic setting for solving a product assortment problem.

In summary, this Chapter proposes a novel algorithm to decide what products to place in the FDC given the large transaction databases of e-commerce companies. We first show how the deterministic problem can be solved as a parametric cut in section 1.3. We then present, in section 1.4, two algorithms extending the previous methods to non-deterministic setting assuming demand is stationary and we show how to use product level forecast. Finally, We provide the reader with preliminary analysis and information about the data used to generate results before comparing our work to the benchmark methods in Section 1.5.

1.2 Problem definition

E-retailers provide customers with an online catalog containing a large number of products. The DCs where these products are kept, however, are constrained by the number of SKUs they can hold. Unlike for brick-and-mortar stores, customers decisions on an e-commerce website are rarely influenced by stock-outs, as every product remain visible online. In this section we present a method for leveraging these observations to maximize, for a single RDC-FDC pair, the number of orders that can be fulfilled from the FDC under a capacity constraint.

Definitions and notation

Consider a pair of warehouses with one RDC and one FDC. The RDC holds every stock-keeping unit (SKU) and can fulfill any order. The FDC is dedicated to a single customer area. The number of SKUs that can be stored at the FDC level is constrained as explained above by capacity and daily replenishment. When an order is placed, either all products purchased are located in the FDC, or the order is routed to the RDC. The goal is to maximize the number of orders that will be fulfilled from the FDC in the upcoming period. Let I be the set of products stored in the RDC. For $i \in I$, the random variable G_i models the demand for product i with mean μ_{G_i}. G_i is the *product level demand* for product i. We define the random vector $\mathbf{G}_I = \{G_i\}_{i \in I}$ as a vector of product level demands with a mean $\mu_{\mathbf{G}_I}$.

Let O be the set of unique order types: the set of product combinations. For $o \in O$, o is a combination of products, and $o \subset I$. The demand for order type o is a random variable D_o with a mean μ_{D_o}. D_o is called the *order level demand*. We define the random vector $\mathbf{D}_O = \{D_o\}_{o \in O}$ as the vector of order level demands with a mean $\mu_{\mathbf{D}_O}$.

Let k be the maximum number of SKUs that can be placed in the FDC. $\mathscr{S} \subset I$ is a feasible assortment if $|\mathscr{S}| \leq k$. We define $f_{\mu_{\mathbf{D}_O}}$ such that $f_{\mu_{\mathbf{D}_O}}(\mathscr{S})$ is the expected number of orders that can be fulfilled with $\mathscr{S}$, given a random vector of demands $\mathbf{D}_O$, with mean $\mu_{\mathbf{D}_O}$.

Mathematical formulation

For each product $i \in I$, we associate a binary decision variable X_i that models whether or not product i is selected in the assortment. We let X_i be 1 if product i is selected and 0 otherwise. As the goal is to maximize the number of orders that can be fulfilled, for each order $o \in O$, we associate a binary decision variable Y_o that characterizes whether or not order o can be fulfilled from the FDC using assortment $\mathscr{S}$. The FDC cannot fulfill order $o \in O$ using assortment $\mathscr{S}$ if any product $i \in I$, purchased in order o is missing from the assortment. Therefore, Y_o needs to verify the constraint $Y_o \leq X_i,\ \forall o \in O,\ i \in I,\ i \in o$. We can formulate the problem as an MIP:

$$\max_{X,Y} E\Big[\sum_{o\in O} D_o Y_o\Big] = \sum_{o\in O} \mu_{D_o} Y_o \qquad (MIP)$$

$$s.t. \sum_{i\in I} X_i \leq k \qquad (1)$$

$$Y_o \leq X_i \qquad \forall o \in O,\ i \in I,\ \text{if } i \in o \qquad (2)$$

$$X_i \in \mathbb{B}, Y_o \in \mathbb{B}$$

The objective is to maximize the expected number of orders that can be fulfilled from the FDC, subject to (1) no more than k unique SKUs can be placed in the FDC and (2) if any product is missing from the FDC inventory then the order cannot be locally fulfilled. The k-densest subgraph problem reduces to a specific instance of (*MIP*). Therefore, the FDC inventory placement problem is NP-hard (see appendix A.1).

1.3 Solution to the problem given known demands

In this section, we assume that the mean demand for each order type $o \in O$ is given as μ_{D_o}. As we have shown above (1.2) that the FDC inventory placement is NP-hard, we first derive the Lagrangian relaxation of the problem in subsection (1.3) and we show that this problem reduces to a selection problem that is equivalent to

a minimum s-t cut problem on a bipartite graph in subsection (1.3). In subsection (1.3), we show that every integer solution to the Lagrangian relaxation for any value of the Lagrangian parameter λ can be found with a complexity no greater than the complexity of solving for a single minimum s-t cut in the bipartite graph. Finally, in subsection (1.3), we define a product score for each SKU that will be used for solving the non-deterministic problem.

Lagrangian relaxation

As we showed in subsection (1.2) that the problem (MIP) is NP-hard, we first consider solving the Lagrangian relaxation of the original formulation. We define λ as the Lagrangian multiplier, the Lagrangian relaxation of the problem is given by

$$\begin{aligned} \max \ & \sum_{o \in O} \mu_{D_o} Y_o - \lambda \sum_{i \in I} X_i && && (LR) \\ & Y_o \leq X_i && \forall o \in O,\ i \in I,\ \text{if } i \in o && (1) \\ & X_i \in \mathbb{B}, Y_o \in \mathbb{B} \end{aligned}$$

(LR) is a problem that can be solved efficiently. For a fixed λ, (LR) is a selection problem where each order is a set with benefit μ_{D_o} and each product is an item with cost λ. The constraint matrix of this problem has exactly one coefficient being $+1$ and one coefficient being -1 for each row. The constraint matrix is thus totally unimodular, and there exists an optimal linear programming solution that is an integer solution. [3] showed that this problem can be solved faster by solving a minimum s-t cut problem in a graph than by solving the linear program.

Solving a selection problem as a minimum s-t cut

In the following paragraph, we construct a bipartite graph on which solving for a minimum s-t cut is equivalent to solving LR. This graph has on the left-hand side a set of nodes V_I such that for each SKU $i \in I$, a corresponding node v_i lies in V_I. On the right-hand side, it has a set of nodes V_O such that for each unique order type $o \in O$, a corresponding node v_o lies in V_O. We define $G = (V = \{s\} \cup V_I \cup V_O \cup \{t\}, A)$ as a bipartite graph with V being the set of nodes and A the set of arcs. For each order type $o \in O$ and each product $i \in I$, if i is placed in o, i.e., $i \in o$, an arc of capacity infinity links the corresponding SKU node v_i to the corresponding order node v_o, i.e., $(v_i,\ v_o) \in A$. In addition, there exists a source node s and a sink node t for the graph, and for each SKU $i \in I$, we add an arc $(s,\ v_i)$ of capacity $u_{s,\ v_i} = \lambda$

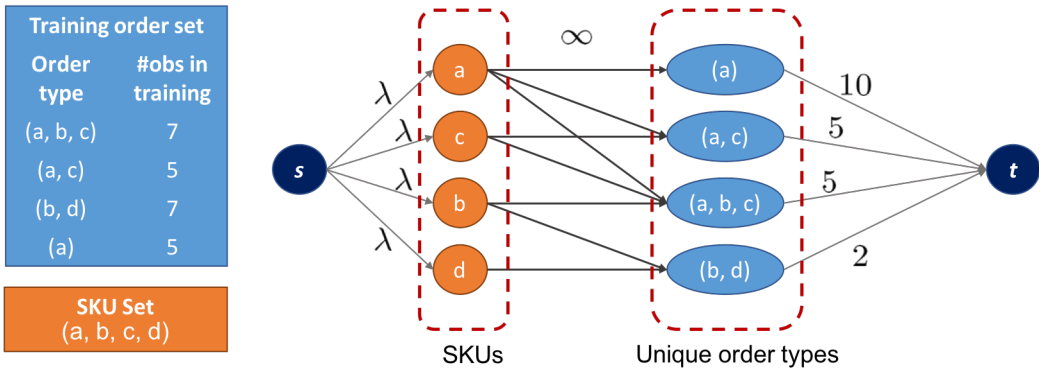


Figure 1.2: Example of a graph corresponding to a set of orders

to A. Furthermore, for each order type $o \in O$, we add an arc $(v_o,\ t)$ of capacity $u_{v_o,\ t} = \mu_{D_o}$ to A. Figure 1.2 provides an example of such a bipartite graph.

An s-t cut in this graph is a partition of the node set $\{s\} \cup V_I \cup V_O \cup \{t\}$ to two subsets S and its complement $\bar{S} = V \setminus S$ such that $s \in S$, $t \in \bar{S}$ and $S \cap \bar{S} = \emptyset$. We define the capacity of the cut $C(S,\ \bar{S})$ as the sum of the capacities of every arc linking a node in S to a node in $\bar{S}$, i.e., $C(S,\ \bar{S}) = \sum_{v \in S, v' \in \bar{S}} u_{v,v'}$. For a given Lagrangian parameter λ, let G_λ be the bipartite graph build as instructed above. Solving for the minimal source set minimum s-t cut, we find a minimum cut $(S^*_\lambda, \bar{S}^*_\lambda)$, where $S^*_\lambda = \arg\min_S\{|S| : (S, \bar{S}) \text{ is a minimum cut in } G_\lambda\}$. Given $(S^*_\lambda, \bar{S}^*_\lambda)$, setting X_i (resp. Y_o) to 1 for all products $i \in I$ (resp. order type $o \in O$) such that $v_i \in \bar{S}^*_\lambda$ (resp. $v_o \in \bar{S}^*_\lambda$) and 0 otherwise is optimal for (LR).

Let us prove the above statement. We first show that solving for a minimum s-t cut is a feasible solution for (LR). Note that because $S = \{s\}$ is a feasible s-t cut with $C(S,\ \bar{S}) = \lambda \times |I|$, any minimum s-t cut is finite. Consequently, no arcs of infinite capacity can lie in a minimum s-t cut, i.e., if two nodes v and v' are linked by an arc $(v,\ v')$ of infinite capacity, $v \in S^*_\lambda \implies v' \in S^*_\lambda$. For each product $i \in I$ and for each order type $o \in O$ in which product i is placed, an arc (v_i, v_o) of capacity infinity is added to the graph. Therefore, v_o can lie in $\bar{S}^*_\lambda$ only if for all products $i \in I$ placed in o, v_i lies in $\bar{S}^*_\lambda$. This indicates that setting X_i (resp. Y_o) to 1 for all products $i \in I$ (resp. order type $o \in O$) such that $v_i \in \bar{S}^*_\lambda$ (resp. $v_o \in \bar{S}^*_\lambda$) and 0 otherwise is feasible for (LR).

Let us show it is optimal for (LR) by deriving the capacity of the minimum s-t

cut:

$$\begin{aligned} C(S^*_\lambda, \bar{S}^*_\lambda) &= \min_S \sum_{v_i \in V_I \cap \bar{S}} \lambda + \sum_{v_o \in V_O \cap S} \mu_{D_o} + \sum_{v_i \in V_I \cap S, v_o \in V_O \cap \bar{S}} \infty 1_{i \in o} \\ &= \max_S \sum_{v_o \in V_O \cap \bar{S}} \mu_{D_o} - \sum_{vi \in V_I \cap \bar{S}} \lambda + \Big(\sum_{o \in O} \mu_{D_o} \Big) \end{aligned}$$

Note that $\sum_{o \in O} \mu_{D_o}$ is a constant. Therefore, setting X_i (resp. Y_o) to 1 for all products $i \in I$ (resp. order type $o \in O$) such that $v_i \in \bar{S}^*_\lambda$ (resp. $v_o \in \bar{S}^*_\lambda$) and 0 otherwise gives

$$\sum_{o \in O} \mu_{D_o} Y_o - \sum_{i \in I} \lambda X_i = \max_S \sum_{v_o \in V_O \cap \bar{S}} \mu_{D_o} - \sum_{vi \in V_I \cap \bar{S}} \lambda$$

Therefore, solving for the minimum s-t cut solves (LR). For a given Lagrangian parameter λ and $\bar{S}^*_\lambda$ the sink set of the minimum s-t cut in the bipartite graph constructed as instructed above, $\mathscr{S}_\lambda = \{i \in I | v_i \in \bar{S}^*_\lambda\}$ is the optimal set of products for this selection problem with fixed λ. Let $k' = |\mathscr{S}_\lambda|$. Then if the capacity of the FDC is k', $\mathscr{S}_\lambda$ is the optimal solution to the inventory placement problem (MIP).

Solving for all ranges of λ by solving for the parametric minimum s-t cut

In this subsection we will show that we can solve (LR) for any range of Lagrangian parameter λ efficiently. This will provide a list of nested assortments of products that are optimal for specific capacity constraints. This has high value for a company as it provides an upper bound for the optimal value for any capacity constraint and an optimal solution for some capacity constraints. The company can then decide to adapt the capacity of the FDC accordingly.

Let $T(n, m)$ be the time complexity for solving a minimum s-t cut on a graph with n nodes and m arcs. For example, the pseudoflow algorithm used in this paper achieves $T(n, m) = O(mn \log(\frac{n^2}{m}))$([17]). [12] showed that the solving the minimum s-t cut problem in a graph where the arc leaving the source nodes have a capacity increasing with some parameter λ can be done simultaneously for any range of λ in the same complexity as a single minimum s-t cut problem. Solving for the minimum s-t cut on such a graph for a range of λ is called solving for the minimum parametric s-t cut. For a fixed Lagrangian parameter λ, solving for the minimum s-t cut is equivalent to solving (LR). Therefore, solving the minimum parametric cut problem is equivalent to solving (LR) for any range on $\lambda \geq 0$ and thus (LR) is solvable in $O(T(n, m))$.

[17] proved that source sets of the parametric cut verify the *nestedness property*: for $\lambda_2 \leq \lambda_1$, $S^*_{\lambda_2} \subseteq S^*_{\lambda_1}$. We showed in subsection (1.3) that for a given λ, the optimal assortment is the set of products whose product nodes lie in $\bar{S}^*_\lambda$. Therefore, if $\lambda_1 \geq \lambda_2$, $\mathscr{S}_{\lambda_1} \subseteq \mathscr{S}_{\lambda_2}$. When λ decreases, this algorithm outputs nested optimal assortments of products for increasing capacities.

We will now show that the capacity of the minimum parametric s-t cut is a piecewise linear function of λ. We thus call values of λ that correspond to a breakpoint of the capacity of the minimum parametric s-t cut *breakpoints*. Note that, given a feasible source set S, $\lambda \mapsto \lambda|V_I \cap \bar{S}| + \sum_{v_o \in V_O \cap S} \mu_{D_o}$ is a linear function of λ. Let $g(\lambda) = \lambda|V_I \cap \bar{S}^*_\lambda| + \sum_{v_o \in V_O \cap S^*_\lambda} \mu_{D_o}$ it follows from the definition that g is the minimum of a set of linear functions of λ; it is thus a piecewise linear concave function. Furthermore, as for any feasible source set S, $|V_I \cap S| \geq 0$, it is also a non-decreasing function. We can conclude that the capacity of the parametric minimum s-t cut is a piecewise linear non decreasing function of lambda.

λ is said to be a *breakpoint* if for any $\epsilon > 0$, $S^*_{\lambda-\epsilon} \subset S^*_\lambda$, that is, when the slope of g changes. For $\lambda_2 \leq \lambda_1$, because of the nestedness property, we have $S^*_{\lambda_2} \subseteq S^*_{\lambda_1}$. Therefore, we cannot find more than $|I|$ distinct nested optimal source sets. This indicates that at most $|I| - 1$ breakpoints can be found.

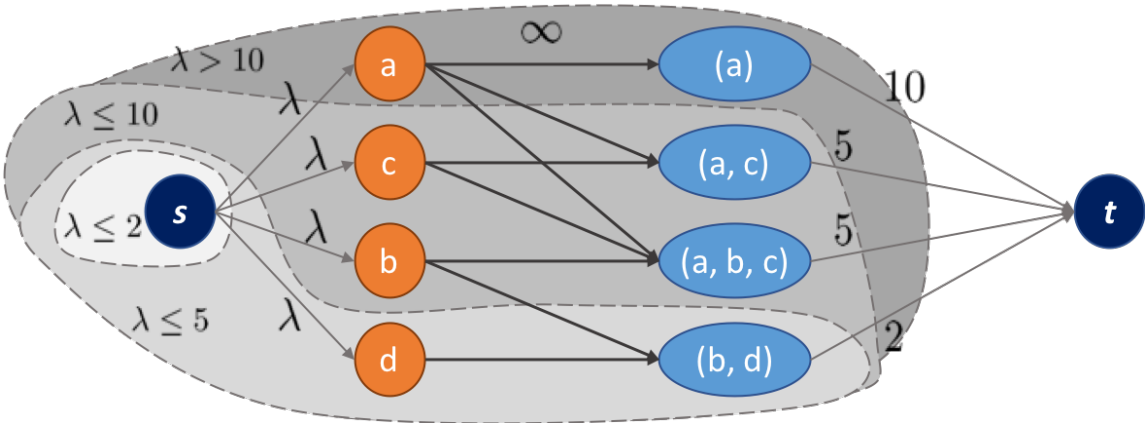


Figure 1.3: Nested cuts obtained by solving the parametric cut for Figure 1.2

Solving the minimum parametric cut problem outputs a set of $q \leq |I|$ nested sets such that $\lambda_1 > \lambda_2 > \cdots > \lambda_q$ are the breakpoints and $\bar{S}^*_{\lambda_1} \subset \bar{S}^*_{\lambda_2} \subset \cdots \subset \bar{S}^*_{\lambda_q}$ are the corresponding sink sets of the minimum s-t cut. Figure 1.3 shows nested minimum s-t cuts for increasing values of λ.

This indicates that solving the parametric minimum s-t cut outputs nested sets of products $\mathscr{S}_{\lambda_1} \subset \mathscr{S}_{\lambda_2} \subset \cdots \subset \mathscr{S}_{\lambda_q}$ for the decreasing Lagrangian parameter $\lambda_1 > \lambda_2 > \cdots > \lambda_q$ such that the selection is optimal for the relaxation of the FDC inventory placement problem (LR). Furthermore, by letting $k_i = |\mathscr{S}_{\lambda_i}|$, we have q optimal assortments respectively for the q capacities $k_1,\ k_2,\ \ldots, k_q$.

Solving the minimum parametric s-t cut does not solve all instances of (MIP). A counter example is $O = \{(a), (b,c), (a,b)\}$ such that $D_{(a)} = 3$, $D_{(d,e)} = 4$, and $D_{(a,b,c)} = 5$ with a probability of 1. We can easily that the optimal solutions to (MIP) and their objective value for unit increasing capacity are $\{\mathscr{S} = (a),\ f_{\mu_{\mathbf{D}_O}}(\mathscr{S}) = 3\}$, $\{\mathscr{S} = (d,e),\ f_{\mu_{\mathbf{D}_O}}(\mathscr{S}) = 4\}$, $\{\mathscr{S} = (a,b,c),\ f_{\mu_{\mathbf{D}_O}}(\mathscr{S}) = 8\}$, and $\{\mathscr{S} = (a,b,c,d,e),\ f_{\mu_{\mathbf{D}_O}}(\mathscr{S}) = 12\}$. In such an instance, solving the parametric cut outputs the nested sets (a), (a,b,c), (a,b,c,d,e) and their corresponding breakpoints. Note that, if solving the minimum parametric s-t cut problem provided a solution for any instance of (MIP), it would contradict the assumption $P \neq NP$, as (MIP) is NP-hard, and the parametric minimum s-t cut problem is polynomial.

Product score

We have shown in the previous subsection (1.3) that solving the relaxation (LR) of the original problem outputs solutions optimal for the original problem for specific capacities. E-retailers sometimes set constraints on the marginal benefits of adding an SKU to the assortment rather than on the size of the assortment. In this section, we introduce the concept of a product score. This product score will later be used in sampling and aggregating to refine the solutions obtained from solving the relaxation.

Let us define for each product $i \in I$ a *product score* r_i that corresponds to the maximum value of the Lagrangian parameter such that product i is included in the FDC assortment, i.e., λ is such that $r_i = \max\{\lambda \in \{\lambda_1, \ldots, \lambda_q\} | i \in \mathscr{S}_\lambda\}$. To obtain a solution to the FDC inventory placement problem, we select products in decreasing order of product scores.

Note that the definition of breakpoints places an upper bound on the marginal benefit of adding an additional product to the FDC. We have proven that the capacity of the minimum parametric s-t cut is a piecewise linear concave function. It follows that at each breakpoints, at least two s-t cut have equal capacities. Given two consecutive breakpoints λ_{l-1} and λ_l (recall that $\lambda_{l-1} > \lambda_l$), by definition the corresponding minimum s-t cut $(S^*_{\lambda_l},\ \bar{S}^*_{\lambda_l})$ is optimal for $\lambda = \lambda_l$ but not for $\lambda_l + \epsilon$ for any $\epsilon > 0$. Recall that $\lambda_l < \lambda l - 1$, it follows that the two solution have equal

capacity when the parameter λ is set to λ_l.

$$\sum_{v_i \in V_I \cap \bar{S}^*_{\lambda_l}} \lambda_l + \sum_{v_o \in V_O \cap S^*_{\lambda_l}} \mu_{D_o} = \sum_{v_i \in V_I \cap \bar{S}^*_{\lambda_{l-1}}} \lambda_l + \sum_{v_o \in V_O \cap S^*_{\lambda_{l-1}}} \mu_{D_o}$$

$$|\mathscr{S}_{\lambda_l}|\lambda_l + \sum_{v_o \in V_O \cap S^*_{\lambda_l}} \mu_{D_o} = |\mathscr{S}_{\lambda_{l-1}}|\lambda_l + \sum_{v_o \in V_O \cap S^*_{\lambda_{l-1}}} \mu_{D_o}$$

$$\lambda_l = \frac{\sum_{v_o \in V_O \cap (S^*_{\lambda_{l-1}} \setminus S^*_{\lambda_l})} \mu_{D_o}}{|\mathscr{S}_{\lambda_l} \setminus \mathscr{S}_{\lambda_{l-1}}|}$$

$$= \frac{\#\text{additional fulfillable orders when going from } \mathscr{S}_{\lambda_{l-1}} \text{ to } \mathscr{S}_{\lambda_l}}{\#\text{product added when going from } \mathscr{S}_{\lambda_{l-1}} \text{ to } \mathscr{S}_{\lambda_l}}$$

This result follows from $\mathscr{S}_{\lambda_{l-1}} \subset \mathscr{S}_{\lambda_l}$ and $S^*_{\lambda_l} \subset S^*_{\lambda_{l-1}}$ proven in part (1.3).

$\mathscr{S}_{\lambda_{l-1}}$ remains optimal for $\lambda \in [\lambda_l, \lambda_{l-1}]$, for $\lambda \leq \lambda_l$, $\mathscr{S}_l$ is optimal. For any $\mathbf{s} \subset \mathscr{S}_{\lambda_l} \setminus \mathscr{S}_{\lambda_{l-1}}$, $\mathscr{S}_{\lambda_{l-1}} \cup \mathbf{s}$ is suboptimal or not maximal for the selection problem with $\lambda \in [\lambda_{l+1}, \lambda_l]$. Therefore, the marginal increase of adding $\mathbf{s}$ to $\mathscr{S}_{\lambda_{l-1}}$, defined as $\frac{f_{\mu_{\mathbf{D}_O}}(\mathscr{S}_{\lambda_{l-1}} \cup \mathbf{s}) - f_{\mu_{\mathbf{D}_O}}(\mathscr{S}_{\lambda_{l-1}})}{|\mathbf{s}|} = \frac{\#\text{addition fulfillable orders}}{\#\text{additional products}}$, is no greater than λ_l. The products lying in $\mathscr{S}_{\lambda_l}$ and not in $\mathscr{S}_{\lambda_{l-1}}$ are called the l^{th} layer of products added to the FDC inventory.

This indicates that every product lying in the l^{th} layer has a product score of λ_l. Consequently, each product in a layer has a product score equal to the marginal benefit of its layer. Furthermore, any subset of the l^{th} layer of products yields a marginal benefit as a set smaller than or equal to λ_l.

Note that the product score, i.e., $\lambda_l = \frac{\#\text{addition fulfillable orders}}{\#\text{additional products}}$, of every item lying in the l^{th} layer, i.e., $i \in \mathscr{S}_{\lambda_l} \setminus \mathscr{S}_{\lambda_{l-1}}$, corresponds to the average number of orders each of these products add to the set of locally fulfillable order types. For a business that selects breakpoint solutions, this score can be interpreted as the marginal benefit of adding a product to the DC, provided that every product with greater score have been already assigned. For industry use, it may be interesting to know the product score of an SKU with respect to a given assignment. This can easily be done by modifying the graph on which the parametric minimum cut is solved. For each SKU already assigned, remove the arc between the source and the SKU node. Removing the arc ensures the product node always fall in the sink set and thus is always selected in the assortment. Solving the parametric cut and computing the product score will return the marginal benefit of each product with respect to the given starting assignment. This product score provides managerial implication on evaluating the benefit of an additional product which can be compared to the cost of sorting and storing an additional product.

1.4 Solution approach to the non-deterministic problem

In the following section, we present three algorithms to solve the problem under specific assumptions. These algorithms heavily rely on the approach we chose for solving the deterministic FDC inventory placement problem (see section 1.3). Given $\mathscr{T}$ time steps, for $t \in \{1, \ldots, \mathscr{T}\}$, D_o^t and G_i^t are the random variables respectively modeling the order level and the SKU level demand. For a given decision cycle τ, the realization d_o^t and g_i^t of these random variables for $t \in \{1, \ldots, \tau - 1\}$ can be used to decide what assortment $\mathscr{S}$ to select for the upcoming cycle. In the first subsection (1.4), we assume the order level demands are stationarily distributed random variables. As it is difficult to estimate order level demands, we compute a very sparse Monte Carlo estimate assuming that every unobserved order type is removed for the set of order types O. In the second subsection (1.4), we use random sampling and aggregating to increase the stability of the solution. Finally, in the last subsection (1.4), the stationary order demands assumption is relaxed. Using an oracle for SKU level demands, SKU costs are derived to compute better assortments.

Product placement under stationary order level demands

Assuming that the order demands are stationary, for a given order type $o \in O$, for $t \in \{1, \ldots, \mathscr{T}\}$, $\mu_{D_o^t} = \mu_{D_o}$ is a static quantity. For each decision cycle, the sales observation is a random sample of the demand with respect to the stationary distribution. We can thus obtain a Monte Carlo estimate $\bar{\mu}_{D_o^t}$ of $\mu_{D_o^t}$ by simply averaging the number of times order type $o \in O$ is placed during the last decision cycles. This assumption is restrictive, however, as it greatly reduces the complexity. The number of possible order types grows exponentially with the number of products considered. However, most of the possible order types are never observed in the transactions data. Therefore, with the above assumption, each order type $o \in O$ that is not observed in the training set by setting $\bar{\mu}_{D_o^t} = 0$ can be ignored.

Let cycle τ be the upcoming cycle. Replacing $\mu_{D_o^\tau}$ by the estimate $\bar{\mu}_{D_o^\tau}$ in (LR) gives the following mathematical formulation:

$$
\begin{aligned}
\max \ & \sum_{o \in O} \bar{\mu}_{D_o^\tau} Y_o - \lambda \sum_{i \in I} X_i && && (SLR)\\
& Y_o \leq X_i && \forall o \in O,\ i \in I,\ \text{if } i \in o \\
& X_i \in \mathbb{B}, Y_o \in \mathbb{B}
\end{aligned}
$$

For the selection problem (see section 1.3), setting $\bar{\mu}_{D_o^\tau} = 0$ is equivalent to removing o from O, as the value of Y_o is not significant. This reduces the number of nodes in the corresponding bipartite graph on which minimum parametric cut is solved, thus reducing the time complexity (see 1.3). Empirically, with this assumption, the number of order types, is only one order of magnitude larger than the number of products.

Algorithm 1: Stationary PC

Data: Training set of several cycles of transaction data
Capacity k
Result: Assortment of products to maximize future order fulfillment from the FDC
Step 1: Ignore every order types not observed in the training set
Step 2: Compute the Monte Carlo estimate $\bar{\mu}_{D_o^\tau}$ for every order type in the training set
Step 3: Solve (SLR) by solving for minimum parametric s-t cut on the graph as detailed in section (1.3) with Pseudo-Flow algorithm [14]
Step 4: Find the assortment for each breakpoint
Step 5: Select the assortment $\mathscr{S}_{\lambda_l}$ corresponding the the breakpoint λ_l such that the capacity k_l of the assortment is the closest to the capacity constraint k of the original problem (MIP)
return $\mathscr{S}_{\lambda_l}$

Solving the minimum s-t parametric cut using $\bar{\mu}_{D_o}$ in place of μ_{D_o} outputs nested assortments along with product scores that can be used to decide what subset of SKU to keep in the FDC. We will refer to this algorithm as *Stationary PC*.

Though there is no guarantee that a breakpoint will be found in a range of the Lagrangian parameter λ, algorithm *Stationary PC* yields many breakpoints and rather small layers. We removed the 5% SKUs with highest product score and the 40% SKUs with smallest product score as they respectively always fall in or out of the assortment. To be most conservative, we remove these easy to assort SKUs and empirically measure the size of the layers as a percentage of the 55% remaining SKUs. We observe that most layers contains less than 0.1% of the SKU of interest (see Figure 1.4).

Bootstrapping and aggregating results

In the previous section, we explained how to compute a Monte Carlo estimate $\bar{\mu}_{D_o}$ of μ_{D_o} for $o \in O$ using the training transaction data set and use these in place of μ_{D_o} to solve the problem. This algorithm yields very unstable assortment prediction.

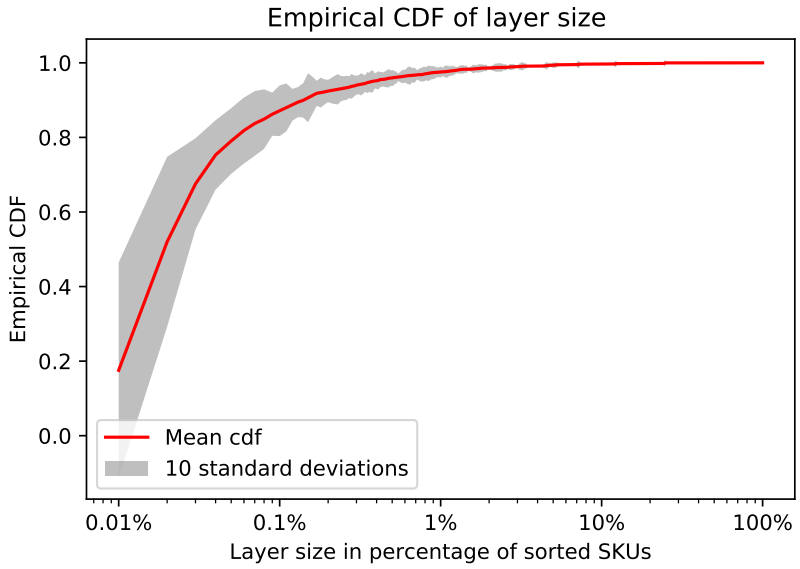


Figure 1.4: Empirical CDF of layer sizes measured as percentage of the number SKUs of interest

Instability in a prediction problem occurs when a small perturbation of the training set causes significant changes in the prediction. Figure 1.5 provides an example of two training sets that differ from each other by a single unit of order level demand moved from one order type (e, f) to (b, c). Solving for each training set outputs two predicted assortment with empty intersection.

[4] proposes an ensemble algorithm that can greatly reduce instability and improve accuracy. The Bagging predictor described in this paper is a method for generating multiple versions of a predictor and using these to build an aggregated predictor. The idea is to build multiple learning sets by making bootstrap replicates of the training set. Recall that a bootstrap sample is a uniform sample with replacement. A predictor is trained on each learning set and the aggregation averages the numerical outcome of each predictor. This idea can easily be extended to our setting. In our work, we create learning transaction sets by making bootstrap replicates of the training transaction set. By solving for the minimum parametric *s-t* cut using algorithm *Stationary PC* for each replicate of training data, we get a numerical product score for each product in I. The aggregation averages the scores of each product $i \in I$ provided by each result of algorithm *Stationary PC* over a replicate.

The training transaction data set contains all the transactions of the most re-

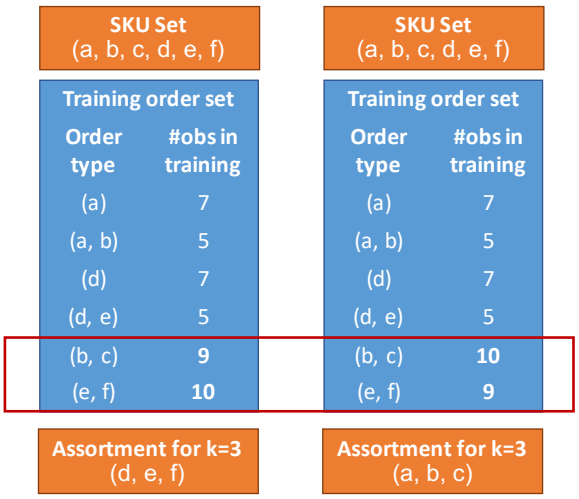


Figure 1.5: Example of two very similar transaction sets leading to significantly different assortments

cent cycles. We sample N batches of b orders selected uniformly at random with replacement from the transaction data. b is set to the average number of orders placed during a cycle. For each batch, solving the minimum parametric cut outputs a vector of product scores. Missing products receive a score 0. The aggregated score of product $i \in I$ over batches is $\hat{r}_i = \frac{1}{N} \sum_{j=1}^{N} r_i^j$ where r_i^j is the product score output for i obtained by solving the minimum parametric cut on the graph corresponding to the j^{th} batch of transaction data. This is algorithm *Bagging PC*.

Algorithm 2: Bagging PC

Data: training set of several cycles of transaction data
capacity constraint k
number of batches N
batch size b
Result: Assortment of products to maximize future order fulfillment from the FDC

Step 1: **for** $j \leftarrow 0$ **to** N **do**
  Create a bootstrap replicate of the training set by uniformly sampling b orders out of the transaction dataset:
  Sum up the observation to get the replicate $\mathbf{d}_O$ such that $\sum_{o \in O} d_o = b$
  Run algorithm 1 with $\mathbf{d}_O$ as the estimate $\bar{\mu}_{\mathbf{D}_O}$; **for** $i \in I$ **do**
    Define $r_i^j = \max\{\lambda | i \in \mathscr{S}_\lambda\}$

Step 2: Define $\hat{r}_i = \frac{1}{N} \sum_{j=1}^{N} r_i^j$
Step 3: Define $\mathscr{S}$ as the set of k products of highest score $\hat{r}_i$
return $\mathscr{S}$

Note that for the general problem with an expected demand vector $\mu_{\mathbf{D}_O}$, optimal solutions for increasing capacity are not necessarily nested. Recall the example we provided in section 1.3, where $O = \{(a), (d, e), (a, b, c)\}$ such that $\mu_{D_{(a)}} = 3$, $\mu_{D_{(d,e)}} = 4$ and $\mu_{D_{(a,b,c)}} = 5$. The optimal solutions to (*MIP*) and their objective value for unit increasing capacity are $\{\mathscr{S} = (a),\ f_{\mu_{\mathbf{D}_O}}(\mathscr{S}) = 3\}$, $\{\mathscr{S} = (d, e),\ f_{\mu_{\mathbf{D}_O}}(\mathscr{S}) = 4\}$, $\{\mathscr{S} = (a, b, c),\ f_{\mu_{\mathbf{D}_O}}(\mathscr{S}) = 8\}$, and $\{\mathscr{S} = (a, b, c, d, e),\ f_{\mu_{\mathbf{D}_O}}(\mathscr{S}) = 12\}$. However, Algorithm 2 outputs a ranking of the products; therefore, for two different capacity constraints, it returns two nested assortments. Consequently, it does not solve (*MIP*) in a non-deterministic setting. Running algorithm 2 on the above example outputs $\{\mathscr{S} = (a),\ f_{\mu_{\mathbf{D}_O}}(\mathscr{S}) = 3\}$, $\{\mathscr{S} = (a, b),\ f_{\mu_{\mathbf{D}_O}}(\mathscr{S}) = 3\}$, $\{\mathscr{S} = (a, b, c),\ f_{\mu_{\mathbf{D}_O}}(\mathscr{S}) = 8\}$, and $\{\mathscr{S} = (a, b, c, d, e),\ f_{\mu_{\mathbf{D}_O}}(\mathscr{S}) = 12\}$. It is not the optimal solution for (*MIP*); it however generalizes well if order level demands receive a small perturbation, and it does not show a large optimality gap. As this algorithm reduces the instability, the output assortment generalizes better, thus leading to a performance improvement. In the section 1.5, we show that the experimental results are very promising.

Product placement under non stationary product level demand

Assuming stationary demand is strong, given that most products have a life cycle, for a given cycle $\tau \in \{1, \ldots, \mathscr{T}\}$, solving Algorithm *Stationary PC* or 2 using the

estimate $\bar{\mu}_{\mathbf{D}_O^\tau}$ computed with past demand realizations or using the demand realization vector $\mathbf{d}_O^\tau$ of the upcoming cycle as demands estimate, we observe a gap in term of objective values between the two solutions. This is attributed to product level demand that can vary with seasonality and promotions. Figure 1.6 shows an example of a demand pattern that is not addressed by the former algorithms. In this example, the periodic cycle of high sales and low sales is nearly in phase with the training/testing transaction sets. When testing on a high sales period (resp. on a low sales period), the algorithm is trained on a low sales period (resp. on a high sales period). Therefore, the SKU is selected by the algorithm for upcoming cycle of low sales and left out of the assortment for upcoming cycles of high sales, which prevents it from appearing in the FDC inventory when needed. Assuming an oracle, e.g., the forecasting team of the company, can provide expected demand at the product level, by incorporating the knowledge of seasonality, cycles, etc., we need to relax the stationary assumption and improve the inventory placement decision by taking in such additional information.

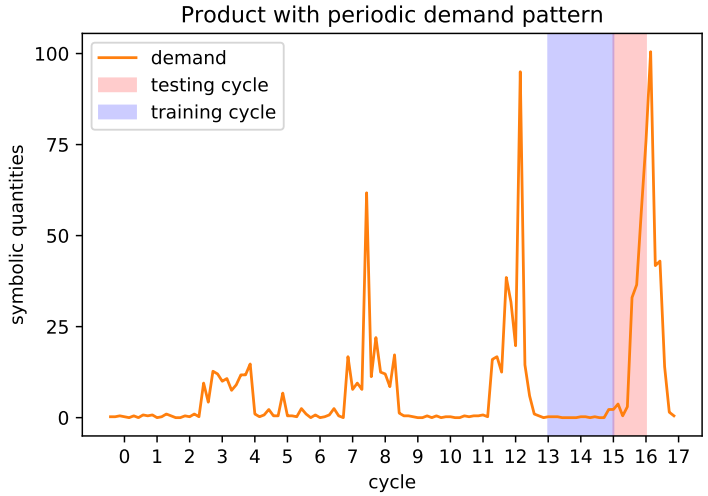


Figure 1.6: Example of a product having a periodic demand pattern

$\mu_{\mathbf{G}_I^t}$ is defined as $\mu_{\mathbf{G}_I^t} = \{\mu_{G_i^t}\}_{i \in I}$, the vector of product level demand. Assuming that an oracle provides an estimate $\bar{\mu}_{\mathbf{G}_I^t} = \{\bar{\mu}_{\mathbf{G}_I^t}\}_{i \in I}$ of the mean product level demand vector, the assumption of stationary demand is relaxed. We defined α^t (see section 1.2) as the sparse matrix that maps the spread of product level demands to order level demands. Recall that for $i \in I$, $\sum_{o \in O} \alpha_{i,o}^t = 1$ and $\mu_{D_o^t} = \alpha_{i,o}^t \mu_{G_i^t}$. A natural assumption is to assume that product level demands is divided into order level demands stationarily, i.e., α is stationary.

This assumption is weaker than the stationary assumption because it allows product level demands and order level demands distributions to change with respect to time. We observe experimentally that even though demands can widely change from one cycle to the next, sales for a given item $i \in I$ are divided into sales of order types in similar proportions. Therefore, assuming that α is stationary is a reasonable assumption. Given an estimate of the product level demand vector $\bar{\mu}_{\mathbf{G}_I^t}$, under this assumption, an estimate of the order level demand vector $\bar{\mu}_{\mathbf{D}_O^t}$ can be computed. Such an estimate would be nearly impossible to compute directly without this assumption as the size of the vector would grow exponentially with the number of products.

Given this assumption and the product level demand estimate vector $\bar{\mu}_{\mathbf{G}_I^\tau}$ provided by the oracle for the cycle τ, the straightforward approach is to construct an order level demands estimate $\bar{\mu}_{\mathbf{D}_O^\tau}$. We cannot simply set $\bar{\mu}_{D_o^\tau}$ to $\alpha_{i,o}\bar{\mu}_{G_i^\tau}$ for some product i placed in order type o. For two products $i, i' \in I$ such that both i and i' are placed in order type o, i.e., $i \in o$ and $i' \in o$, there is indeed no guarantee that $\alpha_{i,o}\bar{\mu}_{G_i^\tau} = \alpha_{i',o}\bar{\mu}_{G_{i'}^\tau}$. Therefore, assuming that α is stationary, our best guess is to build an estimate $\bar{\alpha}^{\tau-1}$ of the spread over the training set and an estimate $\bar{\alpha}^\tau$ of the spread over the testing set that provides feasible order level demands while minimizing $||\bar{\alpha}^{\tau-1} - \bar{\alpha}^\tau||_F$:

$$
\begin{aligned}
&\min_{\bar{\alpha}^\tau} ||\bar{\alpha}^{\tau-1} - \bar{\alpha}^\tau||_F && && \text{(CP)}\\
&\text{s.t.} \sum_{o \in O} \bar{\alpha}_{i,o}^\tau = 1 && \forall i \in I\\
&\quad \bar{\alpha}_{i,o}^\tau \bar{\mu}_{G_i^\tau} = \bar{\alpha}_{i',o}^\tau \bar{\mu}_{G_{i'}^\tau} && \forall i,\ i' \in I,\ \forall o \in O
\end{aligned}
$$

Rather than solving (CP) for additional time complexity, we modify (LR) to incorporate new information. Assuming that α is stationary, for a given assortment, if a product $i \in I$ is expected to sell twice as much, then each order type $o \in O$ fulfillable locally that contain product i should yield approximately twice as many additional fulfillable orders.

Intuitively, if the oracle predicts that a product is going to sell more, there should be more incentive to place the product in the FDC. On the contrary, if the product is predicted to sell less, there should be a penalty for placing this product in the FDC. This can easily be achieved by assigning costs to the products in the selection problem. For $i \in I$, $g_i^{\tau-1}$ is the sales observation for i over the most recent cycle as training transaction data, and $\bar{\mu}_{G_i^\tau}$ the product level demand estimate for i provided by the oracle for the upcoming cycle. We assign to product i a cost c_i^τ. If sales have

been recorded for SKU i in the training set, we set the cost to $c_i^\tau = \frac{g_i^{\tau-1}}{\bar{\mu}_{G_i^\tau}}$, otherwise we let $c_i^\tau = 1$ and we set $\bar{\mu}_{D^\tau_{(i)}} = \bar{\mu}_{G_i^\tau}$. We further assume that the order level demand estimate vector can be constructed as in part 1.4 or 1.4. With this additional cost, (LR) becomes:

$$\begin{aligned} \max \ & \sum_{o\in O} \bar{\mu}_{D_o^\tau} Y_o - \lambda \sum_{i\in I} c_i^\tau X_i && (SLR2) \\ & Y_o \le X_i && \forall o \in O,\ i \in I,\ \text{if } i \in o \\ & X_i \in \mathbb{B}, Y_o \in \mathbb{B} \end{aligned}$$

Solving the parametric minimum s-t cut problem associated with the above problem is referred to as Algorithm *PC with forecast*.

Algorithm 3: PC with product level forecast

Data: training set of several cycles of transaction data
Capacity k
product level demands estimate $\bar{\mu}_{G_i^\tau}$
Result: Assortment of products to maximize future order fulfillment from the FDC
Step 1: Ignore every order type not observed in the training set
Step 2: Compute the Monte Carlo estimate $\bar{\mu}_{D_o^\tau}$ for every order type in the training set
Step 3: **for** $i \in I$ **do**
  if $g_i^{\tau-1} > 0$ **then**
    Compute costs $c_i^\tau = \frac{g_i^{\tau-1}}{\max(\bar{\mu}_{G_i^\tau}, 1)}$
  else
    set $c_i^\tau = 1$
    set $\bar{\mu}_{D^\tau_{(i)}} = \bar{\mu}_{G_i}$ for order type $(i) \in O$
Step 4: Solve $(SLR2)$ by solving for minimum parametric s-t cut on the graph as detailed in section (1.3) with the above modifications, find the assortment for each breakpoint
Step 5: Select the assortment $\mathscr{S}_{\lambda_l}$ corresponding the breakpoint λ_l such that the capacity k_l of the assortment is the closest to the capacity constraint k of the original problem (MIP)
return $\mathscr{S}_{\lambda_l}$

Let us develop the intuition motivating the creation of this product cost. Let $\mathscr{S}_{\lambda_1} \subset \mathscr{S}_{\lambda_2} \subset \cdots \subset \mathscr{S}_{\lambda_q}$ be the set of nested solutions for $(SLR2)$ with $\lambda_1 > \lambda_2 > \cdots > \lambda_q$ being the corresponding breakpoints. Suppose there exists $i \in I$ and $l \in$

$\{1, \ldots, q\}$ such that $\mathscr{S}_{\lambda_{l+1}} \backslash \mathscr{S}_{\lambda_l} = \{i\}$. Then we have $\lambda_{l+1} = \frac{f_{\bar{\mu}_{\mathbf{D}_O^\tau}}(\mathscr{S}_{\lambda_l} \cup \{i\}) - f_{\bar{\mu}_{\mathbf{D}_O^\tau}}(\mathscr{S}_{\lambda_l})}{c_i^\tau} = \frac{\#\text{additional orders fulfillable by adding } i \text{ (w.r.t. } \bar{\mu}_{\mathbf{D}_O^\tau})}{c_i^\tau}$. λ_{l+1} is equal to the additional number of orders that can be fulfilled by adding i to $\mathscr{S}_{\lambda_l}$ given $\bar{\mu}_{\mathbf{D}_O^\tau}$ (i.e., the marginal increase in the number of locally fulfillable orders, assuming order level demand is stationary) divided by the product cost $c_i^\tau = g_i^{\tau-1}/\bar{\mu}_{G_i^\tau}$. Let $\tilde{\mu}_{D_o}^\tau = \bar{\mu}_{D_o^\tau} \times \mu_{G_i^\tau}/g_i^{\tau-1}$, note that the breakpoint value can be written: $\lambda_{l+1} = \frac{\#\text{additional orders fulfillable by adding} i \text{w.r.t. } \tilde{\mu}_{D_o}^\tau}{1}$. Because of how we defined c_i^τ, dividing the marginal increase in the number of locally fulfillable orders of i by c_i^τ re-scales it to how much we expect the marginal increase to be under the assumption that α is stationary. If furthermore, solving (CP) first and the (1.3) also places product i in a singleton layer, the cost c_i^τ does re-scale the marginal increase so that solving (*SLR2*) yields the same product score as solving (CP) first and then solving the minimum parametric s-t cut with the estimates $\bar{\mu}_{D_o}$ obtained in place of μ_{D_o}.

This result does not extend to $|\mathscr{S}_{\lambda_{l+1}} \setminus \mathscr{S}_{\lambda_l}| \geq 2$. However, if the oracle provides a close enough product level demand estimates of the real upcoming sales, we obtain very good experimental results. In practice, adding this product cost breaks the degeneracy of most minimum s-t cut solutions, thus yielding more breakpoints. We showed in section 1.3 that there are at most $|I|$ breakpoints. The more the breakpoints, the more the layers that can be of cardinality 1, thus verifying the above intuition. Furthermore, when more breakpoints are found, more nested assortments are output, thus making it easier to find a breakpoint solution whose cardinality is very close to the capacity constraint of (*MIP*). Many of the nested sets differ by a single product. This algorithm empirically outperforms the former ones in our experiments.

Heuristic for building an assortment between two breakpoints

Although it is not guaranteed that many breakpoints are found, in practice we find many granular layers. Solving the problem between two layers can be complex and it loses the nestedness property. Therefore selecting a breakpoint assignment and completing the assignment by selecting the optimal selection of SKU from the layer is not necessarily optimal. The algorithm however provides an upper bound on the optimal solution for every SKU capacity. One can complete a breakpoint assignment by solving the integer program or by using the industry standard algorithm. Consider the case where the target number of SKUs in the assortment in k. Solving the parametric minimum cut outputs 2 consecutive assortments $\mathscr{S}_1$ and $\mathscr{S}_2$ such that

$|\mathscr{S}_1| < k < |\mathscr{S}_2|$. These assortments are associated with a product score $r_1 > r_2$. By definition of a breakpoint solution and of the product score, we know that for any $\lambda \in [r_2, r_1]$, every assortment $\mathscr{S}$ verifies:

$$f_{\mu_{\mathbf{D}_O}}(\mathscr{S}) - \lambda|\mathscr{S}| \leq f_{\mu_{\mathbf{D}_O}}(\mathscr{S}_1) - \lambda|\mathscr{S}_1|$$

This is thus also true for the optimal assortment of k products $\mathscr{S}_k^*$. It follows that the absolute optimality gap is:

$$f_{\mu_{\mathbf{D}_O}}(\mathscr{S}_k^*) - f_{\mu_{\mathbf{D}_O}}(\mathscr{S}_1) \leq r_2 * (k - |\mathscr{S}_1|)$$

Once an assortment of size k has been built by completing $\mathscr{S}_1$, the optimality gap can only be smaller and can be computed.

1.5 Numerical Analysis

JD.com selected two typical RDCs to evaluate the impact of this new method. As most information about these dataset must remain confidential, we will provide order of magnitude when possible. For each pair of RDC/FDC, we selected for our experiment 15 cycles such that we can use up to 5 cycles for training and the 10 remaining cycles can all be used for evaluation purposes. Each cycle accounts for over 500 thousand SKUs and over 2 million unique order types. Some preliminary analyses show that 72% of the orders are singleton orders. Figure 1.7 shows the distribution of order size for RDC 1 with error bars of two times the standard deviation. These estimates are computed using 15 cycles as samples. Because about 30% of the orders are not singletons, it is extremely important for e-retailers to consider how products are purchased together to optimize product placement in warehouses.

Because the demand is not always comparable from one cycle to another and as we study FDCs with different capacities, an invariant metric is needed to compare the results of the algorithms on different datasets. The main metric we consider is the order fulfillment ratio, which is the fraction of all orders routed to the FDC and can be fulfilled from the FDC locally. The objective is equivalent to selecting a placement of products that maximizes the fulfillment ratio under capacity constraints.

As forecasting is a challenging problem, we use several oracles providing product level demands estimates with different levels of error. The output estimate is the demand realization of next cycle with added Gaussian noise. We want the predictor to be centered on the demand realization and have a level of error proportional to the demand realization. Consequently, we re-scale the noise by the demand realization. As demands cannot be negative, we truncate the estimate. For $i \in I$ and the

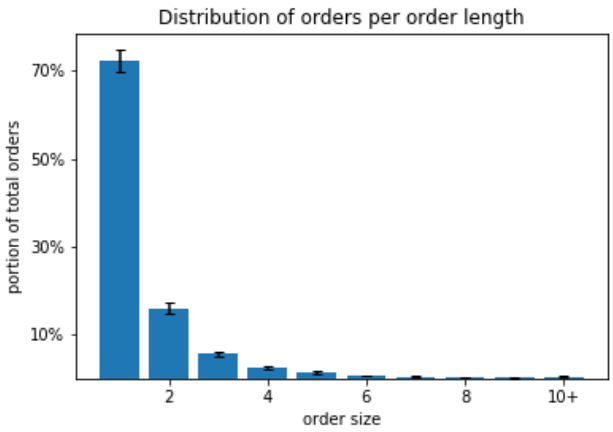


Figure 1.7: Distribution of order size at RDC 1

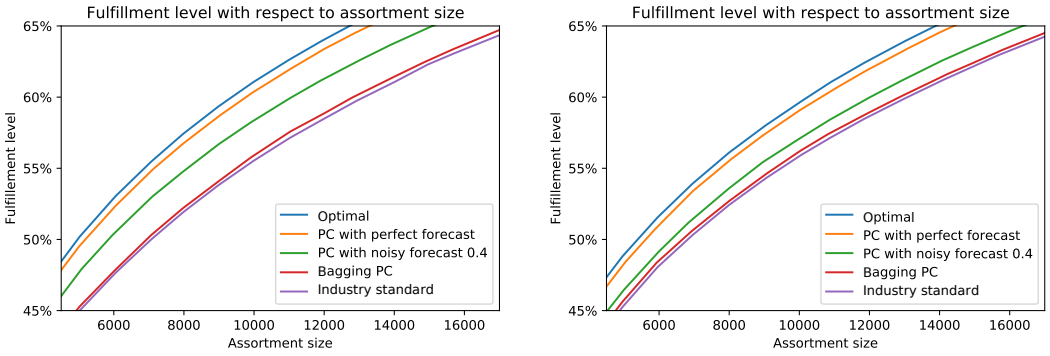
upcoming cycle $\tau \in \{1, \ldots, \mathscr{T}\}$, the oracle with parameter σ outputs $\bar{\mu}_{G_i^\tau} = (1+\epsilon)^+ g_i^\tau$ with $\epsilon \sim \mathscr{N}(0, \sigma^2)$. For instance, the algorithm *PC with noisy forecast 0.4* has $\sigma = 0.4$ resulting in $\bar{\mu}_{G_i^\tau} \in [0.5g_i^\tau,\ 1.5g_i^\tau]$ with probability 79% and $\bar{\mu}_{G_i^\tau} > 2g_i^\tau$ with probability 0.6% Letting σ take different values results in oracles with different levels of error with respect to the demand realization. Using this set of oracles, we can compare the performances of our algorithms given forecasts with increasing average errors. Using $\sigma = 0$ corresponds to using perfect knowledge of the product demands. Using this, e-retailers can choose whether or not to use their own forecast data based on how confident they are about the forecasting error.

We use two comparisons to assess the quality of our results. For the lower bound on the performance, we use an industry standard algorithm that ranks the products based on how often they sold (see Appendix A.2 for the algorithm and its theoretical performance). For the upper bound on the performance, we call *optimal* the solution of *Stationary PC* on the realized vector of demand. This is equivalent to solving the minimum parametric cut a posteriori. We only benchmark against breakpoints of the solution on the test data so that we compare them to an optimal decision for a given FDC capacity.

We compare our results under different capacity settings to show the robustness of our algorithms. In Figure 1.8, each curve is a piecewise linear function. We benchmark the results of our algorithms to breakpoint solutions on the test set and for better visualization, we connect the dots.

Figure 1.8 shows that about a 1% improvement over the industry standard algorithm can be achieved using algorithm *Bagging PC*, while algorithm *PC with forecast* can achieve values that are less than 2% away from the optimal assortment depend-

ing on the quality of the forecast. This improvement for a company such as JD.com accounts for a delayed order reduction of over 100 million annually.



(a) RDC 1 third cycle

(b) RDC 1 third cycle

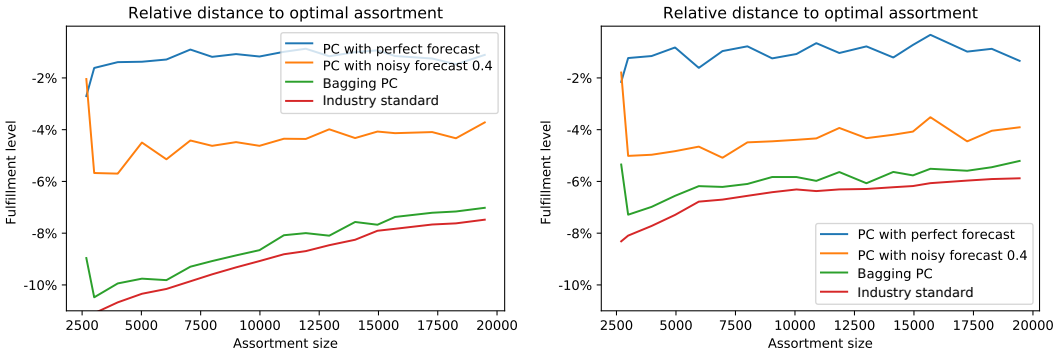
Figure 1.8: Comparison of fulfillment rates on different RDC/FDC pairs for different algorithms

It appears from Figure 1.9 that the optimality gap of solutions obtained by the proposed methods varies with respect to the warehouse. Though algorithm *Bagging PC* has an optimality gap of around 8 - 10% depending on the capacity, the performance of algorithm *PC with forecast* is quite impressive with an optimality gap that is as low as 2% perfect product level demands and Monte Carlo order level demand estimates. We note a clear improvement over the performance of the industry standard algorithm. However, there is still room for better placement optimizations if no good product level forecast is available.

The experiments ran on an Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz. For RDC1 the training transaction dataset covers two cycles of operations: 3,833,283 orders out of which 927,596 are unique order types and 265,967 are unique SKUs. On this real size dataset, our algorithm, runs in 4 to 6 minutes.

Sensitivity Analysis on order length distribution

The algorithms we presented above are designed for capturing the co-purchase behavior of products and reduce order split. Although less than 70% of JD.com orders are singleton orders, the distribution of order length can vary a lot in the e-retail industry. In this Subsection, we study the impact of order length distribution on the performance of *Stationary PC*. From JD.com sales data set, it appears that the order length follows an exponential distribution. Consequently, we analyze the per-



(a) RDC 1 third cycle

(b) RDC 2 second cycle

Figure 1.9: Comparison of optimality gaps on an RDC-FDC pair for different algorithms

formance improvement of *Stationary PC* over the industry standard for different parameters of exponential distribution of order length.

To create synthetic data sets to experiment upon, we use importance sampling on JD.com's data to obtain realistic sales data with a different order length distribution. Let g be the empirical distribution of order length on the sales data set, $g(l)$ returns the observed probability of an order to be of size l. Let f be the targeted distribution of order length. Instead of sampling orders with probability proportional to their demand, using importance sampling, an order o with demand d_o and length l is sampled with probability proportional to $d_o * f(l)/g(l)$.

Letting the parameter of the exponential distribution vary from 0.2 to 1.2, 10 sampled data sets of 2 million orders are created for each parameter. On each sampled data set, the size of the assortment is selected as a percentage of the total number of SKUs needed to fulfill all the 2 million orders. Figure 1.10 shows the performance of the algorithm for assortments ranging from 3% to 7% of the total number of SKUs. As the number of long order increases, it is harder to fulfill many orders with a small proportion of the SKUs. Note that this range of assortment results in fulfillment ratio ranging from 27% to 43% for Exponential(0.2) and from 51% to 64% for Exponential(1.2).

This analysis provides insight on what business will benefit the most from our algorithms. It shows e-retailers with a heavy distribution of singleton orders, like JD.com, can get a few percentage points of performance from using our algorithm and that the performance gain quickly increases when the distribution is flatter. This algorithm should provide maximum performance gain for e-retailers offering a sub-

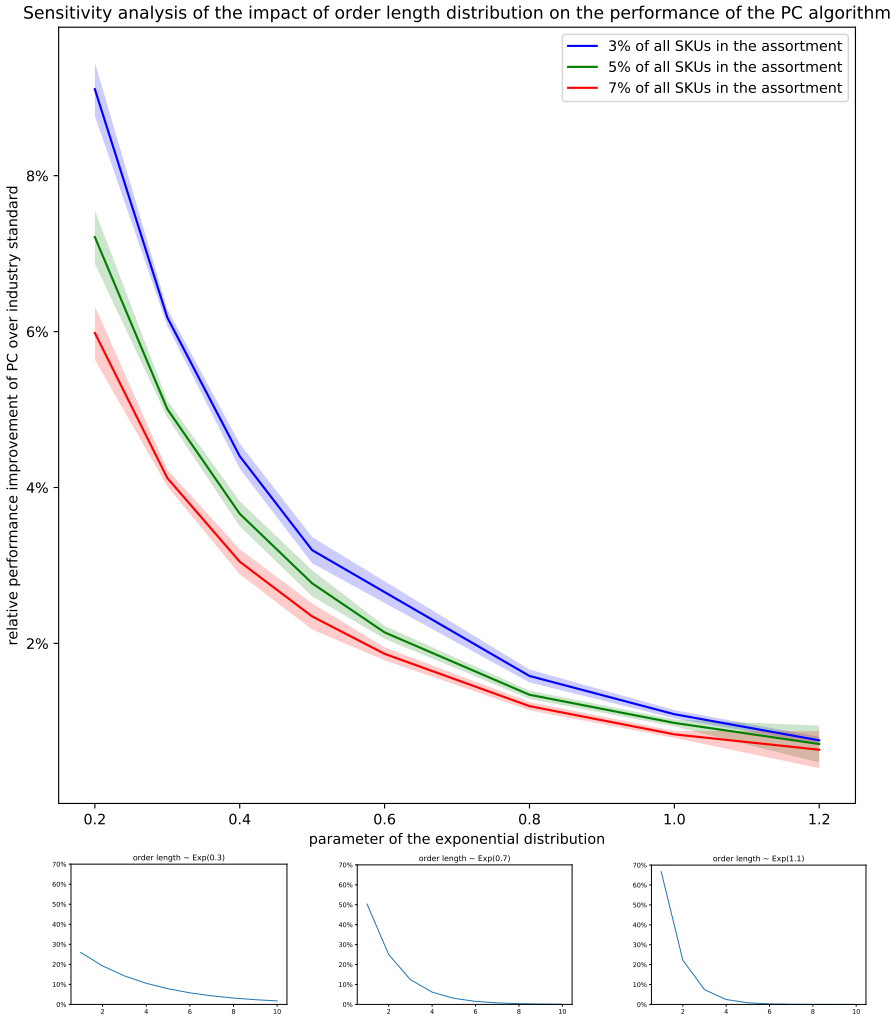


Figure 1.10: Performance of *Stationary PC* for different distribution of order lengths

scribe and save service. Such a service offers discount to a customer ordering several products together regularly. This results in stationary demand and a distribution of order length less heavy on singleton orders. Such a service verifies the assumptions for using *Stationary PC* and has an order length distribution suited for high performance increase.

Comparative analysis of industry standard ranking and our product score

With the algorithm, we proposed a product score which captures the co-purchase behavior of SKUs. In Section 1.3, we showed that this score captures the marginal benefit of adding a layer of SKUs to the assortment. The industry standard algorithm (see Appendix A.2) builds a score for each SKU based on the ratio of demand for the product by size of the orders it is purchased in. The two scores are very much related to the SKU sales, we thus conduct a comparative analysis to understand how our product score differs from the industry ranking.

Figure 1.11 shows a strong correlation between the industry standard ranking and the product score. This is expected as each algorithm is based on the amount of sales that an SKU accounts for. This result also clarifies why the performance of the two algorithms differs only by a few percentage points. On Figure 1.11, two red dashed lines separate the main cloud of points from points that either received high industry rank (in the upper part of the figure) or a low industry rank (in the lower part of the figure) compared to their product score. This few points account for the gap in fulfillment, consequently we focus our analysis on the corresponding SKUs. Note that the scores on Figure 1.11 have been rescaled to keep all information about JD.com sales confidential.

We already explained that *Stationary PC* solves the problem to optimality under some assumptions. Therefore, in this Subsection, we are trying to understand what type of product get misclassified by the industry standard algorithm. Specifically, it is important to understand what type of demand pattern leads the industry standard algorithm to overestimate or underestimate the value of a given SKU in the assortment.

By construction, the industry standard gives a heavy weight to an SKU with very high demand and lowers it if it is often in long orders. However, the algorithm only considers the length of orders without considering the diversity of product a given SKU is ordered with. For similar demand and order length, an SKU that is always purchased with a small group of other SKUs is preferable to an SKU that is randomly purchased with any other SKU. Consequently, we expect the industry standard algorithm to overestimate the importance of SKUs falling in shorter orders on average even if they are often ordered with a wide variety of other SKUs.

To verify the above hypothesis, we compare a few metrics. As we expect the industry standard to give too much importance to the average order length, we measure the empirical cumulative density function of order length associated with an SKU. Last, we know that the product score we designed capture the co-purchase behavior. It captures the marginal benefit of adding a product to the closest break-

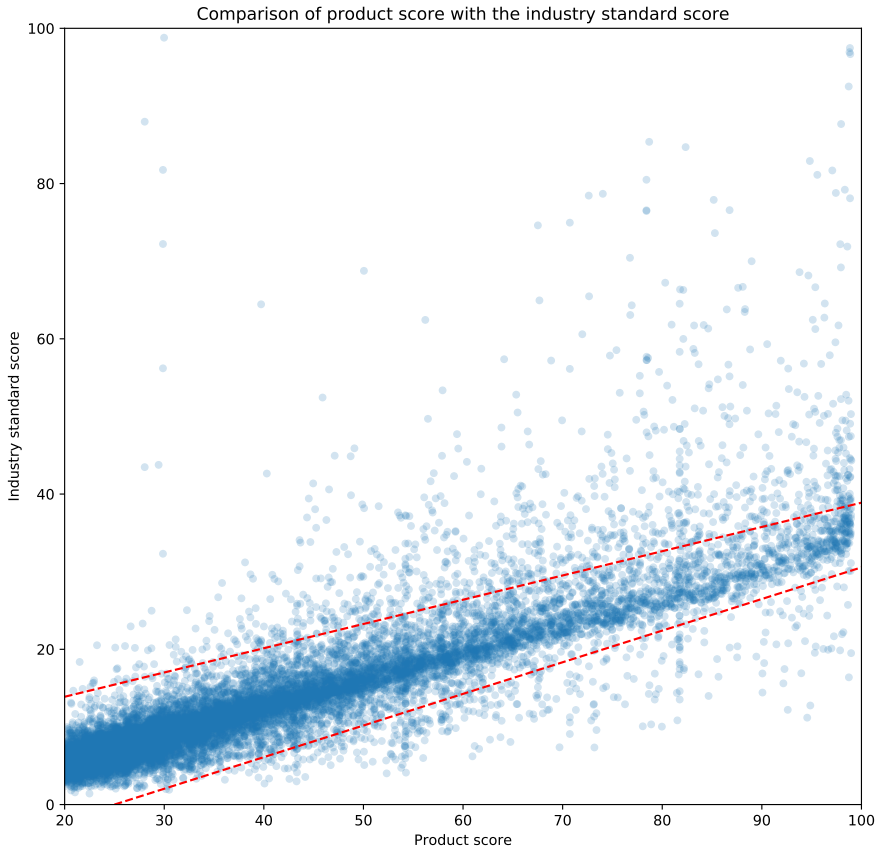


Figure 1.11: Correlation between industry standard ranking and product score

point assortment. Since this is our best metric of how important a product is in the optimal assortment, we measure the empirical distribution of product score of the co-purchased SKUs weighted by how often they are sold with the SKU of interest.

Figure 1.12 shows the comparison between two SKUs, one overestimated by the industry standard algorithm and the other underestimated. For this example although they seem to have similar distribution of order length, the SKU that is overestimated by the industry standard algorithm has a distribution less concentrated on the mean. The two SKU have a very different empirical distribution of product scores. On Figure 1.12, we show on the left a product which score is under-estimated

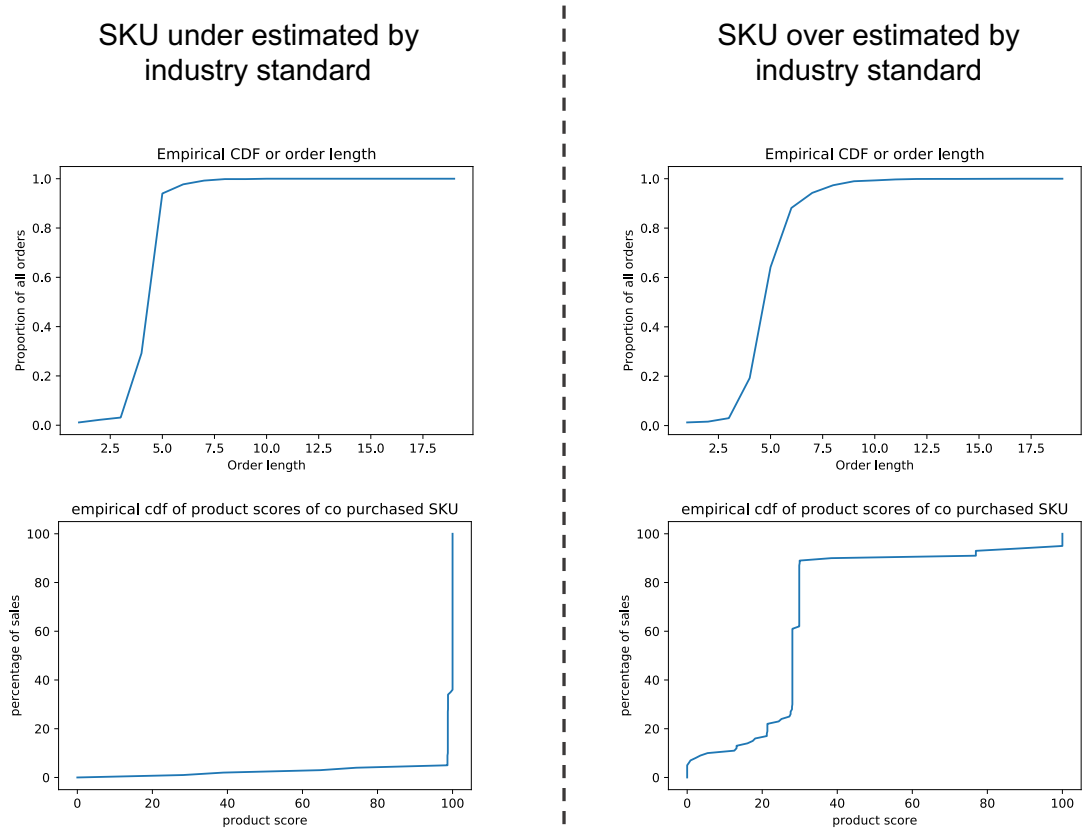
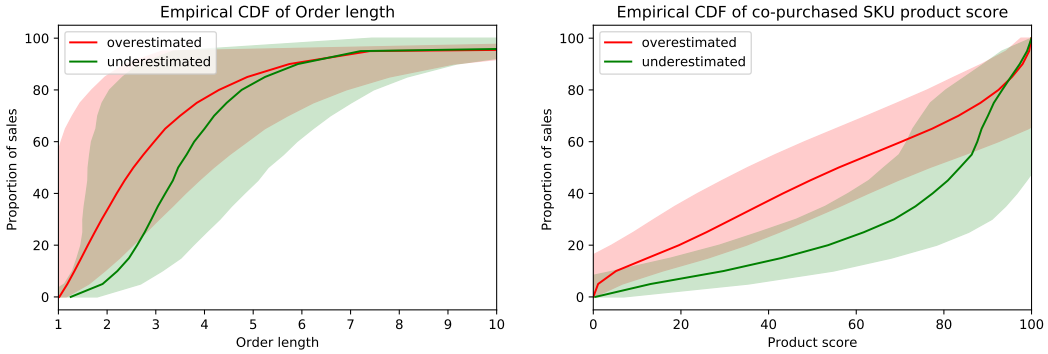


Figure 1.12: Comparison of two SKUs that the industry standard algorithm fails to score correctly

by the industry standard algorithm. We note that co-purchased items accounting for 90% of the sales have very high product score. Consequently, the co-purchased items are among the most important to be placed in the assortment, therefore, although this SKU is purchased on average in long orders, the items it is purchased with are in the optimal assortment thus this SKU has a higher score. On the contrary, the SKU on the right is assigned an industry standard score that over-estimates its value in the assortment. We note that only co-purchased products accounting for 10% of the sales have high product score. Consequently most of the co-purchased products needed to fulfill the demand for the SKU of interest are not placed in a very constrained assortment. Thus this SKU receives a lower score.

To validate these observations, we aggregate the resulting curves from all SKUs which scores have been over-estimated or under-estimated by the industry standard algorithm. We first note that the industry standard algorithm ranks properly most

SKUs with heavy singleton demand. As shown on Figure 1.13a, very few of these SKUs are often sold as singleton. We explained above that the industry standard algorithm attributes a score inversely proportional to the average order length of an SKU. Figure 1.13a also shows that the industry standard algorithm tends to overestimate the score of SKUs with order length distribution shifted to shorter orders and underestimate the score of SKUs with order length distribution shifted to longer orders. We note that the items which score is under-estimated by the industry standard have a low singleton demand. Finally, and most importantly, Figure 1.13b shows the shortcoming of the industry standard in only considering the order length and ignoring what are the co-purchased SKUs. Figure 1.13b shows that a large portion of the sales of SKUs which scores are under-estimated by the industry standard algorithm are attached to co-purchased products with high product score. On the contrary, SKUs which scores are over estimated tend to have a uniform empirical CDF resulting in fewer orders containing highly scored products.



(a) Comparison of empirical CDF of order length

(b) Comparison of empirical CDF of product scores

Figure 1.13: Comparison of empirical CDF for overestimated and underestimated ranking of SKUs by industry standard

From this analysis, we can conclude that for distribution of order length similar to JD.com, the industry standard algorithm tends to attribute a similar ranking to SKUs than our algorithm. Our algorithm draws most of its benefits from flatter distribution of order length and from SKUs with high demand but low singleton sales. Our algorithm can manage that last category of SKUs much better than the industry standard.

1.6 Discussion

The problem of selecting the best assignment of SKUs to be placed in a FDC is a multi-stage problem. Swapping an SKU for another between two period comes at a cost that has been ignored until now. Because the number of order types grow exponentially with the number of SKU, e-retailer don't try to forecast demand at the order level. Estimating demand at a product level is very common, it is a very complex problem and the error of the prediction grows quickly with how far in the future the model is predicting. Consequently, we assumed that the order level demand was stationary and considered the impact of product level demand forecast in our model. For all these reasons, although the problem is multistage by design, we solved it with a myopic approach. This approach tends to be very good from one cycle to the next, the seasonality of demand can be seen when evaluating the performance of an assignment on the demand a few cycles later. Figure 1.14 shows that a large portion of the SKUs assigned for a given cycle remain in the optimal assignment for many cycle while 10% to 20% of SKUs are removed after the first cycle due to seasonality.

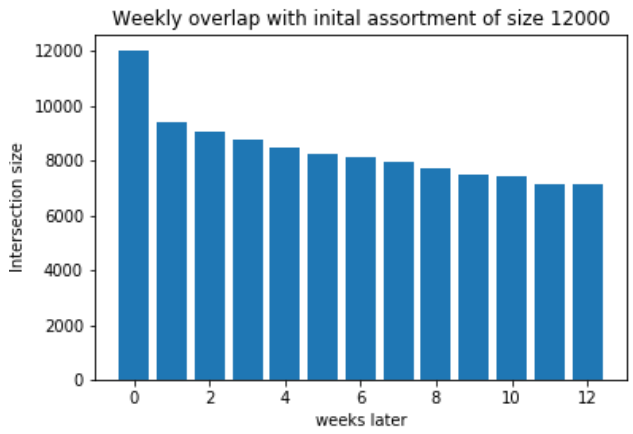


Figure 1.14: Size of the intersection of consecutive placements of size 12,000

Depending on the business seasonality of demand may become a bigger issue. For businesses with large seasonality in demand from one cycle to the next, this algorithm can still be used although two problems will be solved. First step solve the problem on a very large number of cycle to capture the steady order level demand. Doing so we obtain breakpoints, layers and product scores for the steady demand. Then by solving the problem on less cycles, the seasonality and the trend in the demand can be captured. Once an assignment is selected using the steady order level demand, a

cost γ can be assigned to removing an SKU from the steady assignment by adding an arc from each SKU node corresponding to selected SKU to the sink with capacity *gamma*. This cost can also be a linear function of the product level demand for the SKU. The cost needs to be comparable to the optimized unit, that is either the number of locally fulfilled orders or the profit from locally fulfilled orders.

1.7 Conclusion

Focusing on the industry problem of the largest Chinese e-retailer, we presented in this Chapter novel algorithms to optimize the product assignment in a delivery network. For the setting of JD.com with regional and local DCs structured as a tree, our algorithm outperformed significantly the industry standard that was formerly used by the company.

Given the resources of the company and the predictability of the product level demands, we presented algorithms that outperform the industry standard while not increasing the running time excessively. These algorithms highlight the products responsible for bundle sales and allow for a better inventory placement. Thanks to the product score, the most important products for local fulfillment can be easily identified thus allowing e-retailer or other business to investigate the demand pattern of such product more in depth.

In practice, though RDC can be very far from an FDC customer area, neighboring FDCs can be much closer. Therefore, they may be able to fulfill some of the neighboring customer demands in a timely manner. With JD.com we also developed a similar algorithm for a more complex network to investigate the impact of flexibility in the network. This work will be presented in Chapter 2. By adding a few arcs in the warehouse network and proposing an inventory placement algorithm we aim at getting close to the performance of a fully connected network. The algorithm relying on a parametric cut minimization cannot work on this generalized network. It is, however, possible to design intuitive extensions such as solving FDC inventory placement problems iteratively until convergence. Furthermore under additional assumptions on the delivery network structure, the problem becomes tractable.

Chapter 2

Warehouse Assortment Planning with Fulfillment Flexibility

In Chapter 1, we have studied the problem of maximizing local fulfillment for each pair of RDC and FDC independently. In this chapter, we study the impact of flexibility in the fulfillment network. At the high level, adding flexibility to a delivery network introduces a trade-off between the in-stock rate and fulfillment cost. A fully connected network will reach a maximum in-stock rate, whereas the low utilization of many links in the network will result in a high per package fulfillment cost. On the contrary, an inflexible network will have a much lower in-stock rate owing to capacity constraints and will achieve much lower fulfillment cost. Additionally, a fully connected network may not be realistic. Each DC has a limited number of docks, thereby limiting the number of outbound routes. Therefore, each node in the network has a maximum number of outgoing arcs. To further increase the flexibility of the network, sortation centers, which are additional nodes in the network, are needed as intermediaries so that each DC can reach each region.

In China, shipping by air is considerably more expensive than in the US. This led our business partner, JD.com, to originally invest in an inflexible delivery network. Our work focuses on optimizing the in-stock rate of groups of SKUs, i.e., orders, by introducing flexibility. Owing to their current delivery network design, the problem we consider excludes sortation centers (JD.com's sortation centers are not set up to handle cross-front distribution center (cross-FDC) shipping) and admits limited flexibility. We solve the product placement problem and illustrate the performance of our algorithms on real-world datasets.

Although having a fully connected fulfillment network would definitely reduce the number of split orders, it would not ensure fast delivery, as packages may have to travel long distances to reach the customer. Approximately one third of the orders

placed on JD.com consist of multiple products. When an SKU placed in a customer order is missing from the dedicated FDC, the e-retailer can still fulfill this order from another DC, if it holds every SKU placed in the order. This fulfillment decision prevents the order from being split into multiple packages, thereby saving on packing costs and achieving higher customer satisfaction. When the closest DC cannot fulfill an order and it is routed to a more distant DC, this is known as a *spillover*.

Historically, JD.com has restricted spillover to only be from the RDC to the FDC (see Figure 1.1). Adding flexibility, i.e., spillover from one FDC to another, to such a constrained network would allow risk pooling and improve the overall performance of the delivery network. Consequently, JD.com is investigating how much flexibility is needed and how to manage the inventory in a flexible network. This work, along with others published with the company ([11]), highlights the benefit of such a network and how to manage it. We assume that an order placed in a customer area can be fulfilled by neighboring FDCs instead of the closest one if it avoids splitting the order or routing it to the closest RDC. Spillover is only considered if the order can reach the customer faster than if it were shipped from the RDC. To illustrate the benefit of allowing flexibility in a fulfillment network, we use real-world data from JD.com and test the impact of various degrees of flexibility in the network.

Given an inventory allocation, deciding which DC should fulfill which order leads to a very complex and dynamic program, which is intractable owing to the curse of dimensionality. We relax this by assuming that if a product is placed at the FDC level, it will be replenished often enough to never stock out. Although this assumption simplifies the problem, determining the constrained SKU allocation of each FDC is a combinatorial optimization that is difficult to solve, owing to the large scale of the problem.

These challenges motivate the construction of a flexible delivery network that reduces the cost of fulfilling demand without slowing down the delivery speed, in which a product assortment allocation can be determined efficiently. The model proposed in this study was developed in close collaboration with expert practitioners to capture the most relevant aspects of network flexibility and split orders. Although the model relies on simplifying assumptions, experts from our business partner consider it to be sufficiently detailed to guide strategic decision-making about network flexibility and product assortment allocation.

The rest of the paper is organized as follows. In Section 2.1, we review the literature about network flexibility in e-retailing. In Section 2.2, we introduce a mathematical model for the SKU allocation problem for a single RDC-FDC pair and for a flexible fulfillment network. In Section 2.3, we prove that selecting a subset of SKUs for each FDC to minimize fulfillment costs can be solved efficiently for specific configurations of the flexible fulfillment network. In Section 2.4, we

study the impact of gradually adding flexibility to a fulfillment network using real-world transaction data from our business partner. Finally, in Section 2.5, we present some promising results by applying this method to our business partner's full-scale transaction dataset.

2.1 Literature Review

Our work is related to an effort to reduce the rate of split orders. Split orders occur when a customer order is fulfilled from multiple DCs. Opportunities to reduce customer order splitting exist at different echelons of the supply chain. Starting with choosing a product assortment to place in a DC, the problem has been proven to be difficult for a single warehouse with capacity constraints and we provided an algorithm to solve it efficiently under reasonable assumptions in Chapter 1. In the case where several facilities can hold a subset of SKUs, determining the assortment allocation to minimize the total number of facilities that must be accessed to fulfill every order is tackled with heuristics and neighborhood search [6], [21].

The design of flexible fulfillment networks is motivated by the concept of process flexibility [19]. In their paper, they focus on the question of how much flexibility is needed and show that, under realistic assumptions, a well-designed limited-flexibility system can yield most of the benefits of a fully flexible system. In this work, we only review flexibility studies related to e-retailing. [2] studied the resource allocation problem with long-chain design and online stochastic requests. [32] extended their research and provided a method for designing a flexible network when the number of facilities is much smaller than the number of customer areas. [11] further refined their work by considering the spillover cost when designing a fulfillment policy. These studies provide methods for designing a fulfillment strategy; however, they either assume that the initial inventory is given, or compute the initial inventory of a small number of SKUs (tens of thousands). Our work differs by focusing on inventory placement, i.e., what SKU to hold at what DC, in a flexible delivery network.

The inventory placement problem for e-retailers, along with the replenishment policy, has been studied by [7]. The study considers the general problem of choosing what SKUs to place in what DCs to minimize the total cost. The literature focusing on fulfillment policy and inventory placement considers each SKU individually, thus simplifying the problem. Our paper differs from existing studies, as it focuses on designing product assortments to be placed in distribution centers to minimize the shipping costs while limiting order splitting. This research is inspired by a preliminary data analysis, which shows that nearly one third of the orders placed on our business partner's website are bundles of several products.

2.2 Problem statement

Flexibility provides the ability to fulfill an order from different neighboring FDCs. A customer order that is fulfilled from an FDC is *efficiently fulfilled*, as oppose to an order fulfilled from the RDC. Spillover enables this group of FDCs to cover a wider set of order types, i.e. sets of SKUs, thus increasing the number of efficiently fulfilled orders. In this section, we explore the modeling of flexibility in a fulfillment network.

Definition and notation

Consider a set of FDCs, $j \in J$, associated with a single RDC. The RDC holds every SKU and can fulfill any order. FDC $j \in J$ is dedicated to a single customer area. Let I be the set of SKUs of the company.

Owing to the manner in which products are stored at the FDC and RDC, the time needed to load and unload the daily shipment from the RDC to the FDC is proportional to the number of SKUs that are shipped. Thus, the number of SKUs that FDC j can hold is upper-bounded by its capacity and the time consumption of the daily replenishment. Let k_j denote the maximum number of SKUs that can be placed in the FDC j. When an order is placed at FDC j, either all products purchased in the order are located in FDC j, or it is routed to another warehouse.

An SKU combination is called an *order type* and is denoted $o \subseteq I$. We denote by O be the set of order types, i.e., the set of product combinations. For order type $o \in O$ and FDC $j \in J$, the random variable D_o^j models the demand for order type o at FDC j with mean $\mu(D_o^j)$.

We assume that the RDC virtually holds every SKUs the company sells. An order that is routed to the RDC yields a penalty of 1, i.e., the maximum penalty since the RDC can fulfill any order type. Under the flexible setting, a neighboring FDC j' can fulfill the order placed at j for a penalty of $p_{j',j} \in [0,\ 1]$. Note that assuming that no spillover is allowed from j' to j is equivalent to setting the penalty $p_{j',j} = 1$. By definition $p_{j,\ j} = 0$.

For each product $i \in I$ and FDC $j \in J$, we associate a binary decision variable X_i^j that models whether or not product i is located at FDC j. We let X_i^j be 1 if product i is located at FDC j and 0 otherwise. The set of products located at FDC j is called an *assortment* and denoted $\mathscr{S}^j = \{i \in I | X_i^j = 1\}$.

For each order type $o \in O$, and for each FDC $j \in J$, we associate a binary decision variable Y_o^j that models whether or not order type o can be fulfilled from FDC j. This variable is set to 1 if and only if FDC j can fulfill order o.

The goal is to select an assortment $\mathscr{S}^j$ for each FDC to minimize the total penalty. Note that if for all $j, j' \in J$ such that $j' \neq j$, we have $p_{j',j} = 1$ this problem

is identical to the single RDC/FDC pair problem of Chapter 1.

Problem formulation

In this subsection, we derive the mathematical formulation of the problem. We formulate it as a maximization problem.

An order type $o \in O$ placed at FDC $j \in J$ is fulfilled by the FDC j' which minimizes the shipping penalty to j, i.e. $p_{j',j}$, subject to the constraint that FDC j' can fulfill order type o. As shipping from the RDC yields a penalty of 1, we assume for simplification that if no FDC can ship an order for a penalty lower than 1, the order is shipped from the RDC.

If FDC j' can fulfill order type o, i.e. $Y_o^{j'} = 1$, when placed at FDC j, it allows to reduce the penalty from 1, i.e. shipping from RDC, to $p_{j',j}$, i.e. shipping from j'. We denote W_o^j the decision variable such that the penalty associated with an order type o placed at FDC j is $\mu(D_o^j)(1 - W_o^j)$. Because the order type is fulfilled by the FDC with minimum shipping penalty, $W_o^j = \max_{j' \in J}\{Y_o^{j'}(1 - p_{j',j})\}$.

A given FDC j cannot fulfill an order type o from its assortment $\mathscr{S}^j$ if any product i included in the order type o, is missing from the assortment. Therefore, Y_o^j must satisfy the constraint $Y_o^j \leq X_i^j,\ \forall o \in O,\ i \in I,\ j \in J,\ i \in o$.

Minimizing the sum of penalties over all order types results in the following objective function:

$$\min_{X_i^j, Y_o^j, W_o^j} \quad \sum_{j \in J,\ o \in O} \mu(D_o^j)(1 - W_o^j)$$
$$\max_{X_i^j, Y_o^j, W_o^j} \quad \sum_{j \in J,\ o \in O} \mu(D_o^j) W_o^j$$

We can formulate the flexible problem as a mixed integer program (f-MIP):

$$\max_{X_i^j, Y_o^j, W_o^j} \sum_{j, \in J,\ o \in O} \mu(D_o^j) W_o^j \qquad \text{(f-MIP)}$$

$$s.t.\ W_o^j = \max_{j' \in J}\{Y_o^{j'}(1 - p_{j',j})\} \qquad \forall o \in O, j \in J \quad (2.1)$$

$$\sum_{i \in I} X_i^j \leq k_j \qquad \forall j \in J \quad (2.2)$$

$$Y_o^j \leq X_i^j \qquad \forall o \in O,\ i \in I,\ j \in J,\ j' \in J, \text{ if } i \in o \quad (2.3)$$

$$X_i^j, Y_o^j \in \{0, 1\} \qquad \forall i \in I,\ j \in J, o \in O \quad (2.4)$$

Similarly to the single RDC/FDC formulation (*MIP*) from Chapter 1, the f-MIP is intractable to solve. We have shown in Chapter 1 that the Lagrangian relaxation

of *MIP* is efficiently solvable by reduction to a minimum s–t cut. The same method cannot be used for f-MIP. With additional assumptions on the topology of the fulfillment network, we show in Section 2.3 that the Lagrangian relaxation of f-MIP can be evaluated by solving a minimum s-t cut problem.

2.3 Solving f-MIP for closed chain fulfillment network

It is well known in the process flexibility literature (e.g., [19]) that limited flexibility configured in the right way can yield most of the benefit of total flexibility. In their paper, they introduce the notion of *chain* and *closed chain* where one flexible link is added per product–plant pair.

In the context of the flexible assortment assignment, we call $\mathscr{N} = (\mathscr{V}, \mathscr{A})$ the fulfillment network such that each pair of FDC-customer is modelled by a node in $\mathscr{V}$. We denote by $v_j \in \mathscr{V}$ the node associated with the FDC-customer pair of FDC j. When spillover is allowed from j to j', i.e. $p_{j',j} < 1$, an arc $(v_j, v_{j'})$, is added to the fulfillment network.

We say there is no flexibility when each FDC is responsible for shipping to a single customer areas. No spillover from other FDC is allowed, i.e. $\mathscr{A} = \emptyset$, therefore, if an order type can not be fulfilled from an FDC, it is routed to the RDC. This configuration is illustrated on Subfigure 2.1a.

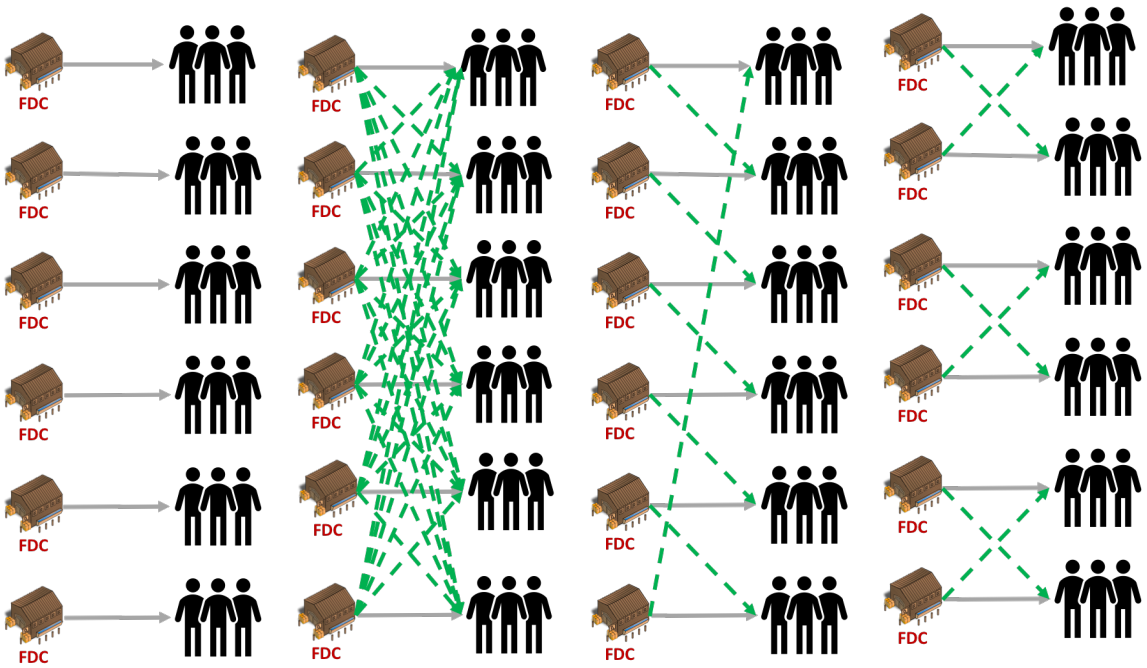
The fulfillment network is *fully connected* if each FDC can ship to every other FDC with a penalty lower than 1, i.e. the penalty for shipping from the RDC. $\mathscr{N}$ is fully connected, $\mathscr{A} = \mathscr{V}^2$. This configuration is illustrated on Subfigure 2.1b.

For any node v_j in the fulfillment network, if the count of incoming arcs $(\cdot, v_j) \in \mathscr{A}$, i.e. in degree, and the count of outgoing arcs $(v_j, \cdot) \in \mathscr{A}$, i.e. out degree, are set to 1, the fulfillment network is made of cycles. Additionally, if $\mathscr{N}$ is connected, there exists a single cycle in the network, this configuration is called a *closed chain*. Such configuration is illustrated on Subfigure 2.1c

In this Section, we explore the Lagrangian relaxation of the capacity constraint for the closed chain configuration. We show that the Lagrangian dual function can be efficiently solved as a minimum cut and we provide an algorithm to optimize the Lagrangian function.

(f-MIP) reformulation

Let us simplify the mathematical formulation (f-MIP) for a flexible fulfillment network structured as a closed chain. We will show that for any number of chains of



(a) No flexibility (b) Full flexibility (c) Closed chain (d) Three occurrences of two-chains

Figure 2.1: Flexibility comparison

even number of links, the Lagrangian relaxation of the problem can be solved efficiently. In this flexible fulfillment network, we call $n(j)$ the FDC such that spillover is allowed from $n(j)$ to j. In a closed chain configuration, such FDC is unique. Under this setting, each order o placed at FDC j can only be fulfilled either by the dedicated FDC j or by the neighboring FDC in the chain $n(j)$. Therefore, as each FDC j has a unique neighbor FDC, we can denote the penalty of fulfilling an order from the neighboring FDC as $p_j = p_{n(j),j} \in [0, 1]$.

We called W_o^j the decision variable such that the penalty associated with an order type o placed at FDC j is $\mu(D_o^j)(1 - W_o^j)$. Only FDC j and its neighbor FDC j' can fulfill this order type at this location. Note that $\mu(D_o^j) \geq \mu(D_o^j)(1 - p_j)$. Therefore, if one of the two FDCs can fulfill order type o, the reward is at least $\mu(D_o^j)(1-p_j)$. If the dedicated FDC j can fulfill the order, then the reward increases by an additional $\mu(D_o^j)p_j$. We let $Z_o^{j,n(j)}$ be a binary variable that is 1 only if neither the dedicated FDC j nor the neighboring FDC $n(j)$ can fulfill the order. That is,

$Z_o^{j,n(j)} + Y_o^j + Y_o^{n(j)} \geq 1$. The objective of f-MIP becomes:

$$\max_{X_i^j, Y_o^j, Z_o^{j,n(j)}} \sum_{j \in J,\ o \in O} \mu(D_o^j) p_j Y_o^j + \mu(D_o^j)(1-p_j)(1 - Z_o^{j,n(j)})$$
$$s.t.\ Z_o^{j,n(j)} + Y_o^j + Y_o^{n(j)} \geq 1 \qquad \forall o \in O, j \in J$$

We will write this constraint as $Z_o^{j,n(j)} + Y_o^j - (1 - Y_o^{n(j)}) \geq 0$ to make more apparent the similarity with the mathematical formulation of the minimum s-t cut. The mathematical formulation becomes:

$$\min_{X_i^j, Y_o^j, Z_o^{j,n(j)}} \sum_{j \in J,\ o \in O} \mu(D_o^j) p_j (1 - Y_o^j) + \mu(D_o^j)(1-p_j) Z_o^{j,n(j)}$$

$$s.t.\ Z_o^{j,n(j)} + Y_o^j - (1 - Y_o^{n(j)}) \geq 0 \qquad \forall j \in J, o \in O \qquad (2.5)$$
$$\sum_{i \in I} X_i^j \leq k_j \qquad \forall j \in J \qquad (2.6)$$
$$Y_o^j \leq X_i^j \qquad \forall o \in O,\ i \in I,\ j \in J, \text{ if } i \in o \qquad (2.7)$$
$$Y_o^j,\ X_i^j,\ Z_o^{j,n(j)} \in \{0,1\} \qquad \forall o \in O,\ i \in I,\ j \in J \qquad (2.8)$$

For this new formulation, the Lagrangian relaxation only introduces $|J|$ Lagrangian parameters denoted as λ_j for $j \in J$.

Evaluating the Lagrangian dual function by solving a minimum cut

The Lagrangian dual function $g(\boldsymbol{\lambda} = \{\lambda_j\}_{j \in J})$ is formulated as follows:

$$g(\boldsymbol{\lambda}) = \min_{X_i^j, Y_o^j, Z_o^{j,n(j)}} \sum_{j \in J,\ o \in O} \mu(D_o^j) p_j (1 - Y_o^j) + \mu(D_o^j)(1-p_j) Z_o^{j,n(j)} + \sum_{i \in I, j \in J} \lambda_j X_i^j - \sum_{j \in J} \lambda_j k_j$$

$$s.t.\ Z_o^{j,n(j)} + Y_o^j - (1 - Y_o^{n(j)}) \geq 0 \qquad \forall j \in J, o \in O$$
$$Y_o^j \leq X_i^j \qquad \forall o \in O,\ i \in I,\ j \in J, \text{ if } i \in o$$
$$Y_o^j,\ X_i^j,\ Z_o^{j,n(j)} \in \{0,1\} \qquad \forall o \in O,\ i \in I,\ j \in J$$

Note that $\sum_{j \in J} \lambda_j k_j$ is not a function of any of the decisions variable, hence it can be ignored when searching for optimal solutions.

Assuming that there is an even number of FDCs in the closed chain, this reduces to solving a minimum s-t cut problem. Actually, we will prove a slightly more general result: assuming that a customer node can receive orders from at most two FDCs and that there exist two groups of FDCs such that no two FDCs from the same group can ship to the same customer node, the Lagrangian relaxation of the flexible FDC assortment problem can be solved efficiently as a minimum s–t cut. In the case of a closed chain of an even number of FDCs, the FDCs can trivially be partitioned into two such groups by assigning a randomly chosen FDC to the first one, and then iteratively assigning the neighbor of the most recently assigned FDC to the other group.

Recall that the minimum s-t cut problem in a graph $G = (V, A)$, where V is the set of nodes and A is the set of arcs, can be formulated as follows:

- $d_{u,v} \geq 0 \ \forall (u, v) \in A$ a variable per arc
- $z_v \ \forall v \in V \setminus \{s, t\}$ a variable per nonterminal node
- $c_{u,v} \geq 0 \ \forall (u, v) \in A$ a nonnegative capacity per arc

$$
\begin{aligned}
\min_{d_{u,v}, z_v} \quad & \sum_{(u,v)\in A} c_{u,v} d_{u,v} & \\
s.t. \quad & d_{u,v} - z_u + z_v \geq 0 & \forall (u, v) \in A, u \neq s, v \neq t \\
& d_{s,v} + z_v \geq 1 & \forall (s, v) \in A \\
& d_{u,t} - z_u \geq 0 & \forall (u, t) \in A \\
& d_{u,v} \geq 0 & \forall (u, v) \in A
\end{aligned}
$$

We note that the first constraint of this formulation is very similar to the first constraint of the Lagrangian relaxation of the flexible FDC assortment problem.

Let us construct a graph on which solving the minimum cut is equivalent to solving the Lagrangian relaxation. Let $G = (V, A)$ be a graph such that V is its set of nodes and A is its set of arcs. For $o \in O$ and $j \in J$, let v_o^j be a node in V. For SKU i and FDC j, let v_i^j be a node in V. By assumption, the FDCs can be separated into two groups such that no two FDCs from the same group can ship to the same customer node. Let J^+ and J^- be these two groups.

For $o \in O$ and $j \in J^+$, by assumption, $n(j) \in J^-$, We set:

- $(s, v_o^j) \in A$ with capacity $c_{s,v_o^j} = p_j \mu(D_o^j)$
- $(v_o^{n(j)}, v_o^j) \in A$ with capacity $c_{v_o^{n(j)}, v_o^j} = (1 - p_j)\mu(D_o^j)$

- $(v_o^j, v_i^j) \in A$ with capacity $c_{v_o^j, v_i^j} = \infty$
- $(v_i^j, t) \in A$ with capacity $c_{v_i^j, t} = \lambda_j$

Conversely, for $o \in O$ and $j \in J^-$, by assumption, $n(j) \in J^+$. We set:

- $(v_o^j, t) \in A$ with capacity $c_{v_o^j, t} = p_j \mu(D_o^j)$
- $(v_o^j, v_o^{n(j)}) \in A$ with capacity $c_{v_o^j, v_o^{n(j)}} = (1 - p_j)\mu(D_o^j)$
- $(v_i^j, v_o^j) \in A$ with capacity $c_{v_i^j, v_o^j} = \infty$
- $(s, v_i^j) \in A$ with capacity $c_{s, v_i^j} = \lambda_j$

The minimum s-t cut formulation on G is:

$$
\begin{aligned}
\min_{d_{u,v}, z_v} \quad & \sum_{(u,v) \in A} c_{u,v} d_{u,v} & \\
s.t. \quad & d_{v_o^{n(j)}, v_o^j} - z_{v_o^{n(j)}} + z_{v_o^j} \geq 0 & \forall o \in O, j \in J^+ \\
& d_{v_o^j, v_o^{n(j)}} - z_{v_o^j} + z_{v_o^{n(j)}} \geq 0 & \forall o \in O, j \in J^- \\
& z_{v_i^j} - z_{v_o^j} \geq 0 & \forall o \in O, i \in I, j \in J^+, \text{ if } i \in o \\
& z_{v_o^j} - z_{v_i^j} \geq 0 & \forall o \in O, i \in I, j \in J^-, \text{ if } i \in o \\
& d_{s, v_o^j} + z_{v_o^j} \geq 1 & \forall o \in O, j \in J^+ \\
& d_{v_o^j, t} + z_{v_o^j} \geq 0 & \forall o \in O, j \in J^- \\
& d_{v_i^j, t} + z_{v_i^j} \geq 0 & \forall i \in I, j \in J^+ \\
& d_{s, v_i^j} + z_{v_i^j} \geq 1 & \forall o \in O, j \in J^-
\end{aligned}
$$

Note that the objective can be rewritten as follows:

$$
\begin{aligned}
\min_{d_{u,v}, z_v} \sum_{o \in O, i \in I, j \in J^+} & \mu(D_o^j) p_j d_{s, v_o^j} + \mu(D_o^j)(1 - p_j) d_{v_o^{n(j)}, v_o^j} + d_{v_i^j, t} \lambda_j \\
& + \sum_{o \in O, i \in I, j \in J^-} \mu(D_o^j) p_j d_{v_o^j, t} + \mu(D_o^j)(1 - p_j) d_{v_o^j, v_o^{n(j)}} + d_{s, v_i^j} \lambda_j
\end{aligned}
$$

$$\min_{d_{u,v},z_v} \sum_{o\in O, i\in I, j\in J^+} \mu(D_o^j)p_j(1-z_{v_o^j}) + \mu(D_o^j)(1-p_j)d_{v_o^{n(j)},v_o^j} + z_{v_i^j}\lambda_j + \sum_{o\in O, i\in I, j\in J^-} \mu(D_o^j)p_j z_{v_o^j} + \mu(D_o^j)(1-p_j)d_{v_o^j,v_o^{n(j)}} + (1-z_{v_i^j})\lambda_j$$

By letting $Z_o^{j,n(j)} = d_{v_o^{n(j)},v_o^j}$, $Y_o^j = z_{v_o^j}$, and $X_i^j = z_{v_i^j}$ for $i \in I$, $o \in O$, and $j \in J^+$; conversely, $Z_o^{j,n(j)} = d_{v_o^j,v_o^{n(j)}}$, $Y_o^j = (1-z_{v_o^j})$, and $X_i^j = (1-z_{v_i^j})$ for $i \in I$, $o \in O$, and $j \in J^-$. The objective becomes:

$$\min_{Y_o^j,X_i^j,Z_o^{j,n(j)}} \sum_{o\in O, i\in I, j\in J^+} \mu(D_o^j)p_j(1-Y_o^j) + \mu(D_o^j)(1-p_j)Z_o^{j,n(j)} + X_i^j\lambda_j + \sum_{o\in O, i\in I, j\in J^-} \mu(D_o^j)p_j(1-Y_o^j) + \mu(D_o^j)(1-p_j)Z_o^{j,n(j)} + X_i^j\lambda_j$$

$$\min_{Y_o^j,X_i^j,Z_o^{j,n(j)}} \sum_{o\in O, i\in I, j\in J} \mu(D_o^j)p_j(1-Y_o^j) + \mu(D_o^j)(1-p_j)Z_o^{j,n(j)} + X_i^j\lambda_j$$

The objective of the minimum s-t cut on G is equal to the objective of the Lagrangian relaxation. The z variables are the labels of the nodes. $z_v = 1$ if node v is in the source set and 0 otherwise.

The above conversions from $z_{v_o^j}$ to Y_o^j and from $z_{v_i^j}$ to X_i^j are equivalent to selecting a SKU in the assortment of FDC $j \in J^+$ if the corresponding node lies in the source set of the minimum s-t cut and conversely it is in the assortment of FDC $j \in J^+$ if the corresponding node lies in the sink set. Furthermore, $\forall o \in O, j \in J$, we have:

$$\left.\begin{array}{ll} d_{v_o^{n(j)},v_o^j} - z_{v_o^{n(j)}} + z_{v_o^j} \geq 0 & \forall o \in O, j \in J^+ \\ d_{v_o^j,v_o^{n(j)}} - z_{v_o^j} + z_{v_o^{n(j)}} \geq 0 & \forall o \in O, j \in J^- \end{array}\right\} \rightarrow Z_o^{j,n(j)} + Y_o^j + Y_o^{n(j)} \geq 1$$

And $\forall o \in O, i \in o, j \in J$:

$$\left.\begin{array}{ll} z_{v_i^j} - z_{v_o^j} \geq 0 & \forall o \in O, i \in o, j \in J^+ \\ z_{v_o^j} - z_{v_i^j} \geq 0 & \forall o \in O, i \in o, j \in J^- \end{array}\right\} \rightarrow Y_o^j \leq X_i^j$$

This confirms that the two problems are equivalent. Let n be the number of nodes in G, m be the number of arcs, and $T(n,m)$ be the time complexity of solving

the minimum s-t cut on such a graph.

$$\begin{aligned} n &= (|I| + |O|) * |J| + |\{s, t\}| \\ m &= (|I| + 2 * |O| + \sum_{o \in O} |o|) * |J| \\ &= (|I| + (2 + \delta)|O|) * |J| \end{aligned}$$

Where δ is the average order length. Using the Pseudoflow algorithm [14], we get $T(n, m) = O(mn \log(\frac{n^2}{m}))$. As the number of orders dominates the number of products, this simplifies the expression to:

$$O\left((2 + \delta)|O|^2|J|^2 \log\left(\frac{|O||J|}{2 + \delta}\right)\right)$$

This confirms that for a closed-chain structure of an even number of DCs, taking the Lagrangian relaxation of the SKU capacity constraint leads to a tractable problem. In the following section, we will explain how to update the Lagrangian parameter.

Optimizing Lagrangian parameters

In this section, we show how to update the Lagrangian parameter.

Note that the Lagrangian dual objective function:

$$g(\boldsymbol{\lambda}) = \min_{X_i^j, Y_o^j, Z_o^{j,n(j)}} \sum_{j \in J,\ o \in O} \mu(D_o^j) p_j (1 - Y_o^j) + \mu(D_o^j)(1 - p_j) Z_o^{j,n(j)} + \sum_{i \in I, j \in J} \lambda_j X_i^j - \sum_{j \in J} \lambda_j k_j$$

is a concave piecewise linear function of $\boldsymbol{\lambda}$.

Let $\beta(\boldsymbol{X}, \boldsymbol{Y})$ be a scalar such that $\beta(\boldsymbol{X}, \boldsymbol{Y}) = \min_{X_i^j, Y_o^j} \sum_{j \in J,\ o \in O} \mu(D_o^j) p_j (1 - Y_o^j) + \mu(D_o^j)(1 - p_j) Z_o^{j,n(j)}$. Let $\boldsymbol{\alpha}(\boldsymbol{X}, \boldsymbol{Y})$ be a vector of dimension $|J|$, such that $\boldsymbol{\alpha}(\boldsymbol{X}, \boldsymbol{Y}) = \left\{\sum_{i \in I} X_i^j - k_j\right\}_{j \in J}$. Let $\langle \cdot | \cdot \rangle$ be the usual inner product. We can formulate $g(\boldsymbol{\lambda})$ as follows:

$$g(\boldsymbol{\lambda}) = \min_{\boldsymbol{X}, \boldsymbol{Y}} \langle \boldsymbol{\alpha}(\boldsymbol{X}, \boldsymbol{Y}) | \boldsymbol{\lambda} \rangle + \beta(\boldsymbol{X}, \boldsymbol{Y})$$

For X_i^j, Y_o^j binary, $\left\{\langle \boldsymbol{\alpha}(\boldsymbol{X}, \boldsymbol{Y}) | \boldsymbol{\lambda} \rangle + \beta(\boldsymbol{X}, \boldsymbol{Y})\right\}$ is a set of linear functions of $\boldsymbol{\lambda}$. Taking the minimum over this set of functions for any value of $\boldsymbol{\lambda}$ results in a concave piecewise linear function of $\boldsymbol{\lambda}$.

We have proven that evaluating $g(\boldsymbol{\lambda})$ is tractable. Solving for a given Lagrangian parameter $\boldsymbol{\upsilon}$, the minimum cut provides an optimal value for all the decision variables X_i^j, Y_o^j, we can thus set $\boldsymbol{\alpha_\upsilon}$ and $\beta_{\boldsymbol{\upsilon}}$ such that $g(\boldsymbol{\upsilon}) = \langle \boldsymbol{\alpha_\upsilon} | \boldsymbol{\upsilon} \rangle + \beta_{\boldsymbol{\upsilon}}$. By definition, we have $g(\boldsymbol{\lambda}) \leq \langle \boldsymbol{\alpha_\upsilon} | \boldsymbol{\lambda} \rangle + \beta_{\boldsymbol{\upsilon}}$. Each time $g(\boldsymbol{\lambda})$ is evaluated, a linear constraint on *dual* can be stored thus building a matrix A and a vector B such that $g(\boldsymbol{\lambda}) \leq A\boldsymbol{\lambda} + B$.

To solve the Lagragian relaxation, gradient descent would work. However, because of the structure of the problem we presented above, different values of $\boldsymbol{\lambda_j}$ may be associated with the same cut. Hence, gradient descent may lead the algorithm to solve multiple minimum cut problem with identical optimal solutions. We have shown that every time we solve a minimum cut problem to evaluate $g(\boldsymbol{\lambda})$, it is equivalent to finding a bounding constraint for:

$$g(\boldsymbol{\lambda}) = \min_{\boldsymbol{X},\boldsymbol{Y}} \langle \boldsymbol{\alpha}(\boldsymbol{X}, \boldsymbol{Y}) | \boldsymbol{\lambda} \rangle + \beta(\boldsymbol{X}, \boldsymbol{Y})$$

Consequently, by solving the following linear program:

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \ & obj \\ s.t. \ & obj \leq A\boldsymbol{\lambda} + B \\ & \boldsymbol{\lambda} \geq 0 \end{aligned}$$

we find a $\boldsymbol{\lambda_j}$ corresponding to an extreme point of the set of feasible solutions. Evaluating $g(\boldsymbol{\lambda})$ for this value returns one of two outcomes: (i) the cut that minimizes the Lagrangian relaxation is not yet stored in the matrix A and vector B, hence we add it to the set of constraint and continue solving, (ii) the cut that minimizes the Lagrangian relaxation is already in matrix A and vector B, we have found an optimal solution of $\boldsymbol{\lambda_j}$

2.4 Comparative analysis of configurations of the fulfillment network

To better understand how different configurations of flexible fulfillment network impact the optimal value of the objective function of f-MIP, in this Section, we explore the impact of flexibility on two toy use cases. In the first experiment, we consider that for $j \in J$, $p_{j,\,j} = 0$ and for $j' \in J$ with $j \neq j'$, $p_{j',j}$ is either 0 or 1, i.e., either as efficient as a dedicated FDC or as costly as routing the order to the RDC. This means that if spillover is allowed from FDC j' to FDC j, fulfilling an order type placed at FDC j from either FDC j or FDC j' is equivalent. In the second experiment, the

penalty is proportional to the distance between the two FDCs. The toy use cases are solved to optimality using the Gurobi solver on different flexible networks.

To explore the performance of each structure, we generate small instances of f-MIP, such that $|O| = 1,000$, by randomly sampling order types from our business partner's transaction dataset. We let the number of FDCs $|J|$ take values from 4 to 16.

Metrics of interest

To measure the performance of each configuration, we define the following metrics. For a configuration and a set of orders, we call *score* the optimal objective value of f-MIP.

We call *fulfillment ratio*, the expected ratio of the score obtained for a configuration by the maximum score that could be reached should all the capacity constraints be relaxed.

We also evaluate how robust each configuration is to demand changes by adding perturbations to the demand datasets. Each output assortment allocation is tested on several perturbed transaction datasets. By repeating this experiment several times, we compute an estimate of standard deviation of the fulfillment ratio associated with each configuration over these perturbed datasets. We call this metric *robustness*

The no-flexibility configuration provides an lower bound on the objective function of f-MIP, hence on every configuration's fulfillment ratio. We call *baseline* the fulfillment ratio of the no-flexibility configuration. To compare every other configuration to the no-flexibility one, we call *performance increase* the difference between a configuration's fulfillment ratio and baseline, divided by baseline, i.e. the relative fulfillment ratio increase over baseline.

We will report the fulfillment ratio of each studied configuration, along with the performance increase per flexible link. The first metric provides quantitative insights on the quality of the configuration for our objective function, while the second shows the marginal value of each flexible link. Although we do not directly optimize for the second metric in this Chapter, we remind the reader that each flexible link requires fixed costs to be set up and kept active. For each new flexible link, the e-retailer needs to solve many optimization problems such as route planning, labor planning, and scheduling, thereby making configurations with high performance increase per link more desirable than others. If the volume of goods shipped through a flexible link is not large enough it may be too costly to operate. In reality, e-retailers balance the fulfillment ratio and the cost associated with setting up the extra fulfillment links to select new links to operate.

Evaluating the impact of flexibility in fulfillment network with binary penalty parameters

We focus on the flexible fulfillment network presented on Figure 2.7. For each of these configurations, we solve the problem to optimality. The fulfillment ratios are reported as bars in Figure 2.2 while the robustness appears as error bars. These experiments show that the closed chain and the set of two-chains tend to yield nearly half of the fully connected performance increase. We also note that the gap between the fulfillment ratio of chaining and the one of fully connected increases with the number of FDCs in the network. The Figure also shows that the closed-chain configuration yields higher fulfillment ratio than multiple two-chains, along with more robustness.

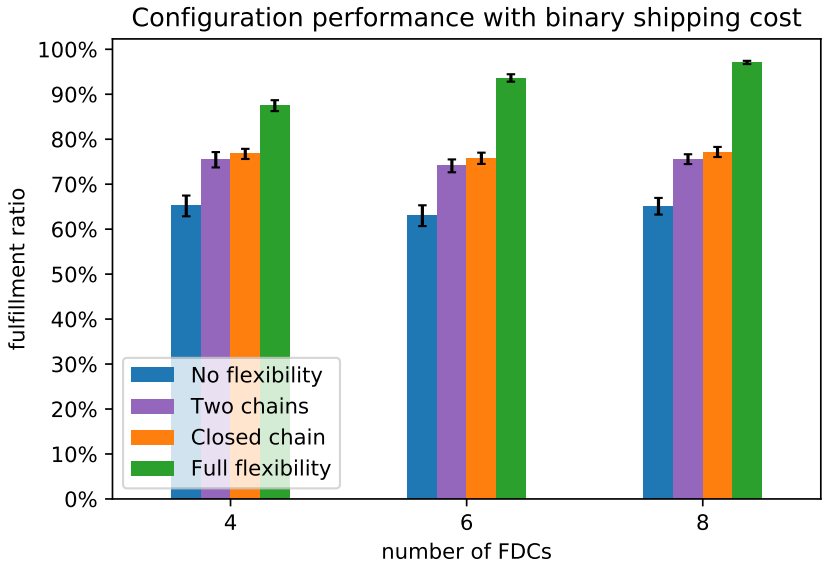


Figure 2.2: Flexibility comparison with binary shipping costs

On Figure 2.3, we report the performance increase per flexible link of each configuration. The Figure shows that even though the fulfillment ratio of the fully connected configuration grows faster with the number of FDCs than the closed chain configuration one, the growth does not compensate for the number of additional flexible links. Among all the configurations we study, the closed chain configuration yields the maximum performance increase per flexible link.

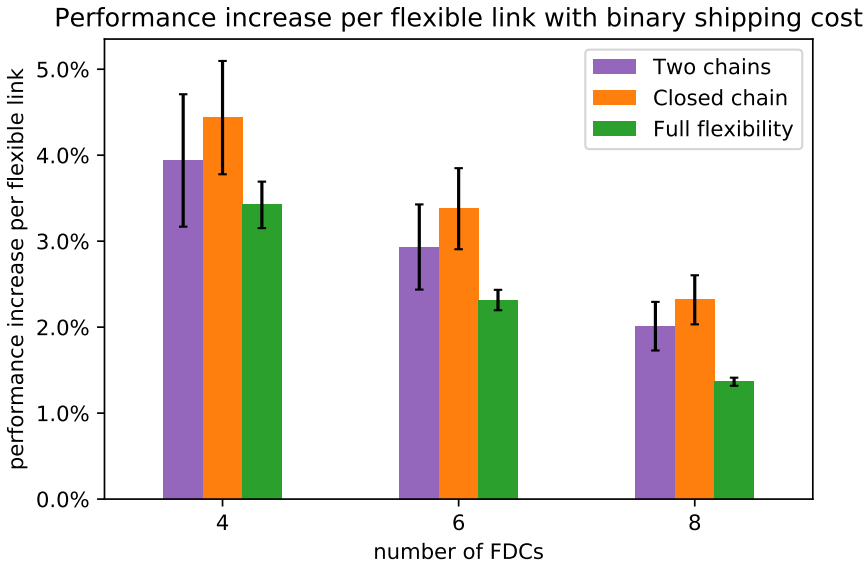


Figure 2.3: Comparison of performance increase per flexible link with binary shipping costs

Evaluating impact of flexibility in fulfillment network with shipping costs proportional to distance between DCs

To investigate the impact of the flexibility configuration in a more realistic setting, we uniformly distribute the FDCs on the unit circle and set the penalty of shipping from FDC j to FDC j' to be equal to $\min\{1, 0.6 * d_{j,j'}\}$, where $d_{j,j'}$ is the usual distance in the 2D plane between these two FDCs. Because the FDCs are distributed on the unit circle, the distance from any FDC to the RDC is $d = 1$. To have more profitable flexible links with a small number of FDCs, the penalty is set to be equal to a fraction of the distance between the two FDCs, i.e., $\min\{1, 0.6 * d_{j,j'}\}$. Figure 2.4 shows that under this assumption, the closed-chain configuration achieves over 50% of the full flexibility benefit for up to 10 chained FDCs; furthermore, its performance remains over 40% of the full flexibility performance even when the number of FDCs is increased to 16.

Similarly to the previous case study, we report on Figure 2.5 the performance increase per flexible link of each configuration. The Figure indicates that although the fulfillment ratio of the fully connected configuration outperforms the closed chain configuration one, the performance increase per flexible link of the closed chain configuration is much higher. The figure tends to show that when shipping penalties

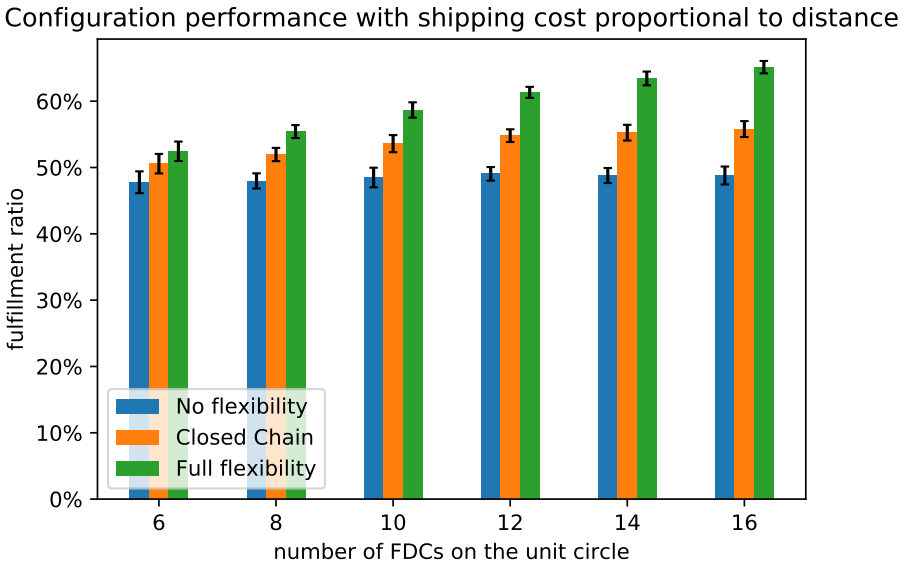


Figure 2.4: Flexibility comparison on the unit circle

are proportional to distances, and the FDCs are distributed over the unit circle, the performance increase per link of the closed chain reaches a maximum for 10 facilities and decreases for longer chains.

Figures 2.6a and 2.6b show the flexible network built as explained above with 8 and 16 FDCs. The blue dot represents the RDC and the orange dots represent the FDCs. Each line represents a flexible link between two FDCs, where the color of the line indicates the penalty from green for 0 to red for 1. This shows that while using a closed chain for 8 FDCs only removes 3 flexible links per FDC, the same configuration for 16 FDCs removes 9 links per FDC. This explains why the relative performance of the closed-chain configuration compared to the fully flexible configuration decreases with the number of FDCs.

2.5 Numerical Analysis

In this section, we evaluate the performance of the algorithm and the value of closed-chain flexibility. We have proven that a solution to the Lagrangian relaxation of the f-MIP is found in polynomial time for closed chains of even number of FDCs. Using the transaction data of our business partner, JD.com, we benchmark the above algorithm against heuristics. Here, we use the data and the warehouses cluster of our

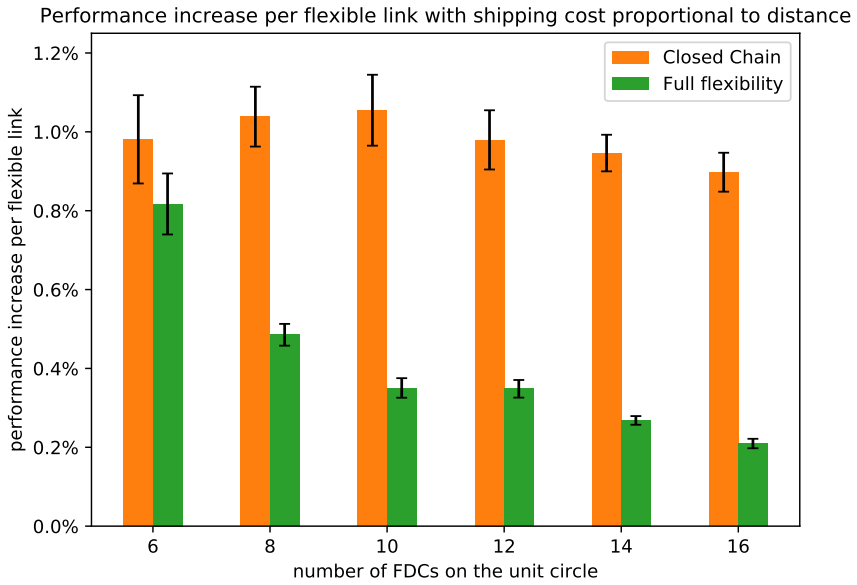


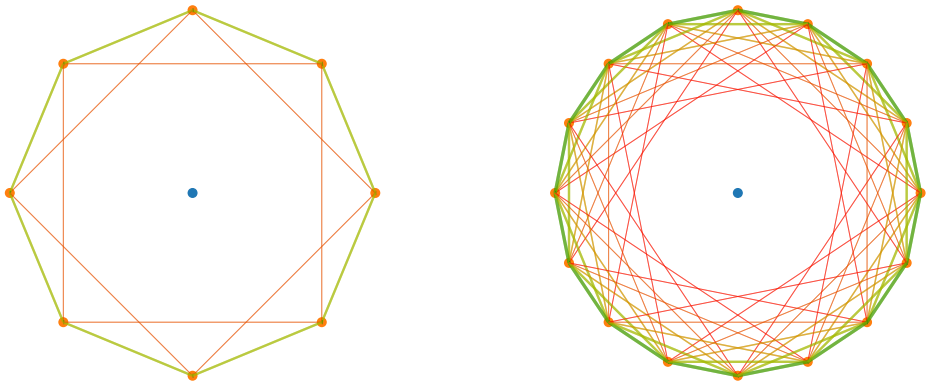
Figure 2.5: Comparison of performance increase per flexible link with shipping costs proportional to distances

business partner to assess the performance of our method when flexibility is added to the fulfillment network. To protect our business partner from revealing sensitive data to the competition, the data presented in this section are a masked version of the real data.

This section is divided into two experiments. The first one has unpenalized flexibility on small datasets. The second models the flexibility cost as proportional to the distance between warehouses in one of our business partner's warehouse clusters and relies on real-sized datasets. The benchmark compares two algorithms as a baseline for assessing the performance of our method: *single iteration* and *multiple iteration*.

The algorithm *single iteration* solves the FDC assortment problem for each FDC customer node pair individually. This algorithm is currently used in the industry for product placement decisions. We use the solution output by this algorithm and evaluate the performance of the assortment under the flexible setting. This provides the benchmark with a baseline.

A simple heuristic for solving the f-MIP when flexibility creates a closed chain of FDCs is to iterate the *single iteration* algorithm. In the original FDC assortment



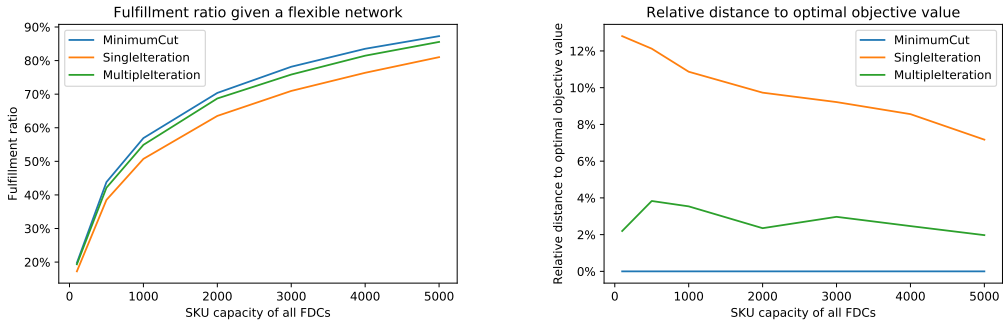
(a) 8 FDCs (b) 16 FDCs

Figure 2.6: FDCs distributed over the unit circle, with green lines representing the flexible links

problem, the benefit of being able to fulfill an order is set to the expected demand $\mu(D_o^j)$. When we iterate *single iteration*, we change the benefit to allow the assortments of each FDC to evolve with respect to the assortments of neighboring FDCs. Let w_o^j be the benefit associated with being able to fulfill order type o from FDC j. As j' is the neighboring FDC, if j' can fulfill o, then placing the required SKUs in j so it can also fulfill this order type only yields an additional reward of $(1-p_j)\mu(D_o^j)$. Because the flexibility forms a closed chain, there exists an FDC j' such that $n(j') = j$. Therefore, if j can fulfill o, it can also fulfill occurrences of this order type at FDC j'. This only has an additional reward of $p_{j'}\mu(D_o^{j'})$ if FDC j' cannot fulfill the order. This confirms that $w_o^j = \mu(D_o^j) - Y_o^{n(j)} p_j \mu(D_o^j) + (1 - Y_o^{j'}) p_{j'} \mu(D_o^{j'})$. The idea of *multiple iteration* is to fix the value of the decision variables for all FDCs apart from one. Without loss of generality, let FDC j be the FDC of which the assortment is being optimized, the decision variables Y_o^j and X_i^j are not fixed. At each iteration, compute the order type reward w_o^j for each $o \in O$ and deduce an assortment for j using *single iteration*. We iterate several times over each FDC randomly and keep the best solution found.

Experiment with unpenalized flexibility

The dataset on which the benchmark is run is built from a real transaction dataset from our business partner. We randomly select 100,000 order types observed during a month of operation and uniformly distribute the demand over four or six fictional FDCs. For different SKU capacities ranging from 100 to 5,000 units, we compute the fulfillment ratio obtained by each algorithm. This ratio is computed by dividing the objective value reached by the algorithm by the maximum objective value reachable if each FDC were to hold every single item. Figure 2.7a shows that when the closed chain has a cost of 0 for shipping to a neighboring FDC, the scores of *multiple iteration* and our algorithm are much higher than the score of *single iteration*. This shows that adding flexibility in the network as a closed chain creates opportunities for better performance. Furthermore, we see that our algorithm outperforms the *multiple iteration* algorithm by a large margin. Note that the capacities chosen for this comparison are such that our algorithm finds an optimal solution to the problem. It appears in Figure 2.7b that *multiple iteration* is consistently only a few percent below the optimal solution, thereby making it a powerful and efficient heuristic for industries to use. Figure 2.8b further shows that the gap between the optimal solution found with the minimum cut algorithm and those found with heuristics widens with the number of FDCs in the closed chain.



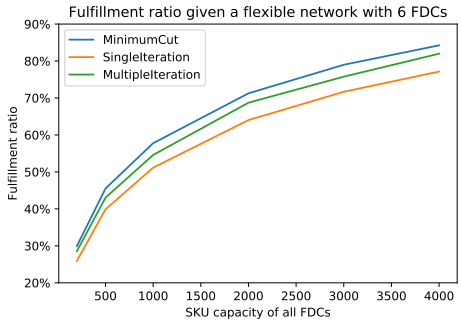
(a) Fulfillment ratio

(b) Relative distance to optimal

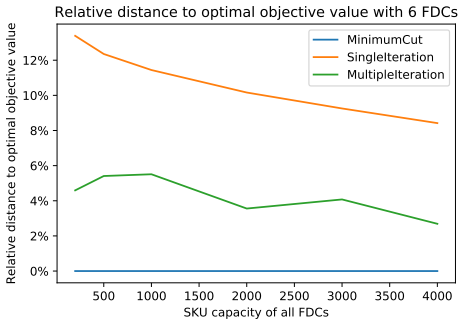
Figure 2.7: Performance comparison for four FDCs, $|O| = 100,000$, $|I| = 71,346$

Experiment on realistic warehouse cluster

In this subsection, our business partner provided transaction data for a warehouse cluster, along with information on how often product placement decisions are made. Figure 2.9 shows the warehouse cluster with the RDC and four FDCs. The red arcs



(a) Fulfillment ratio



(b) Relative distance to optimal

Figure 2.8: Performance comparison for six FDCs, $|O| = 100,000$, $|I| = 71,346$

model the cost of a missed order that is fulfilled from the RDC; the green arcs model the flexible links between FDCs, which are shaped as a chain of four warehouses. The flexibility cost associated with each of the green links is proportional to the distance between the two FDCs. Our business partner selects the assortment of products to keep in each warehouse for every specific length of time. We call the period between two decisions the *decision cycle*. During a decision cycle, approximately one million unique order types are placed across the four FDCs. The set of order types consist of approximately 300,000 SKUs. When an e-retailer selects the assortment of SKUs to keep in each warehouse, the decision is based on past transaction data and forecasts. Therefore, we optimize the assortments on a given decision cycle and evaluate based on both the next cycle and that for which the assortment is optimized. This shows the performance of the algorithm on a given optimization problem compared to industry practices and highlights the robustness of the algorithm for deciding which SKUs to keep in which warehouses

Unlike the previous experiment, where the flexibility costs were set to 0, in this experiment, they are proportional to the distance between FDCs. As the flexibility cost increases, the use of flexible links becomes less profitable and routing an order to the RDC yields a comparatively lower penalty. Therefore, the optimality gap is smaller for industry heuristics. Figure 2.10 shows the performance of each algorithm on the decision cycle for which they are optimized. Our algorithm shows an increase of a few percent, even when the flexibility costs are up to 75% of the cost of routing an order to the RDC.

Figure 2.11 shows the performance of the given assortments on the upcoming decision cycle. As demand varies from one cycle to the next, the fulfillment ratio slightly decreases. However, our algorithm still shows better performance than the

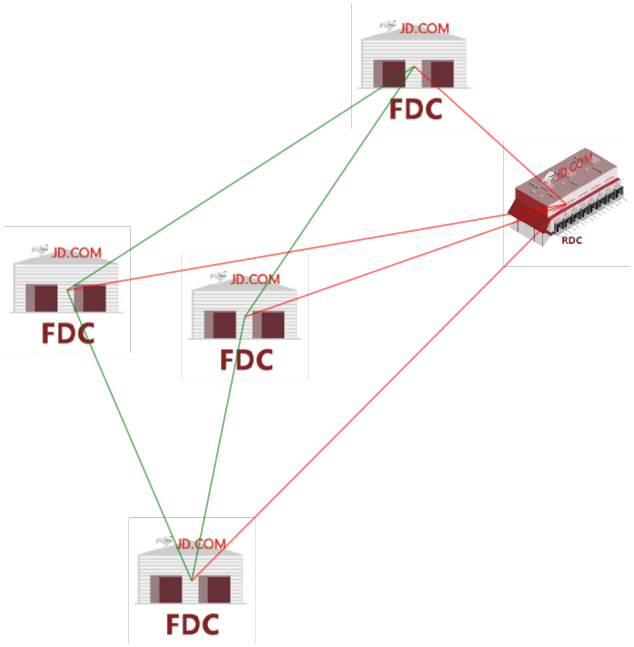
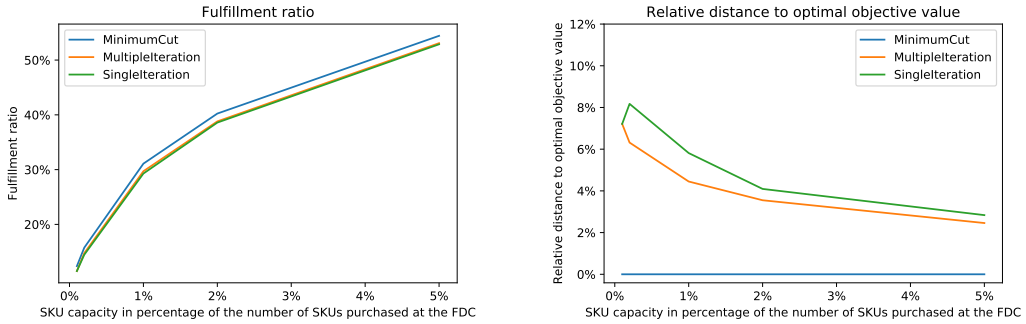


Figure 2.9: Warehouse cluster with flexibility

industry heuristics.

Case study on how optimal assortments are changed by flexibility

We have shown in the above paragraphs that well optimized assortments in a flexible network outperform the assortments in an inflexible network by a large margin. In this subsection, we review a case study to show how products in assortments are replaced with others when flexibility is introduced. This will help managers to evaluate the cost of upgrading their product assortments if they introduce flexibility in the network and provide some intuition on what type of fulfillment networks benefits the most from flexibility.



(a) Fulfillment ratio (b) Relative distance to optimal

Figure 2.10: Performance comparison on a decision cycle for a real warehouse cluster

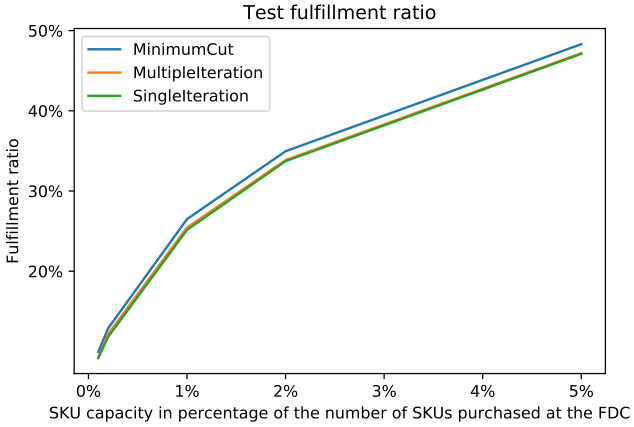


Figure 2.11: Performance comparison on the upcoming decision cycle for a real warehouse cluster

We consider the four warehouses from Figure 2.9. This case study is based on real large-scale demand and arbitrary realistic capacities. Note that all figures have been rescaled to preserve confidentiality. In this example, presented on Figure 2.12a, the upper DC has a larger demand than the other DCs for similar SKU capacity; on the contrary, the lower DC has a much smaller capacity compared to the demand. The capacity constraint of these two DCs have much higher shadow prices than the two other DCs. As expected, the added flexibility takes advantage of this imbalance of demands by redistributing SKUs from the DCs with high shadow price capacity constraint to the others. After adding flexibility to the network, the composition of the assortment changes. The red arrows in Figure 2.12b show how many SKUs

removed from a given DC are placed in the neighboring DC. The numbers are rescaled and do not correspond to percentages.

The right and left DCs have lower shadow price associated with their capacity constraints for the no flexibility configuration. Consequently, we expect the added flexibility to enable these two to fulfill orders placed at the top and bottom DCs, i.e., DCs with higher shadow price on their capacity constraints. Compared with the assortments for the no flexibility configuration, the right and left DCs remove approximately 15% of the SKUs and replace these mostly with SKUs needed to fulfill orders placed at the top and bottom DCs. The red arrows on Figure 2.12b show that:

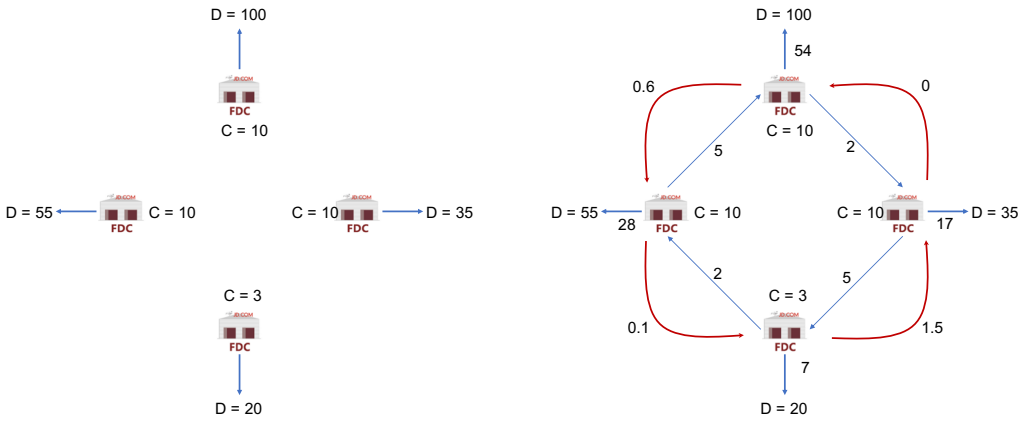
- 1.5 units of SKUs originally held at the bottom DC in the no flexibility configuration are moved to the right DC, such a change represents approximately 15% of the right DC's assortment
- 0.6 units of SKUs originally held at the top DC in the no flexibility configuration are moved to the left DC, such a change represents approximately 6% of the left DC's assortment

How the assortment of the right and left DCs evolve when flexibility is added to the fulfillment network enable these DCs to better support the top and bottom DCs that have high shadow cost associated with their capacity constraints.

On the contrary, the top and left DCs have higher shadow price on their capacity constraints. Consequently, flexibility will enable neighboring DCs to fulfill demand for the top and bottom ones. We observe that the top DC removes approximately 8% of its assortment for the no flexibility configuration, by moving 80% of these products to the left DC, as indicated by the top left red arrow on Figure 2.12b. Similarly, the bottom DC removes 50% of the SKUs of its assortment for the no flexibility configuration and most of these SKUs are held at the right DC for the closed chain configuration, as indicated by the bottom right red arrow on Figure 2.12b. Both DCs rely heavily on their neighboring DCs to fulfill demand.

Figure 2.12b makes it very clear that although the two DCs with lower shadow prices take in SKUs from the other two DC, the reverse does not occur. Adding flexibility in the network balances the SKUs in order to relax the capacity constraint with the highest shadow prices. Finally, the figure indicates on each link the number of orders that are shipped. As expected, the links sending orders to the top and bottom DCs are used more heavily than the other flexible links.

No typical pattern was observed among products that get assigned to neighboring DC when the flexibility is added. We cannot draw general conclusions at the product level. From our case study, we can conclude that flexibility relaxes the constraints with high shadow price, thus improving the overall performance.



(a) Inflexible network with demand and capacity per DC

(b) Flexible network with utilization of flexible links and balancing of SKUs

Figure 2.12: Case study on how assortments change when flexibility is introduced

Sensitivity analysis on order length distribution

The work presented above focuses on optimizing the in-stock rate of a group of SKUs and ensuring efficient fulfillment. The results presented in this paper are tied to the distribution of order length of our business partner, JD.com. E-retailers may have different distributions of order length. In this subsection, we provide a sensitivity analysis, by studying the impact of different order length distributions.

In order to make this analysis as realistic as possible, similarly to how it is conducted in Chapter 1, we use importance sampling to resample real datasets with different order length distributions. For this analysis, we let the exponential parameter vary from 0.2 to 1 and we study three different SKU capacity constraints. For this experiment, the SKU capacity constraints are an indication; the problem is solved with specific Lagrangian parameters leading to assortments close to these capacities. Although all the experiments are run with the same Lagrangian parameters, the resulting capacities are slightly different.

Figure 2.13 shows the relative performance of our algorithm. To compute this metric, we compute the performance of our minimum capacity cut algorithm and the performance of the *multiple iteration* (iterate *industry standard* ten times). The metric shown on the graph is the ratio of performance increase of our minimum cut algorithm over *multiple iteration* by the performance of *multiple iteration*. Our algorithm captures the co-purchase behavior of products and utilizes the added flexibility far better than the industry standard algorithm, thus leading to larger increases in

performance when the average order length increases. As a small number of products are responsible for a large volume of orders, the performance increase rises with tighter capacity constraints. Finally, when the distribution is heavier on singletons (one-SKU orders), all curves converge toward the performance increase presented above.

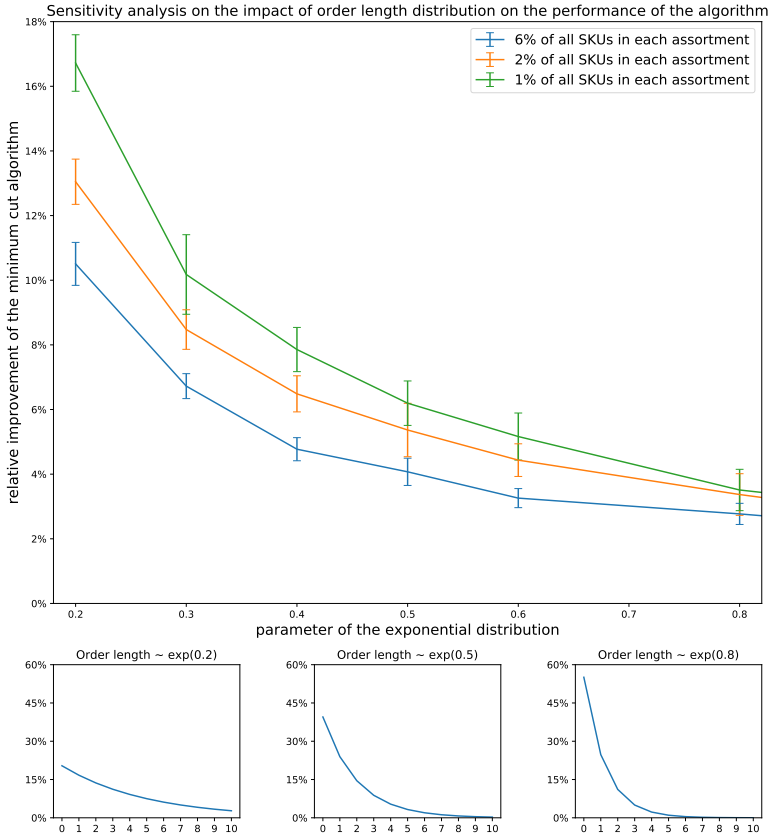


Figure 2.13: Sensitivity analysis of the impact of order length distribution on the performance of our algorithm in a four-warehouse closed-chain network

We observe in the data that products purchased together form clusters that are less connected with other products. This pattern exists because some products pro-

vide a higher value when grouped together compared to independent cases. These patterns are important for our algorithm and flexibility to overperform the industry standard algorithm. If such patterns did not exist, when the order length distribution becomes heavier on longer orders, the flexibility would not increase the performance of the network as much, as each DC would have to hold all products to fulfill highly interconnected orders. Owing to these patterns, a DC can rely on its neighboring DC to handle a cluster of longer orders.

2.6 Discussion

The above experiments show a performance improvement four-fold greater than that measured in Chapter 1 for an inflexible network. Our work highlights the significant improvement that can be achieved by allowing flexibility in the network and optimizing for it.

The method proposed in this study shows sufficient robustness for selecting a product assortment for an upcoming decision cycle. However, the performance of the algorithm could be greatly improved if forecasted information were provided and predictive modeling techniques were applied.In Chapter 1, we developed techniques to make use of forecasted sales of SKUs per warehouse to build better assortments for a single RDC/FDC pair. This algorithm is build on the assumption that each FDC is independent of the other, i.e. there is no flexibility, consequently, it cannot be applied directly to the scenario presented in this work. Adapting the techniques from Chapter 1 and designing new ones to use predictive modeling in a flexible fulfillment network is a challenging problem that could be interesting to tackle in future research.

Selecting which SKUs to place in which DC for each cycle is a multistage problem. Replacing an SKU from one period to the next has a cost. In the work presented above, we assumed this cost can be ignored. Because the cardinality of order types grows exponentially with the number of SKUs, predictive models usually focus on predicting the product-level demand. To solve the multistage problem, the model must first be updated to account for the predicted demand. Therefore, we chose to leave this improvement for future work. We instead provide here a heuristic-based algorithm for the multistage problem. We assume that products can be partitioned between seasonal products whose demand vary significantly with time and nonseasonal products whose demand is relatively steady. Under this assumption, using our algorithm on many periods will result in SKU allocation that captures the steady demand well. Once this assortment allocation is found, the cost of replacing a product in an FDC can be computed. By adding arcs from the product node in an FDC

to the sink node or the source node, according to whether the FDC lies in J^+ or J^- with capacity equal to that cost, the problem can be solved again for each new cycle and will only allow a small number of changes in the assortment allocation.

Finally, it should be noted that solving the Lagrangian relaxation of the problem does not guarantee finding an optimal solution to the f-MIP. It provides sets of feasible solutions that are optimal for more constrained capacity and of unfeasible solutions that would be optimal for relaxed capacity. Solving the Lagrangian relaxation can be used in two ways. We can first consider completing a feasible assortment allocation provided by the Lagrangian relaxation. Using the Lagrangian parameter of this feasible solution, an upper bound on the optimality gap can be easily computed. The Lagrangian parameter also has a managerial impact. It can be used as a cost in terms of a number of orders of adding an SKU to a DC. An industry that has an estimate of how costly it is to add or change an SKU in the assortment allocation can set a value υ, such that only SKUs adding at least υ orders to the set of efficiently fulfilled orders can be considered. These SKU costs per DC can be directly used as Lagrangian parameters to obtain an optimal allocation of SKUs.

2.7 Conclusion

In this Chapter, we have extended the delivery network of our business partner, JD.com, to a more connected flexible network. Using their transaction data, we have experimentally demonstrated the impact of flexibility for product placement in a fulfillment network. We observed that short closed-chain structures — where each local DC serves two customer areas, thus forming a cycle — achieve over 50% of the performance increase of a fully connected network. As most of the warehouse clusters of our business partners are constituted by fewer than 10 DCs, we developed an efficient algorithm for solving the product placement problem in a flexible fulfillment network, where the flexible links create a closed chain of an even number of DCs. This new algorithm outperforms by a factor of four the former performance improvement achieved on an inflexible network in Chapter 1 and other heuristics used in the industry. Such a performance increase represents hundreds of millions of additional orders that can be efficiently fulfilled every year.

Part II

E-hailing Platforms

Chapter 3

Shared Ride Sustainability

The rise of transportation network companies (TNCs) such as Didi, Lyft and Uber has reshaped the ride hailing industry. These companies offer different modes of transportation among which the two main modes are on-demand private rides and on-demand ride-sharing, referred to as classic rides and shared rides. When opening the ride-hailing app, users often choose between two fares: the full fare for classic ride and the discounted fare for shared ride. When selecting the discounted fare, the passenger implicitly accepts that she may share the ride with another shared passenger. When this happens, the passenger is said to experience a *match*. Experiencing a match often increases the duration and the distance of the ride due to an additional detour needed to pickup and/or drop-off the other shared passenger. Shared ride is different from the usual carpooling services such as WazeCarpool, as dispatching drivers and matching passengers are processed in real time by the ride hailing platform. Hence, the detour a passenger experiences when matched is the result of the decisions made by a matching algorithm. Another difference is e-hailing companies typically pay their driver by time and distance the same rates whether the driver is serving a classic ride or a shared ride. Specifically, the e-hailing company pays the driver the same amount regardless of whether the shared passenger experiences a match or not. When two shared passengers are matched, the e-hailing platform pays only one driver to complete both trips thus reducing its operating costs. Owing to such settings, offering shared rides for a discount to passenger is only profitable if the e-hailing company recovers the discount offered to passengers by matching passengers. Not all matches reduce costs for the company. When two passengers share a significant portion of their rides, having a single driver serve both trips can save up to half of the costs incurred by each passenger having its own driver. Such a match is said to be *efficient*, i.e., matching the two passenger saves a large portion of the costs incurred by dispatching one driver per passenger. A shared trip that reaches its

destination without being matched is said to be *unmatched*. The *cost savings* of a match denotes the difference between the costs incurred by letting each of the two passenger unmatched and the costs of serving both passengers with the same driver. The *efficiency* of a match denotes the ratio of its cost savings by the costs incurred when both passengers remain unmatched. Contrary to the former example, when long detours are needed to match two passenger, the company pays the driver for the extra detour and the match is less efficient. Although passengers pay a discounted price when they request a shared ride, this service can still turn a profit if the e-hailing platform achieves an average efficiency per shared ride that compensate for the discount offered to passengers.

High efficiency matches occurs when two passengers request a similar ride almost at the same time. The likelihood of such an event increases with the demand for shared rides, consequently, most ride hailing companies only operate their shared mode in their densest markets.

When deploying shared ride mode in a region, the e-hailing company is carrying the risk of having too many unmatched shared rides for the mode to be profitable. Furthermore, as it can take some time to grow the demand for this new mode of transportation, shared ride mode can remain unprofitable for an extended period of time. Because starting shared ride in a new region is very uncertain, e-hailing companies mostly only propose the mode in their densest markets. Our work focuses on providing quantitative insights on the existence of a profitable equilibrium with both shared and classic in a regional market. We propose a framework based on past classic ride data to compute such insights before introducing shared ride in the market. That way, we aim at answering the managerial question of whether or not should shared ride be offered in a regional market. In our model, we consider that each passenger can choose among shared ride, classic ride and any competitive option. In this paper, we will focus on public transit as the only competitive option. Typically, these three modes can be ranked from the slowest and most affordable mean of transportation to the fastest most expensive mode in this order: public transit, shared ride, classic ride. We assume that the passengers are utility-maximizing and that the utility associated with a trip is a function of how much time and money is spent on the mean of transportation. The utility of shared rides increases with the discount offered and decreases with the detour experienced. A e-hailing company can thus increase the shared ride utility, hence growing the demand for shared ride, by restricting the maximum detour a passenger can experience and by lowering the fare.

To increase demand density, the e-hailing company can improve the user experience by lowering the maximum amount of detour a passenger can experience or by lowering the fare for shared ride. However, tightening the detour constraint, i.e., the

maximum amount of detour a passenger can experience when matched, makes matching passengers more difficult thus decreasing the ratio of matched shared passengers and the average efficiency of the matches.

In this paper, we provide a model for studying an equilibrium with both shared and classic in a regional market. The general formulation of customer choices and expected profit we propose in Section 3.2 is intractable to solve, to provide quantitative insights on the equilibrium, we propose to approximate a few key functions from simulations in Section 3.3 and 3.4. Finally, we conduct in Section 3.5 a case study in Manhattan to showcase the relevance of this framework. As this model is part of a larger effort to understand the conditions for the existence of a profitable equilibrium with both shared and classic, we detail in Section 3.6 the extensions of this research.

3.1 Literature review

Our work is part of a growing effort to estimate the economic sustainability of the ride sharing industry and to measure the impact it has on our cities. Numerous studies analyse the dynamics of a ride sharing system through simulations. A major contribution [25] was made by studying shareability networks in New York City to estimate what percentage of rides could be shared under the constraint that each passenger should not experience a delay greater than some value. Experimenting both with offline and online algorithm, they observed that limiting the capacity of the vehicle to two passenger has very little consequence for a maximum delay of three minutes in a dense city such as New York city. They showed how much travel time and how many trips could be saved if all users were to shift and use a ride sharing service. Also built on simulations, a scaling law of urban ride sharing has been approximated [27]. It provides a close form estimate of the percentage of rides that can be shared given a hourly demand density. These studies remain descriptive of the dynamics of the ride sharing industry. The demand for shared ride is given and none of these study focuses on the equilibrium that exists when both shared and classic are offered in a market.

We next focus our attention on economic contributions. Different model based approaches have been considered to evaluate the profitability of a ride sharing service. [24] proposes a model to estimate the benefit of introducting a ride sharing service in a market. Their analysis focuses on the impact such change would have on driver earnings and on the profit of the e-hailing platform. However, contrary to how most e-hailing platform price sharing services, this work assumes that the discounted price of a shared ride is granted to a rider only if the ride is shared. [33] studies the

equilibrium when customers are choosing among public transport, shared rides and classic taxis. This analysis is built on the assumptions that (i) shared riders can wait until a match occurs, (ii) the match rate of a e-hailing platform is an exogenous parameter, instead of modeling it as a function of shared demand density . Both of these studies assume that the average efficiency of matches is exogenous and is set to the maximum value for two party matching: 50%, i.e., the cost of a driver completing both rides is equal to the cost of a driver completing one of the two rides (assuming equal length).

3.2 Problem statement

In this paper we aim at understanding the economic feasibility of introducing shared rides into a regional market. The goal of our analysis is to provide a quantitative insight on the conditions for the existence of a profitable equilibrium with both shared and classic rides in a given region. In particular, we are interested in understanding the effect of the parameters in the existing economy (e.g., the regional demand density and its geographical distribution, the topology of the transportation network, the quality of public transit as an outside option, etc) on whether or not introducing shared ride rides into a regional market is an economical choice.

In order answer this question, we develop a model for computing the profit realized by an e-hailing platform that offers both modes.

Definition and Notations

We begin by defining notations that will be useful throughout the paper.

Trips and passengers decisions

The road network of the region studied is modeled by an arc weighted graph $\mathscr{N} = (\mathscr{V}, \mathscr{A})$. Each node in $\mathscr{V}$ represents an intersection, if two intersections corresponding to nodes $u, v \in \mathscr{V}$ are linked by a road, then $(u, v) \in \mathscr{A}$ constitutes an arc in $\mathscr{N}$. The graph $\mathscr{N}$ is endowed with a distance function $l : \mathscr{A} \to \mathbb{R}^+$. By extension and for the convenience of notation, we may use $l(u, v)$ as the shortest path distance from node u to node v in $\mathscr{N}$. Without lose of generality, we assume that drivers move on the road network with a speed of 1, which means that getting from u to v will take $l(u, v)$ units of time.

A trip in the $\mathscr{N}$ is defined by a pair of nodes $(O, D) \in \mathscr{V}^2$ and a request time t. A trip is denoted by $\tau = (O, D, t)$. By extension, we let $l(\tau) = l(O, D)$ be the travel distance associated to the trip.

Table 3.1: Notation

Road network:

$\mathscr{N}$	Road network in which drivers and passenger travels
$\mathscr{V}$	Set of nodes in $\mathscr{N}$, a node corresponds to a crossroad
$\mathscr{A}$	Set of arcs in $\mathscr{N}$, an arc corresponds to a street
l	Shortest path length between any two nodes in $\mathscr{N}$

Trip:

τ	Trip from origin O to destination D requested at t
$l(\tau)$	Shortest path length from O to D in $\mathscr{N}$
$\mathscr{T}$	Set of trips, denoted $\mathscr{T}_c$, $\mathscr{T}_s$, $\mathscr{T}_t$ for classic, shared, transit
$\mathscr{D}$	Distribution of set of trips
λ	Demand density, denoted λ_c, λ_s, λ_t for classic, shared, transit
$p_{c,s,t}$	Fare per unit of distance associated with a classic, shared, transit

Passenger choice:

i	Passenger associated to the i^{th} trip in $\mathscr{T}$
τ_i	Trip of passenger i
α_i	Time sensitivity of passenger i
F_α	Cumulative density function of the random variable α
$u^i_{c,s,t}$	Utility function of passenger i for classic, shared or transit
$d,\ D$	Discounts associated with shared and transit compared to classic
$\delta_\pi,\ \Delta$	Expected additional travel time associated with shared and transit compared to classic
$\mathbb{P}_{c,s,t}$	Probability that a user chooses classic, shared or transit
c	Fraction of the classic fare that is paid to the driver by the e-hailing platform

Shared Matching policy:

π	Matching policy
δ_π	Expected detour for matches created by the matching policy π
$\texttt{eff}_\pi$	Expected efficiency realized by matching policy π
δ_{max}	Maximum detour a shared passenger can experience
$\pi(\cdot,\ \delta_{max})$	Matching policy subject to the detour being upper bounded by δ_{max}
$\pi(\mathscr{T}_s,\ \delta_{max})$	Sets of pairs of trips returned by running $\pi(\cdot,\ \delta_{max})$ on $\mathscr{T}_s$
$l_\pi(\{\tau_i, \tau_j\})$	Total distance travelled by the driver to realize $\{\tau_i, \tau_j\}$ under π
$l^i_\pi(\{\tau_i, \tau_j\})$	Total distance travelled by i experiencing match $\{\tau_i, \tau_j\}$ under π
$\delta^i_\pi(\{\tau_i, \tau_j\})$	Detour experienced by i when matched in $\{\tau_i, \tau_j\}$ under policy π
V_π	Expected profit associated with the matching policy π

A stream of passengers arrive at the system according to some random arrival process and request a trip in $\mathscr{N}$ with respect to some distribution. We denote by i the i^{th} passenger and by τ_i the trip she requests. The set of trips requested according to this process is denoted $\mathscr{T}$ and is distributed with respect to $\mathscr{D}$. We define *demand density* λ as the average cardinality of the set of trips per unit of time.

To complete her trip τ, each passenger decides to use one of the available options. In this paper, we consider three options for the passenger while on the platform: two options provided by the platform and one outside option; namely, (i) classic ride denoted by c, (ii) shared ride denoted by s, and (iii) the outside option, e.g. public transit denoted t.

We denote by p_c, p_s and p_t the fare per unit of distance travelled for each mode. (Note that both p_c and p_s are to be decided by the platform – we shall come back to this later.) The price and the travel time associated with each option is proportional to the trip distance $l(\tau)$.

We remind the reader that we chose the scales of time and distance in a way that drivers move with the speed of one unit of distance per unit of time. Hence, the expected time spent in a classic ride is simply $l(\tau)$. However, the topology of the public transit network and the likelihood of experiencing a detour for the pickup of another shared passenger often prevent both transit and shared mode of transportation to travel from O to D along the shortest path. Consequently, it will take more time for a passenger to reach her destination using either public transit or shared ride. We let $\delta_\pi \geq 0$ be an endogenous variable and $\Delta \geq 0$ be an exogenous variable such that the expected time needed to complete trip τ_i would be $(1+\Delta)\cdot l(\tau_i)$ under public transit and $(1+\delta_\pi)\cdot l(\tau_i)$ under shared ride. We emphasize that the subscript π in δ_π denotes the matching algorithm chosen by the platform for shared rides.

We consider utility-maximizing passengers, where each passenger's utility is linear in time and money she spends on the mean of transportation. The utility passenger i associates with completing her trip τ_i is denoted as $u_0^i(\tau_i)$. This utility is large enough such that for all trip, at least one of the three modes we consider is associated with a non-negative expected utility. The time sensitivity of passenger i is a random variable denoted by α_i and distributed i.i.d. with respect to some cumulative density function F_α. For her trip τ_i, we denote by $u_c^i(\tau_i)$, $u_s^i(\tau_i)$ and $u_t^i(\tau_i)$ the expected utility associated with the modes classic, shared, and public transit, calculated as follows.

$$\begin{aligned} u_c^i(\tau_i) &= u_0^i(\tau_i) - l(\tau_i)\cdot(p_c + \alpha_i) \\ u_s^i(\tau_i) &= u_0^i(\tau_i) - l(\tau_i)\cdot(p_s + \alpha_i\cdot(1+\delta_\pi)) \\ u_t^i(\tau_i) &= u_0^i(\tau_i) - l(\tau_i)\cdot(p_t + \alpha_i\cdot(1+\Delta)) \end{aligned}$$

Each passenger chooses the option that maximizes her utility. We denote by $\mathscr{T}_c$, $\mathscr{T}_s$ and $\mathscr{T}_t$ the resulting partition of $\mathscr{T}$ over the three modes of transportation. We also denote by λ_c, λ_s and λ_t the demand density for each mode.

Shared rides characteristics

E-hailing platforms attempt to minimize the operating costs of shared ride while providing a good user experience to retain customers on the platform. Unlike a classic rider, a shared rider can share her trip with another shared rider. This leads to a potentially more efficient transportation platform. In return, the platform offers a discount to the shared rider that we denote by $d \in [0, 1]$, leading to a lower price per unit of distance travelled for shared rides compared to classic rides. In other words, $p_s = p_c \cdot (1 - d)$.

The e-hailing platforms pay drivers based on the time and the distance travelled with passengers. However, as a result of our constant speed assumption, we can focus only on the time component. We denote by $c \cdot p_c$ the driver pay per unit of distance travelled, where $c \in [0, 1]$. One can think of $1 - c$ as the platform's commission percentage in classic rides.

The assumption of perfect supply refers to a situation where a driver is always available at a passenger's pickup location. Under this assumption, there is no cost incurred for driving to pickup a passenger at the beginning of the trip and no additional waiting time once a driver has been dispatched to a passenger. The positioning and dispatch of available drivers to pick up passengers is a crucial challenge for e-hailing platforms. However, we would like to focus our attention on the effect of the quality and pricing of shared ride on the economy. Therefore, in our model we work with the perfect supply assumption.

When two passengers i and j share a vehicle such that a single driver is needed to complete both trips τ_i and τ_j, we say that i and j experience a *match*. We denote this match by $\{\tau_i, \tau_j\}$. A matching is a set of pairs of shared trips, i.e. matches, such that every trip is in a pair and the intersection of two pairs is empty. Note the pair $\{\tau_i, \tau_i\}$ corresponds to leaving trip τ_i unmatched.

A matching policy π is a set of constraint and an algorithm to compute a matching for a given set of trips. The set of constraint models the restrictions that apply to the matching policy. Perhaps the most natural among such restriction is that of online matching policies which asserts that they cannot use knowledge of future ride requests (although, they may use distributional information regarding that, should such information be available). Finally, with a slight abuse of notation, for any match pair of trips $\{\tau_i, \tau_j\}$ under a given matching policy π, we denote by $l_\pi(\{\tau_i, \tau_j\})$ the distance travelled by the *driver* in order to complete the requests τ_i and τ_j should

these two trips be matched together under π. Similarly, let $l^i_\pi(\{\tau_i, \tau_j\})$ denote the total distance travelled by passenger i when matched to j under the matching policy π should these two trips be matched together.

With a slight abuse of notation, let $\pi(\mathscr{T}_s)$ denote the matching returned by the matching policy π for the set of shared trips $\mathscr{T}_s$; in other words, $\pi(\mathscr{T}_s)$ denotes the set of pairs among shared rides $\mathscr{T}_s$ matched together by the matching policy π. url adjustbox xcolor enumerate geometry amsmath amssymb textcomp graphicx subcaption [utf8]inputenc tabularx,tabulary multirow

In order to show these notations in action, let us consider a policy called "optimal offline". Since this is an offline policy, it has the complete knowledge about all the trips. It quantifies how efficient any pair of trips can be matched together. It then finds a maximum weighted matching in a graph where the nodes correspond to the trips and the edge weights demonstrate the efficiency of the two corresponding end-nodes. Let us denote this policy by π^*. For any pair of trips τ_i and τ_j, ignoring the request time of each trip, the match $\{\tau_i, \tau_j\}$ is associated with one of the four orderings of origin and destinations of the two trips. Note that each passenger goes from an origin to a destination and that they only share the trip if the second passenger is picked up before the first passenger is dropped off. The four possible orderings are: (O_i, O_j, D_i, D_j), (O_i, O_j, D_j, D_i), (O_j, O_i, D_j, D_i) and (O_j, O_i, D_i, D_j).

Before providing the formula for calculating l_{π^*} and for the sake of completeness, we must note that an offline policy – by the virtue of its clairvoyance – can potentially route a driver (possibly with the first passenger already in the car) towards the pick-up point of a passenger who has not showed up yet.[1] The car may even reach the pick-up point of the second passenger before she shows up. The time that the driver (along with the first passenger) is waiting at the pick-up point of the second passenger is a part of the trip. Without loss of generality, let us assume that the first passenger is the one in τ_i and the second passenger belongs to τ_j. (This happens in two of the cases, namely (O_i, O_j, D_i, D_j) and (O_i, O_j, D_j, D_i)). We denote by $\omega^\pi_{i,j}$ the waiting time of passenger i at the pick-up point of passenger j before j shows up under a matching policy π. One can see that for the optimal offline policy we have

$$\omega^{\pi^*}_{i,j} = (t_j - (t_i + w + l(O_i, O_j))^+. \tag{3.1}$$

Here, w denotes the maximum time allowed between the arrival time of a request and its pick-up time. Also, $(t_i + w + l(O_i, O_j)$ corresponds to the latest time passenger i could reach to the pick-up point O_j. Note that if passenger i does not reach O_j before t_j, then $\omega^{\pi^*}_{i,j}$ will be zero.

[1] We emphasize that this is by no means limited to the offline policy; an online policy that has access to good enough statistics regarding the requests in the future – such as a concert ending in a specific venue – may route preemptively as well.

Now, we are ready to present the formula for l_{π^*}, the distance of a possible match $\{\tau_i, \tau_j\}$ under the optimal offline policy.

$$l_{\pi^*}(\{\tau_i, \tau_j\}) = \min \left\{ \begin{array}{l} l(O_i, O_j) + \omega_{i,j}^{\pi^*} + l(O_j, D_i) + l(D_i, D_j); \\ l(O_i, O_j) + \omega_{i,j}^{\pi^*} + l(O_j, D_j) + l(D_j, D_i); \\ l(O_j, O_i) + \omega_{j,i}^{\pi^*} + l(O_i, D_j) + l(D_j, D_i); \\ l(O_j, O_i) + \omega_{j,i}^{\pi^*} + l(O_i, D_i) + l(D_i, D_j). \end{array} \right\}$$

Here, each of the four possibilities correspond to (i) the distance between the first and second pick-up points, (ii) the possible wait at the second pick-up point, (iii) the distance between the second pick-up point and the first drop-off point, and finally (iv) the distance between the first and second drop-off point, for the specific ordering. Note that the total distance of the match $l_\pi(\{\tau_i, \tau_j\})$ is lower bounded by $\max\{l(\tau_i);\ l(\tau_j)\}$.

We denoted by d the percentage reduction the classic fare to obtain the shared fare: $p_s = p_c \cdot (1-d)$. Similarly, given a matching policy π, we call efficiency $\texttt{eff}_\pi$ the function that maps a match onto the cost reduction compared to classic. For instance, in the match $\{\tau_i, \tau_j\}$, the cost induced by passenger i is: $l(\tau_i) \cdot c \cdot p_c \cdot [1 - \texttt{eff}_\pi(\{\tau_i, \tau_j\})]$, where $l(\tau_i) \cdot c \cdot p_c$ is the cost of serving τ_i as a classic ride.

$\texttt{eff}_\pi(\{\tau_i, \tau_j\})$ is the ratio of the cost savings realized with the match, i.e., $l(\tau_i) + l(\tau_j) - l_\pi(\{\tau_i, \tau_j\})$, by the cost of serving both rides as classic ride, i.e., $l(\tau_i) + l(\tau_j)$.

$$\texttt{eff}_\pi(\{\tau_i, \tau_j\}) = 1 - \frac{l_\pi(\{\tau_i, \tau_j\})}{l(\tau_i) + l(\tau_j).}$$

As $l_\pi(\{\tau_i, \tau_j\}) \geq \max\{l(\tau_i); l(\tau_j)\}$ and the platform can only match two passenger together, the efficiency ratio is upper bounded by 50%. We also note that by summing up the cost induced by each passenger in the match $\{\tau_i, \tau_j\}$, we recover the cost of the match:

$$l(\tau_i) \cdot c \cdot p_c \cdot [1 - \texttt{eff}_\pi(\{\tau_i, \tau_j\})] + l(\tau_j) \cdot c \cdot p_c \cdot [1 - \texttt{eff}_\pi(\{\tau_i, \tau_j\})] = c \cdot p_c \cdot [l(\tau_i) + l(\tau_j)] \cdot \frac{l_\pi(\{\tau_i, \tau_j\})}{l(\tau_i) + l(\tau_j)}.$$

A shared rider, when matched, can experience a detour since her trip gets longer. Under a matching policy π, we define as δ_π^i the function that maps a match onto the ratio of additional travel distance for passenger i by the shortest distance from origin to destination $l(\tau_i)$. For example, consider the match $\{\tau_i, \tau_j\}$ under the policy π, we have:

$$\delta_\pi^i(\{\tau_i, \tau_j\}) = \frac{l_\pi^i(\{\tau_i, \tau_j\})}{l(\tau_i)} - 1$$

Note that similarly to discount and efficiency, we can use the detour function to compute the expected shared travel time as a function of the expected classic travel time: $l(\tau_i) \cdot [1 + \delta^i_\pi(\{\tau_i, \tau_j\})]$.

To maintain reasonable travel time for a shared rider, a constraint is set to keep the detour experience by each passenger under some maximum allowed value, denoted by δ_{max}. A matching policy subject to such detour constraint is denoted by $\pi(\cdot,\ \delta_{max})$ and the set of pairs among shared rides $\mathscr{T}_s$ matched together by such policy is denoted $\pi(\mathscr{T}_s,\ \delta_{max})$.

The average shared ride detour with respect to a policy is a function that maps a shared demand density λ_s and the maximum allowed detour δ_{max} onto the expected detour experienced by a shared rider δ_π for the matching solution obtained with the matching policy $\pi(\cdot,\ \delta_{max})$.

$$\delta_\pi(\lambda_s, \delta_{max}) = \frac{1}{\lambda_s}\mathbb{E}_{\mathscr{T}_s \sim \mathscr{D}}\left[\sum_{\{\tau_i,\tau_j\} \in \pi(\mathscr{T}_s,\ \delta_{max})} \delta^i_{\pi(\cdot,\ \delta_{max})}(\{\tau_i, \tau_j\}) + \delta^j_{\pi(\cdot,\ \delta_{max})}(\{\tau_i, \tau_j\}) \middle| |\mathscr{T}_s| = \lambda_s\right]$$

Similarly, the average efficiency of a shared ride with respect to a policy is a function that maps a demand density and a maximum detour δ_{max} onto the expected efficiency associated to a shared rider for the matching solution obtained with $\pi(\cdot,\ \delta_{max})$.

$$\texttt{eff}_\pi(\lambda_s, \delta_{max}) = \frac{1}{\lambda_s}\mathbb{E}_{\mathscr{T}_s \sim \mathscr{D}}\left[\sum_{\{\tau_i,\tau_j\} \in \pi(\mathscr{T}_s,\ \delta_{max})} l_{\pi(\cdot,\ \delta_{max})}(\{\tau_i, \tau_j\}) \Big/ \sum_{\tau_i \in \mathscr{T}_s} l(\tau_i) \middle| |\mathscr{T}_s| = \lambda_s\right]$$

Using the definition of shared discount and average detour, we note $p_s = (1-d) \cdot p_c$. The expected utility of user i for shared mode becomes:

$$u^i_s(\tau_i) = u^i_0(\tau_i) - l(\tau_i) \cdot \big(p_c \cdot (1-d) + \alpha_i \cdot \big[1 + \delta_\pi(\lambda_s, \delta_{max})\big]\big)$$

Note that the expected utility of a user for shared is the solution of a complex fixed point equation as the shared demand density λ_s is a function of the distribution of shared expected utility: $\lambda_s = \lambda \cdot \mathbb{P}\big(u_s(\tau) \geq \max\{u^i_t(\tau);\ u_c(\tau)\}\big)$

Alternative option characteristics

We consider the existence of a public transit option that is the cheapest yet slowest option. By comparing this option to a classic ride, similar to our formulation for the characteristics of shared rides, we define D and Δ as the average discount and average detour associated with choosing public transit.

As for shared ride, using the above definitions, we note $p_t = p_c \cdot (1 - D)$. The utility of a user for public transit becomes:

$$u_t^i(\tau_i) = u_0^i(\tau_i) - l(\tau_i) \cdot \big(p_c \cdot (1 - D) + \alpha \cdot (1 + \Delta)\big)$$

Model

Our model estimates the expected profit the company can realize with a choice of discount d and maximum allowed detour a user can experience.

Modeling profit

Profit is a function V_π that maps the shared ride's discount d and maximum allowed detour δ_{max} selected by the e-hailing platform onto its expected profit under some matching policy π. For the sake of simplicity, let l denote the average shortest path distance from origin to destination of a trip: $l = \mathbb{E}_\tau\{l(\tau)\}$. We assumed that the trip requested by a passenger is independent of the mode she selects, consequently the average distance l is an exogenous parameter.

A classic trip generates per unit of distance a revenue of p_c and a cost of $c \cdot p_c$. By summation, the expected profit per classic ride is $(1 - c) \cdot p_c \cdot l$. Similarly, a shared trip generates a revenue of $p_c \cdot (1 - d)$ per unit of distance.

Given a matching policy $\pi(\cdot,\ \delta_{max})$ and a shared demand density λ_s, by definition of expected efficiency $\texttt{eff}_\pi$, a shared trip generates an expected cost of $c \cdot p_c \cdot [1 - \texttt{eff}_\pi(\lambda_s, \delta_{max})]$ per unit of distance. By summation, the expected profit per shared ride is:

$$\big[1 - d - c \cdot (1 - \texttt{eff}_\pi(\lambda_s, \delta_{max})\big] \cdot p_c \cdot l$$

Summing up the expected profit for both shared and classic rides multiplied by their respective demand densities, we compute the expected profit of the e-hailing company:

$$\begin{aligned} V_\pi(d, \delta_{max}) &= p_c \cdot l \cdot \mathbb{E}_{\lambda_c,\lambda_s}\Big\{\lambda_c \cdot (1 - c) + \lambda_s \cdot \big[1 - d - c \cdot (1 - \texttt{eff}_\pi(\lambda_s, \delta_{max})\big]\Big\} \\ &= p_c \cdot l \cdot \mathbb{E}_{\lambda_c,\lambda_s}\Big\{(\lambda_c + \lambda_s) \cdot (1 - c) + \lambda_s \cdot \big[c \cdot \texttt{eff}_\pi(\lambda_s, \delta_{max}) - d\big]\Big\} \end{aligned}$$

here the distribution of λ_s and λ_c is a function of the utility model and the distribution of $\mathscr{T}$, hence a function of $\delta_\pi(\lambda_s, \delta_{max})$ and d.

As $\mathscr{D}$ is not known, it is not possible to solve this expectation.

We will build an estimate of this expectation to approximate the expected profit.

Modeling demand

In this section, we focus on the decisions of passengers conditioned on the total demand density λ. We assume the average detour function $\delta_\pi(\lambda_s, \delta_{max})$ is known to the customers prior to making their decision. This assumption can be supported by the customers acquiring a sense of average detour by spending some time on the platform.

Given the shared demand density, the probability that a passenger chooses classic, shared or transit is a function of the shared demand density λ_s, the maximum allowed detour δ_{max} and the shared discount d. We note these probability respectively $\mathbb{P}_c$, $\mathbb{P}_s$ and $\mathbb{P}_t$.

As each user chooses the option that maximizes her utility function, user i will chose classic ride if: $u_c^i(\tau_i) > u_t^i(\tau_i)$ and $u_c^i(\tau_i) > u_c^i(\tau_i)$. Given the shared demand density, this event occurs if $\alpha_i > \frac{D}{\Delta} p_c$ and $\alpha_i > \frac{d}{\delta_\pi(\lambda_s, \delta_{max})} p_c$. Therefore,

$$\mathbb{P}_s(\lambda_s, \delta_{max}, d) = 1 - F_\alpha\left(\max\left\{\frac{D}{\Delta}; \frac{d}{\delta_\pi(\lambda_s, \delta_{max})}\right\} p_c\right).$$

Similarly, user i will choose public transit if $u_t^i(\tau_i) > u_c^i(\tau_i)$ and $u_t^i(\tau_i) > u_s^i(\tau_i)$. Conditioned on the shared demand density, this event occurs if $\alpha_i \leq \frac{D}{\Delta} p_c$ and $\alpha \leq \frac{D-d}{\Delta - \delta_\pi(\lambda_s, \delta_{max})} p_c$. As a result,

$$\mathbb{P}_t(\lambda_s, \delta_{max}, d) = F_\alpha\left(\min\left\{\frac{D}{\Delta}; \frac{D-d}{\Delta - \delta_\pi(\lambda_s, \delta_{max})}\right\} p_c\right).$$

Finally user i will choose shared ride if $u_s^i(\tau_i) > u_c^i(\tau_i)$ and $u_s^i(\tau_i) > u_t^i(\tau_i)$. The caveat is that if $\frac{D-d}{\Delta - \delta_\pi(\lambda_s, \delta_{max})} > \frac{d}{\delta_\pi(\lambda_s, \delta_{max})}$, then for any customer quality sensitivity α_i, the shared ride option is dominated by either the classic ride option or by public transit and as a result, there will be no demand for shared rides. Hence,

$$\mathbb{P}_s(\lambda_s, \delta_{max}, d) = \left[F_\alpha\left(\frac{d}{\delta_\pi(\lambda_s, \delta_{max})} p_c\right) - F_\alpha\left(\frac{D-d}{\Delta - \delta_\pi(\lambda_s, \delta_{max})} p_c\right)\right]^+.$$

Conditioned on the total demand density λ, the shared demand density will be a solution of the equation

$$\lambda_s = \lambda \cdot \mathbb{P}_s(\lambda_s, \delta_{max}, d).$$

This establishes that the shared demand density conditioned on the total demand density is a function of the total demand density λ, the maximum allowed detour δ_{max} and the shared discount d.

As presented in the model, there are two key parameters – chosen by the platform – that affect the expected profit of shared ride: (i) the shared discount d and (ii) the maximum allowed detour δ_{max}. The expected shared profit decreases when d increases, however increasing δ_{max} increases the shared demand by raising the expected utility. On the other hand, increasing the maximum allowed detour δ_{max} increases the expected unit profit of shared rides while decreasing the shared demand.

The maximum allowed detour is a constraint of the matching policy, thus relaxing the constraint by increasing δ_{max} increases $\texttt{eff}_\pi$. However, the average detour δ_π is also an increasing function of δ_{max}, thus increasing δ_{max} reduces the expected shared utility.

The platform must find the optimal balance between these parameters. To study the profitability of shared rides in a given region of interest, we analyse for what ranges of d and δ_{max} the expected profit increases by introducing shared rides to the market, i.e. $V(d, \delta_{max}) > V(0, \cdot)$.

We emphasize that the expected profit as described above is intractable as it needs solving a discontinuous fixed point equation. The rest of the paper presents the estimators we build for $\delta_\pi(\lambda_s, \delta_{max})$, $\texttt{eff}_\pi(\lambda_s, \delta_{max})$ and $P(d, \delta_{max})$.

3.3 Estimating detour and efficiency

The expected detour δ_π and the expected efficiency $\texttt{eff}_\pi$ functions are intractable to compute for all matching policy π and distribution of set of trips $\mathscr{D}$.

In this section, we develop estimates for the expected detour δ_π and expected efficiency $\texttt{eff}_\pi$ functions.

We have called π a matching policy and for any given set of shared trips $\mathscr{T}_s$, we have denoted by $\pi(\mathscr{T}_s)$ the set of matches realized by π.

Throughout this section, we use π to establish a relationship between the shared demand density λ_s, the maximum allowed detour δ_{max}, the efficiency $\texttt{eff}_\pi$ and the average detour δ_π. For samples of sets of trips $(\mathscr{T}_s^k)$ and a matching policy π, we will compute Monte Carlo estimates of the efficiency $\texttt{eff}_\pi$ and the average detour δ_π.

Due to different constraints on waiting and travelling time, the description of matching policies is not trivial, it is covered in Appendix B.

Demand and trip set distribution

Since the distribution of trips and the utility are independent, we assume that a uniform sample of λ trips from a data set of taxi requests is distributed with respect

to the distribution of set of trips $\mathscr{D}$ conditioned on the cardinality of the set being λ.

The average detour $\delta_\pi(\lambda_s, \delta_{max})$ and the average efficiency $\texttt{eff}_\pi(\lambda_s, \delta_{max})$ are conditional expectations of some random variable with respect to the distribution of set of trips $\mathscr{D}$ conditioned on the cardinality of the set being λ.

We sample N sets of taxi trips $(\mathscr{T}_s^k)_{k\in\{1,\dots,N\}}$ of cardinality λ_s. By averaging the detours and efficiencies, we build Monte Carlo estimates $\hat{\delta_\pi}(\lambda, \delta_{max})$ and $\hat{\texttt{eff}}_\pi(\lambda, \delta_{max})$

$$\begin{aligned}
\hat{\delta_\pi}(\lambda, \delta_{max}) &= \frac{1}{\lambda_s \cdot N} \sum_{\substack{k\in\{1,\dots,N\} \\ \{\tau_i,\tau_j\}\in\pi(\mathscr{T}_s^k,\ \delta_{max})}} \left[\delta^i_{\pi(\cdot,\ \delta_{max})}(\{\tau_i,\tau_j\}) + \delta^j_{\pi(\cdot,\ \delta_{max})}(\{\tau_i,\tau_j\})\right] \\
\hat{\texttt{eff}}_\pi(\lambda, \delta_{max}) &= \frac{1}{\lambda_s \cdot N} \sum_{k\in\{1,\dots,N\}} \left[\sum_{\{\tau_i,\tau_j\}\in\pi(\mathscr{T}_s^k,\ \delta_{max})} l_{\pi(\cdot,\ \delta_{max})}(\{\tau_i,\tau_j\}) \Big/ \sum_{\tau_i\in\mathscr{T}_s^k} l(\tau_i) \right]
\end{aligned}$$

For tractability reasons, we cannot build such Monte Carlo estimates for all units of shared demand density and all possible values of the maximum allowed detour δ_{max}. Hence we propose to approximate the expected detour and efficiency by fitting a function which closed form is known on the Monte Carlo estimates. The parameters of such the closed form estimates are fitted by minimizing the squared distance between the estimate and average detour and efficiency over each sampled data set.

3.4 Estimating the profit function

In this section, we build an estimate of the expected profit of a e-hailing platform using the matching policy $\pi(\cdot,\ \delta_{max})$.

We first estimate the expected profit conditioned on the total demand density λ. From the matching simulations, we have obtained Monte Carlo estimates of the average efficiency and the average detour conditioned on the shared demand density λ_s. Conditioning on the total demand density λ and using $\hat{\delta}_{\pi(\cdot,\ \delta_{max})(\lambda_s,\delta_{max})}$, we let $\hat{\lambda}_s$ and $\hat{\lambda}_c$ be the estimated demand densities.

We define $\hat{V}_{\pi(\cdot,\ \delta_{max})(d,\delta_{max},\lambda)}$ as the estimate of the expected profit conditioned on the total demand density λ.

$$\hat{V}_\pi(d, \delta_{max}, \lambda) = p_c \cdot l \cdot \left((\hat{\lambda}_c + \hat{\lambda}_s) \cdot (1-c) + \hat{\lambda}_s \cdot \left[c \cdot \hat{\texttt{eff}}_\pi(\hat{\lambda}_s, \delta_{max}) - d\right]\right)$$

3.5 Numerical analysis

Our first analysis focuses on the regional market of Manhattan. We study the peak hours on Monday morning between 8 A.M. and 9 A.M. Our goal is to provide quantitative insights on whether or not there is a profitable market for both classic and shared ride in Manhattan during peak hours.

This analysis is based on publicly available data such as Open Street Map [23] for the road network, New York Yellow Taxi Dataset for the distribution of sets of trips, MTA data [22] for the demand density.

Estimating detour and efficiency

The matching policy plays a crucial role in determining the discount and maximum detour a market is suited for and it directly impacts the expected efficiency and the estimated profit a e-hailing company can realize. To increase the relevance of this analysis, we compare three matching policies described in Appendix B: *Greedy batching*, *Vertex additive* and *Clairvoyant offline*.

Clairvoyant offline solves the matching problem to optimality with perfect knowledge of all trip requests. It sets an upper bound on the expected profit of the regional market.

Greedy batching batches trip requests for a short period of time then myopically matches the available requests. This policy can be implemented by a e-hailing company, it thus sets a lower bound on the expected profit of the regional market.

Vertex additive is a matching policy developed to solve a dynamic matching problem under limited time [1]. This policy uses the spatial distribution of trip requests to set spatial minimum efficiencies. Similarly to Greedy batching, this policy batches trip requests for a short period of time. It then matches the available requests subject to the constraint of minimum efficiency. This policy is meant to be comparable with the proprietary algorithms of e-hailing companies.

In the model, we assume the trip request and the choice of mode of transportation are independent. Consequently, a sample of n trips from the New York Taxi Data is distributed with respect to $\mathscr{D}$ knowing that the demand density is n.

Estimates for expected detour and efficiency are built on sets of trips sampled from a data set of New York yellow taxi trips that occurred on Monday morning during peak hours.

For each maximum detour value studied and each demand density of interest, we sample 40 sets of trips and compute an estimate of detour and efficiency.

Estimates of expected efficiency

The Monte Carlo estimates of efficiency for each policy are presented in Figure 3.1 with the policies from left to right in this order: Greedy batching, Vertex additive and Clairvoyant offline. The value of the maximum detour δ_{max} is reported on the y-axis, while the shared demand density λ_s is reported on the x-axis. We note that as the demand density increases and the detour constraint relaxes, the expected efficiency increases to reach an asymptotic maximum. As expected, the efficiency of the Vertex additive policy falls between the one of Greedy batching and Clairvoyant offline.

Figure 3.1 shows a common structure: efficiency gain with demand density or maximum detour value is decreasing exponentially. We use a closed form estimates built on three parameters to capture this structure. By a slight abuse of notation, we denote the first parameter by $\texttt{eff}_\pi^\infty$. It models the maximum expected efficiency reachable by the matching policy when both maximum detour δ_{max} and shared discount d go to infinity. The two other parameters are denoted θ_π^λ (resp. θ_π^δ), modeling the rate at which expected efficiency gain is decreasing with demand density (resp. maximum detour value). We propose the following close form estimate:

$$\hat{\texttt{eff}}_\pi(\lambda_s, \delta_{max}) = \texttt{eff}_\pi^\infty \cdot (1 - \exp\{-\theta_\pi^\lambda \cdot \lambda_s\}) \cdot (1 - \exp\{-\theta_\pi^\delta \cdot \delta_{max}\}) \tag{3.2}$$

Figure 3.2 shows these closed form estimates fitted on the Monte Carlo estimates. We provide in Figure 3.3 the mean squared error associated with each closed form estimate. The policy Greedy batching, Vertex additive and Clairvoyant offline are presented from left to right in that order.

Similarly, the Monte Carlo estimates of expected detour for each policy are presented in Figure 3.4. The expected detour for a match realized by a matching policy π has been defined as a function of shared demand density λ_s and detour constraint value δ_{max}. The caveat with this formulation is that the shared demand density λ_s, conditioned on the demand density λ, is the solution of a fixed point equation:

$$\lambda_s = \lambda \cdot \left(F_\alpha\left(\frac{d}{\delta_\pi(\lambda_s, \delta_{max})} p_c\right) - F_\alpha\left(\frac{D-d}{\Delta - \delta_\pi(\lambda_s, \delta_{max})} p_c\right)\right)^+$$

Figure 3.4 shows that demand density has a limited impact on the estimates of expected detours. Its impact appears to be restricted to lower ranges of shared demand density. Such ranges of demand density are likely to not generate enough efficiency for shared ride to be profitable. Note that ignoring the impact of demand density on the expected detour introduces a positive bias on expected detour for low density thus making the mode less attractive to users. Consequently, such a

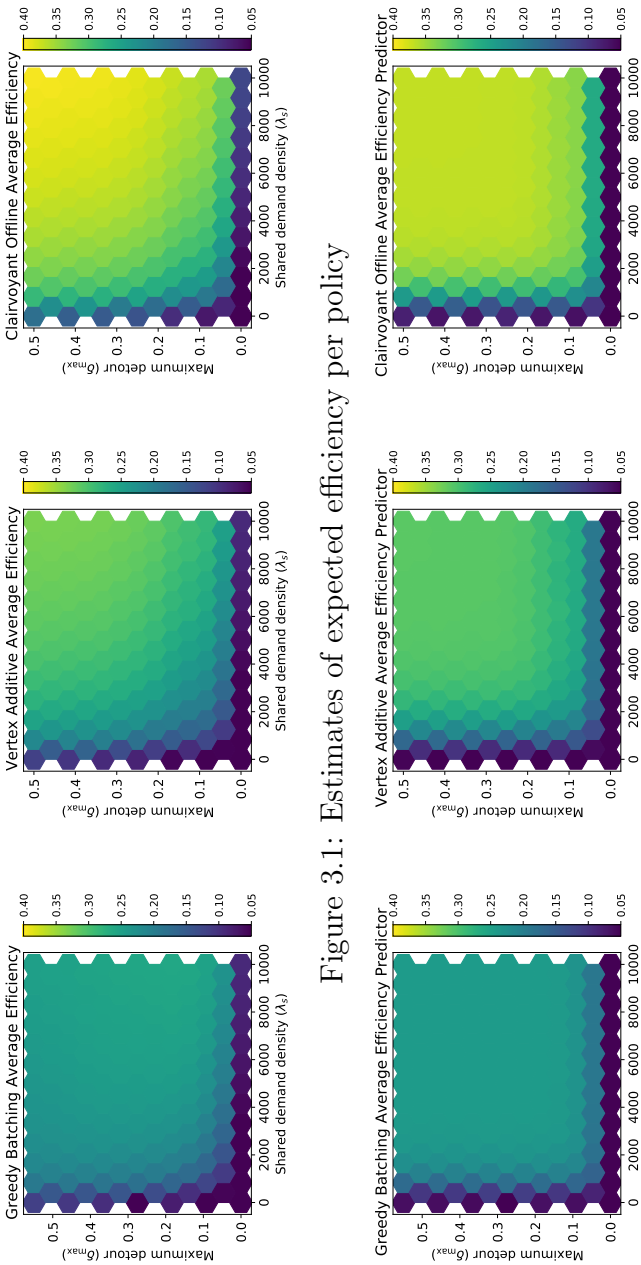


Figure 3.1: Estimates of expected efficiency per policy

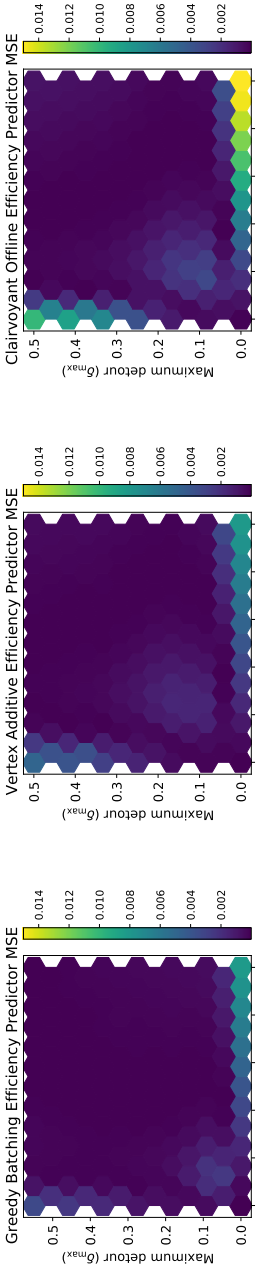


Figure 3.2: Fitted closed form estimates of expected efficiency per policy

Figure 3.3: Mean squared error of the fitted estimates of expected efficiency per policy

simplification does not make low shared density ranges seem more profitable in our estimate of profit than they really is. Therefore, to simplify the equation of expected profit, we propose to use a function of the detour constraint δ_{max} only as a closed form formulation of the estimate of expected detour. Similarly to the structure of efficiency, we note that the marginal increase of expected detour decreases exponentially with the maximum detour value. We use a closed form estimates built on two parameters to capture this structure. Again with a slight abuse of notation, we denote the first parameter by δ_π^∞. It models the maximum expected detour reached by a matching policy π when the maximum detour value δ_{max} goes to infinity. The second parameter, i.e., the rate at which marginal gain decreases exponentially, is denoted γ_π^δ:

$$\hat{\delta_\pi}(\delta_{max}) = \delta_\pi^\infty \cdot (1 - \exp\{-\gamma_\pi^\delta \cdot \delta_{max}\})$$

Figure 3.5 shows such closed form estimate fitted, along with the mean squared error on Figure 3.6. On both figures, policies Greedy batching, Vertex additive and Clairvoyant offline are presented from left to right in this order.

Demand density

A closed form formulation of the demand density for each mode can be computed from the estimate of expected detour. We are specifically interested in the classic and shared demand density conditioned on the trip demand density to compute the profit:

$$\lambda_s(\lambda, \delta_{max}) = \lambda \cdot \left(F_\alpha\left(\frac{d}{\hat{\delta_\pi}(\delta_{max})} p_c\right) - F_\alpha\left(\frac{D-d}{\Delta - \hat{\delta_\pi}(\delta_{max})} p_c\right)\right)^+$$

$$\lambda_c(\lambda, \delta_{max}) = \lambda \cdot \left(1 - F_\alpha\left(\max\left\{\frac{d}{\hat{\delta_\pi}(\delta_{max})};\ \frac{D}{\Delta}\right\} \cdot p_c\right)\right)$$

For this analysis, we assume that the time sensitivity of passenger, denoted α is distributed with respect to an exponential distribution of parameter . At this stage, this assumption is not yet backed by thorough data analyses, following research will focus on developing more realistic utility functions. Note that our goal is to provide a framework to e-hailing platform's managers so they can take decisions driven by quantitative insights. As e-hailing platforms survey their users, they learn the distribution of their time sensitivity and can use this information in the present framework.

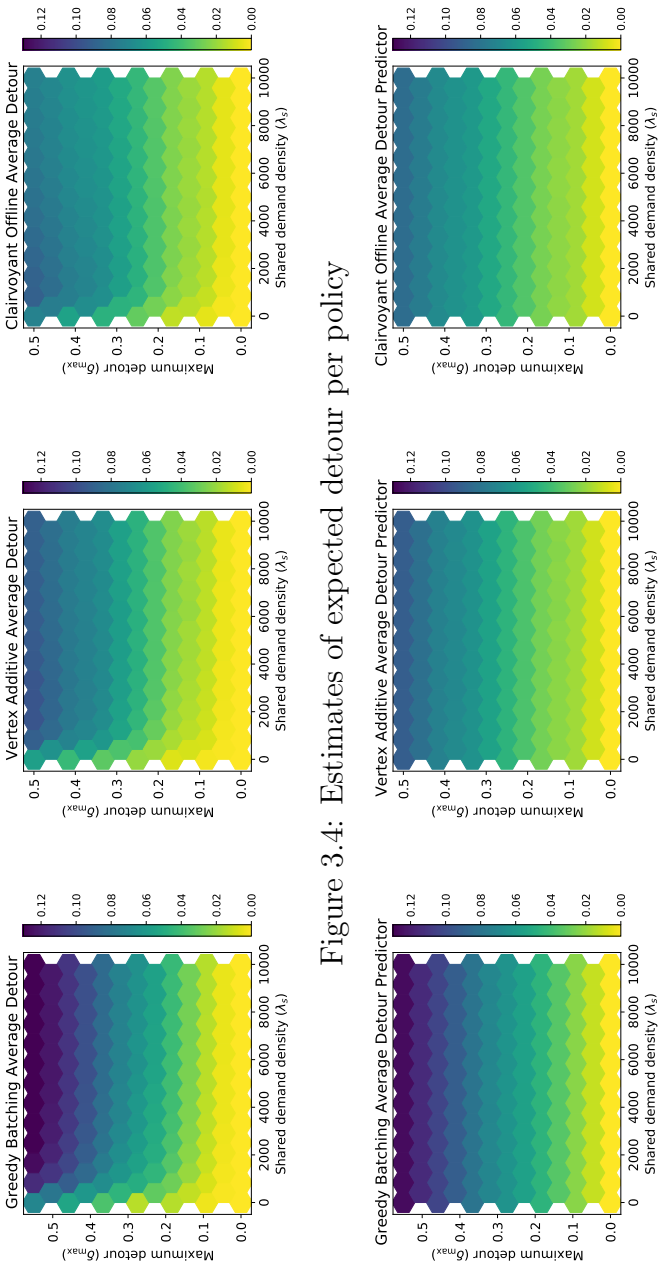


Figure 3.4: Estimates of expected detour per policy

Figure 3.5: Fitted closed form estimates of expected detour per policy

Figure 3.6: Mean squared error of the fitted estimates of expected detour per policy

$$\lambda_s(\lambda, \delta_{max}) = \lambda \cdot \left(\exp\left\{ -\cdot \frac{D-d}{\Delta - \hat{\delta_\pi}(\delta_{max})} p_c \right\} - \exp\left\{ \cdot \frac{d}{\hat{\delta_\pi}(\delta_{max})} p_c \right\} \right)^+$$

$$\lambda_c(\lambda, \delta_{max}) = \lambda \cdot \left(\exp\left\{ \max\left\{ \frac{d}{\hat{\delta_\pi}(\delta_{max})}; \frac{D}{\Delta} \right\} \cdot p_c \right\} \right)$$

Figure 3.7 shows the probability that a passenger selects any of the three modes as a function of the discount d. The maximum detour value δ_{max} is set to 40% and the estimate of expected profit of the Vertex additive policy is used in place on $\hat{\delta_\pi}$.

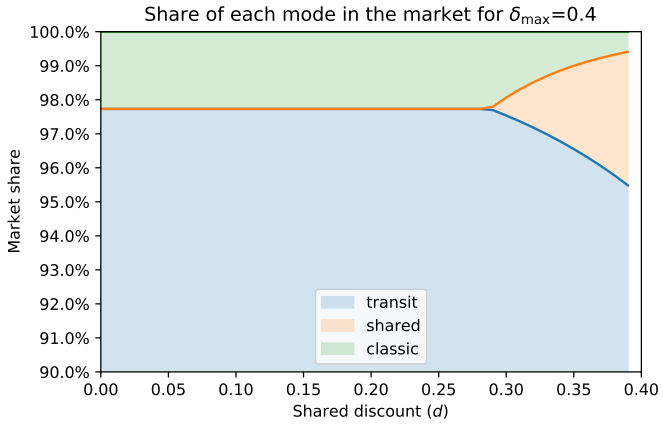
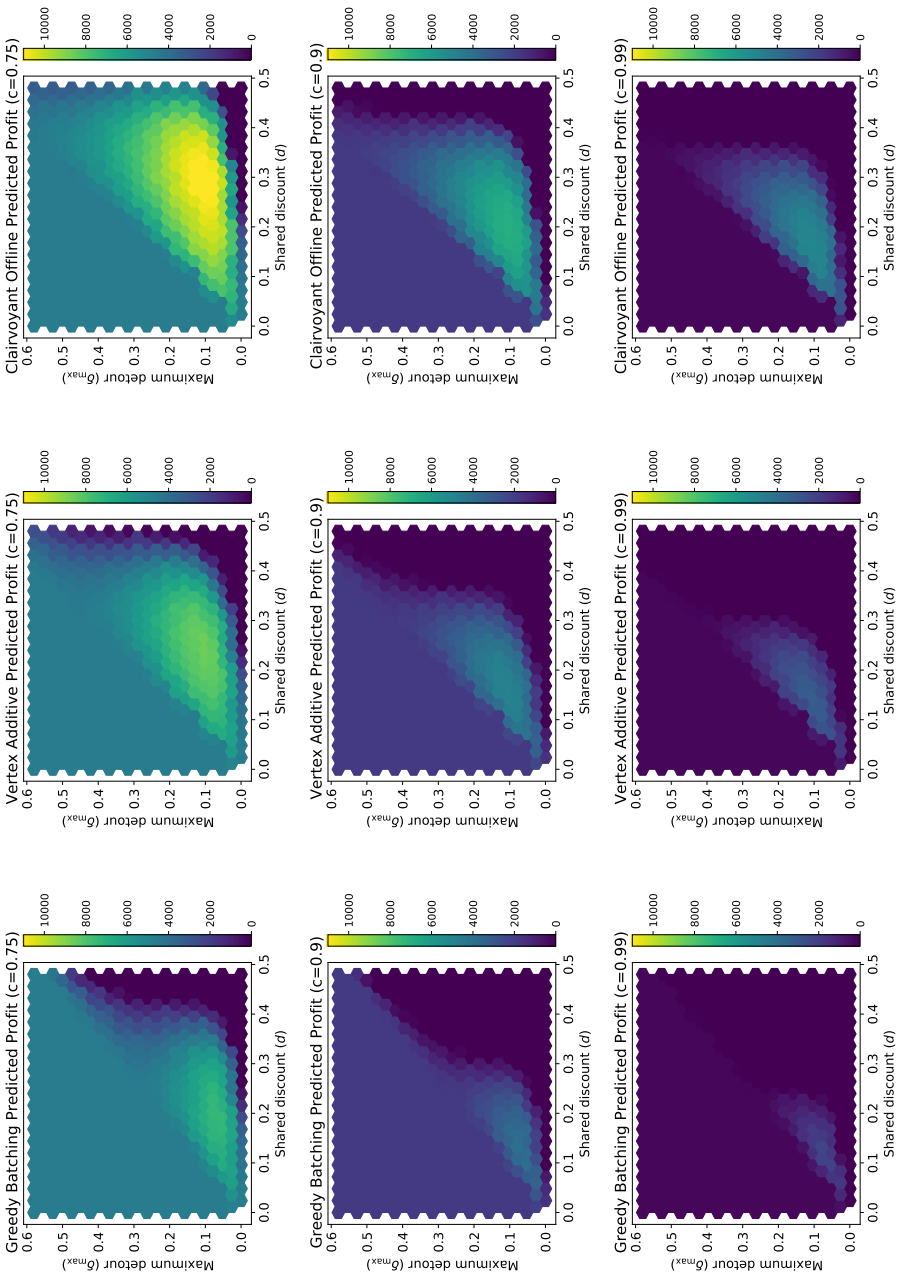


Figure 3.7: Market share of each mode as a function of discount and detour constraint

Profit estimates

We are now equipped to compute an estimate of the profit. During Monday morning peak hours, entry data from the MTA subway [22] indicates that nearly a million people are looking for a mean of transportation. Assuming for instance that the supply cost c is of 75%, Figure 3.8 reports the expected profit as a function of the shared discount d on the x-axis and the detour constraint δ_{max} on the y-axis. In each subfigures of Figure 3.8, the top left triangle shows a constant profit value. For this range of discount and detour constraint, the demand density for shared ride is negligible and the profit is generated by classic rides. To strengthen the relevance of this study, we run sensitivity analysis with respect to the supply cost c, Figure 3.8 also shows the expected profit for $c = 0.9$ and $c = 0.99$.



(a) Greedy Batching (b) Vertex Additive (c) Offline

Figure 3.8: Predicted profit for each policy on Manhattan and driver cost of 75%, 90% and 99%

The numerical analysis indicates there is a profitable market for shared rides in Manhattan during Monday morning peak hours. This observation holds even for the weaker Greedy batching policy. This suggests that even when unsure about the quality of its matching policy, having shared rides in Manhattan during morning commute hours is probably still worthwhile for an e-hailing platform. Note that this regional market is unique and has high expected efficiency even for the Greedy batching policy, such statement may not be possible in other markets or during another time of the day.

3.6 Conclusion

In this Chapter, we proposed a framework to study the existence of a profitable equilibrium with shared and classic rides in a regional market. As the general formulation is intractable to solve, we developed a simulation engine to estimate quantities such as expected efficiency and expected detour. We ran the analysis on Manhattan regional market during morning commute hours and the results indicate that there is a profitable equilibrium for this market.

This project is part of an ongoing effort to understand the interaction between detour and discount in the ride sharing industry. Here are some of the extension we will be working on:

The topology of the road network and the demand pattern have a direct impact on the efficiency $\texttt{eff}_\pi$ of a matching policy π. In this paper, we chose to focus on Manhattan for the shared ride market of the island has been well studied in the literature and the city makes publicly accessible taxi data-sets. A similar approach can be applied to other cities. We will conduct a comparative study on different cities in the United States. The goal of this analysis is to quantify the impact of demand patterns, such as pendulum migrations, on the efficiency in a given road network.

Our analysis is based on a single discount value d set for the regional market. In reality, e-hailing platforms set the discount per trip in hope of nudging passengers away from requesting a shared ride on a less efficient routes. The model of this paper relies on the assumption that a passenger's choice of mean of transportation is independent of the trip she requests, with a trip specific shared discount, this assumption does not hold. We will further our research in this direction to provide impactful insight for e-hailing companies.

The rise of the e-hailing platforms sparked the development of stiffer regulations from the cities. The city of New York has been particularly fast to voting new regulation to protect drivers and favor taxis over e-hailing companies. For instance,

it had e-hailing companies limit the number of cars they have on their apps [28] and enforces a minimum pick-up fee for the pick up of each shared passengers [10]. These regulations often have unintended consequences on the industry. We will extend the framework of this paper to provide quantitative insight on the impact of these regulations on the shared ride industry.

Bibliography

[1] Ali Aouad and Omer Saritac. "Dynamic Stochastic Matching Under Limited Time". In: (2019). DOI: 10.2139/ssrn.3497624.

[2] Arash Asadpour, Xuan Wang, and Jiawei Zhang. "Online Resource Allocation with Limited Flexibility". In: *Ssrn* (2016). DOI: 10.2139/ssrn.2794679.

[3] M. L. Balinski. "Notes—On a Selection Problem". In: *Management Science* 17.3 (1970), pp. 230–231. DOI: 10.1287/mnsc.17.3.230. eprint: https://doi.org/10.1287/mnsc.17.3.230. URL: https://doi.org/10.1287/mnsc.17.3.230.

[4] Leo Breiman. "Bagging predictors". In: *Machine Learning* 24.2 (Aug. 1996), pp. 123–140. ISSN: 1573-0565. DOI: 10.1007/BF00058655. URL: https://doi.org/10.1007/BF00058655.

[5] Tom Brijs et al. "Building an Association Rules Framework to Improve Product Assortment Decisions". In: *Data Mining and Knowledge Discovery* 8.1 (Jan. 2004), pp. 7–23. ISSN: 1573-756X. DOI: 10.1023/B:DAMI.0000005256.79013.69. URL: https://doi.org/10.1023/B:DAMI.0000005256.79013.69.

[6] A Catalán and M Fisher. "Assortment allocation to distribution centers to minimize split customer orders". In: *SSRN Electronic Journal* (2012).

[7] Annie Chen. "Large-Scale Optimization in Online-Retail Inventory Management by Annie I-An Chen". PhD thesis. 2012. URL: http://web.mit.edu/sgraves/www/papers/papers/Annie_Chen_MIT_PhD_thesis.pdf.

[8] C. De Boom et al. "Learning Semantic Similarity for Very Short Texts". In: (Nov. 2015), pp. 1229–1234. ISSN: 2375-9259. DOI: 10.1109/ICDMW.2015.86.

[9] Rebecca DeNale and Deanna Weidenhamer. "Quaterly retail e-commerce sales 2nd quarter 2018". In: *US Census Bureau News* (2018).

[10] Department of Taxation and Finance. *Congestion surcharge*. https://www.tax.ny.gov/bus/cs/csidx.htm. 2019.

[11] Levi DeValve et al. "Understanding the Value of Fulfillment Flexibility in an Online Retailing Environment". In: *SSRN Electronic Journal* (2018). DOI: 10.2139/ssrn.3265838.

[12] G. Gallo, M. Grigoriadis, and R. Tarjan. "A Fast Parametric Maximum Flow Algorithm and Applications". In: *SIAM Journal on Computing* 18.1 (1989), pp. 30–55. DOI: 10.1137/0218003. eprint: https://doi.org/10.1137/0218003. URL: https://doi.org/10.1137/0218003.

[13] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. "Algorithms for Association Rule Mining &Mdash; a General Survey and Comparison". In: *SIGKDD Explor. Newsl.* 2.1 (June 2000), pp. 58–64. ISSN: 1931-0145. DOI: 10.1145/360402.360421. URL: http://doi.acm.org/10.1145/360402.360421.

[14] Dorit S Hochbaum. "An Efficient Algorithm for Image Segmentation, Markov Random Fields and Related Problems". In: *Journal of the ACM* 48.4 (2001), pp. 686–701. ISSN: 00045411. DOI: 10.1145/502090.502093.

[15] Dorit S. Hochbaum. "Dynamic evolution of economically preferred facilities". In: *European Journal of Operational Research* 193.3 (2009), pp. 649–659. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2007.07.027. URL: http://www.sciencedirect.com/science/article/pii/S0377221707010284.

[16] Dorit S. Hochbaum. "Polynomial time algorithms for ratio regions and a variant of normalized cut". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.5 (2010), pp. 889–898. ISSN: 01628828. DOI: 10.1109/TPAMI.2009.80.

[17] Dorit S. Hochbaum. "The Pseudoflow Algorithm: A New Algorithm for the Maximum-Flow Problem". In: *Operations Research* 56.4 (2008), pp. 992–1009. DOI: 10.1287/opre.1080.0524. eprint: https://doi.org/10.1287/opre.1080.0524. URL: https://doi.org/10.1287/opre.1080.0524.

[18] Dorit S Hochbaum and U C Berkeley Ca. "an Efficient and Effective Image Segmentation Interactive Tool". In: *Analysis* i (2001), pp. 459–461.

[19] William C Jordan and Stephen C Graves. "Principles on the benefits of manufacturing process flexibility". In: *Management Science* 41.4 (1995), pp. 577–594.

[20] René De Koster, Tho Le-Duc, and Kees Jan Roodbergen. "Design and control of warehouse order picking: A literature review". In: *European Journal of Operational Research* 182.2 (2007), pp. 481–501. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2006.07.009. URL: http://www.sciencedirect.com/science/article/pii/S0377221706006473.

[21] Dominik Kress, Nils Boysen, and Erwin Pesch. "Which items should be stored together? A basic partition problem to assign storage space in group-based storage systems". In: *IISE Transactions* 49.1 (2017), pp. 13–30. DOI: 10.1080/0740817X.2016.1213469. eprint: https://doi.org/10.1080/0740817X.2016.1213469. URL: https://doi.org/10.1080/0740817X.2016.1213469.

[22] MTA. *Turnstile Data*. http://web.mta.info/developers/data/nyct/turnstile.

[23] OpenStreetMap contributors. *Planet dump retrieved from https://planet.osm.org*. https://www.openstreetmap.org. 2017.

[24] Andrea Paraboschi, Paolo Santi, and Carlo Ratti. "Modeling Urban-level Impact of a Shared Taxi Market". In: *CUPUM*. 2015.

[25] Paolo Santi et al. "Quantifying the benefits of vehicle pooling with shareability networks". In: *PNAS* 111.37 (2014). DOI: 10.1073/pnas.1403657111.

[26] Ramakrishnan Srikant and Rakesh Agrawal. "Mining Quantitative Association Rules in Large Relational Tables". In: *SIGMOD Rec.* 25.2 (June 1996), pp. 1–12. ISSN: 0163-5808. DOI: 10.1145/235968.233311. URL: http://doi.acm.org/10.1145/235968.233311.

[27] R Tachet et al. "Scaling Law of Urban Ride Sharing". In: March (2017), pp. 1–6. DOI: 10.1038/srep42868.

[28] TLC. *Utilization Rate*. https://rules.cityofnewyork.us/tags/utilization-rate. 2018.

[29] Ups. "UPS Pulse of the Online Shopper Study". In: (2017).

[30] Daixin Wang, Peng Cui, and Wenwu Zhu. "Structural Deep Network Embedding". In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 1225–1234. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939753. URL: http://doi.acm.org/10.1145/2939672.2939753.

[31] Julie Ward et al. "HP Transforms Product Portfolio Management with Operations Research". In: *INFORMS Journal on Applied Analytics* 40.1 (2010), pp. 17–32. DOI: 10.1287/inte.1090.0476. eprint: https://pubsonline.informs.org/doi/pdf/10.1287/inte.1090.0476. URL: https://pubsonline.informs.org/doi/abs/10.1287/inte.1090.0476.

[32] Zhen Xu et al. "Online Demand Fulfillment Under Limited Flexibility". In: *SSRN Electronic Journal* 1995 (2018). DOI: 10.2139/ssrn.3201734.

[33] Kenan Zhang and Yu Marco Nie. *To pool or not to pool : Equilibrium , pricing and regulation*. 2019. URL: https://ssrn.com/abstract=3497808.

Appendix A

Data-Driven Inventory Placement for E-Tailer

A.1 Complexity of the FDC inventory placement problem

The FDC assortment problem is NP hard by reduction from the k densest subgraph problem. For $G = (V, E)$ an undirected graph, solving the k densest subgraph problem corresponds to finding a subset $U \in V$ of vertices of size k so that it maximizes the average degree $d^* = 2|E(U)|/k$ where $|E(U)|$ denotes the number of edges in the subgraph induced by U. For each vertex $v_i \in V$, let product i be in the set of products I. For each edge $(v_i, v_j) \in E$, let order type $o = (i, j)$ be in the set of order type O with $\mu_{D_o} = 1$. Solving the k FDC assortment problem selects the set $\mathscr{S}$ of k products that fulfills the maximum number of orders, which is equivalent to the set of nodes that maximizes $|E(U)|$. Therefore, if *MIP* can be solved in polynomial time, so can the k-densest subgraph.

A.2 Industry standard: ranking algorithm

This industry standard algorithm assigns a score to each SKU i equal to $\sum_{o \in O, i \in o} \mu_{D_o}/|o|$. The algorithm outputs the k SKUs with the highest score. This algorithm in practice shows good performance.

There is no performance bound. Let a_j, b_j, $c_1^{a_j}$, $c_2^{a_j}$, $c_3^{a_j}$ be $5k$ unique products for $j \in \{1, \ldots, k\}$. By setting $\mu_{D_o} = 1$ only for $o \in \{\{b_j\},\ \{a_j, c_1^{a_j}\},\ \{a_j, c_2^{a_j}\},\ \{a_j, c_3^{a_j}\}\}$, this algorithm selects the $\{a_j | j \in 1, \ldots, k\}$ missing all the orders where the optimal

solution $\{b_j | j \in 1, \ldots, k\}$ fulfills k orders.

Appendix B

Matching policies for Shared Ride simulations

Since we do not have access to any e-hailing platform proprietary matching policy, we will evaluate the efficiency and the detour of three matching polices. The first matching policy, called *Clairvoyant offline*, provides an upper bound on the efficiency an e-hailing platform can realize. The two others are more realistic policies, both weaker in knowledge and more restrictive than the clairvoyant offline policy.

Clairvoyant offline solves the matching problem with perfect knowledge of all requests. It provides an upper-bound on the efficiency any e-hailing platform can achieve subject to the same constraints.

Greedy batching is a simple myopic matching policy that solves to optimality a weighted matching problem every few seconds. This matching policy will provide a lower bound on the e-hailing platform profit

Vertex Additive is a state of the art matching policy for dynamic stochastic matching under limited time. This algorithm can easily be implemented by an e-hailing platform, it will therefore provide a realistic estimate of the e-hailing platform profit.

B.1 Passenger waiting time

Under the assumption of perfect supply, it is always feasible to immediately pickup a passenger. We also assume that every vehicle moves at a constant speed in the road network $\mathcal{N}$

E-hailing platform have their passenger wait before a driver is dispatched to batch demand and make better matching decisions. To keep waiting time reasonable, the waiting time of a shared rider is upper bounded by w.

During the waiting time, a trip τ_i can be matched to τ_j only if both passengers can be picked up before the end of their waiting windows. We focus on the case where i is picked up first, the other cases are defined analogously. For passenger i being the first pick up, the match is feasible only if i can reach O_j before j leaves its waiting window. The earliest time i can start moving toward O_j is at request time t_i. Thus, the match is feasible only if $t_i + l(O_i, O_j) \leq t_j + w$

If a passenger is still unmatched at the end of her waiting time, she has to be picked up by a driver and start moving in some direction with respect to the matching policy π. For instance, since clairvoyant offline policy has perfect knowledge of all requests, it can route a passenger toward the pickup location of some passenger that has yet to request.

Once the waiting time has elapsed, the passenger is picked up by a driver and any additional wait time or travelled distance is accounted for in the detour experienced by the passenger and the total cost of the route.

B.2 Clairvoyant offline policy

In this Subsection, π will refer to the *Clairvoyant offline* policy.

The offline matching policy is a clairvoyant policy, i.e., it has perfect knowledge of all trip requests. Consequently, this policy can route a passenger in any direction and this at any moment during the passenger's waiting time.

For simplification of notation, we focus on the match $\{\tau_i, \tau_j\}$, where passenger i is picked up first and the driver has to pass by O_i, O_j, D_i and D_j in that order. All other cases (O_i, O_j, D_j, D_i), (O_j, O_i, D_i, D_j) and (O_j, O_i, D_j, D_i) are defined analogously.

Once the waiting time has elapsed, the passenger is picked up by a driver, and can start moving in any direction. In our example, the latest time at which passenger i can reach O_j is $t_i + w + l(O_i, O_j)$, hence the minimum additional waiting time for i in $\{\tau_i, \tau_j\}$ is:$\big(t_j - [t_i + w + l(O_i, O_j)]\big)^+$ where $(\cdot)^+ = \max(0; \cdot)$. This additional waiting time is accounted for in the detour and the total distance travelled.

We can derive the total distance travelled by passenger i when matched with j under the clairvoyant offline policy π. We called this distance $l^i_\pi(\{\tau_i, \tau_j\})$. In our example, passenger i is being picked up first and start moving toward O_j such that her additional waiting time is minimized. Consequently, she will travel $l(O_i, O_j) + l(O_j, D_i)$ to reach her destination and will wait $\big(t_j - [t_i + w + l(O_i, O_j)]\big)^+$ at O_j:

$$l^i_\pi(\{\tau_i, \tau_j\}) = l(O_i, O_j) + l(O_j, D_i) + \big(t_j - [t_i + w + l(O_i, O_j)]\big)^+$$

Passenger i experiences a longer trip due to the match $\{\tau_i, \tau_j\}$. Her trip is delayed by: $l(O_i, O_j) + l(O_j, D_i) + \big(t_j - [t_i + w + l(O_i, O_j)]\big)^+ - l(O_i, D_j)$. This quantity is captured by the ratio of additional travel distance for i that is denoted $\delta^i_\pi(\{\tau_i, \tau_j\})$:

$$\delta^i_\pi(\{\tau_i, \tau_j\}) = \frac{l(O_i, O_j) + l(O_j, D_i) + \big(t_j - [t_i + w + l(O_i, O_j)]\big)^+ - l(O_i, D_j)}{l(O_i, D_j)}$$

Similarly, passenger j stops at D_i, hence she travels $l(O_j, D_i) + l(D_i, D_j)$. Since the match is feasible only if the second pickup j can be picked up during her waiting time, j does not experience any additional waiting time:

$$l^j_\pi(\{\tau_i, \tau_j\}) = l(O_j, D_i) + l(D_i, D_j)$$

Passenger j experiences a longer trip due to the match $\{\tau_i, \tau_j\}$. Her trip is delayed by: $l(O_j, D_i) + l(D_i, D_j) - l(O_j, D_j)$. The ratio of additional travel distance is computed as follows:

$$\delta^j_\pi(\{\tau_i, \tau_j\}) = \frac{l(O_j, D_i) + l(D_i, D_j) - l(O_j, D_j)}{l(O_j, D_j)}$$

The total distance and additional wait time that has to be paid to the driver has been called $l_\pi(\{\tau_i, \tau_j\})$. In our example, it is computed as follows:

$$l_\pi(\{\tau_i, \tau_j\}) = l(O_i, O_j) + l(O_j, D_i) + l(D_i, D_j) + \big(t_j - [t_i + w + l(O_i, O_j)]\big)^+$$

To make the mode more attractive, e-hailing platforms have maximum detour value δ_{max}, the offline policy subject to such detour constraint is denoted $\pi(\cdot,\ \delta_{max})$. Under this matching policy, the match $\{\tau_i, \tau_j\}$ is feasible only if

$$\delta^i_\pi(\{\tau_i, \tau_j\}) \leq \delta_{max} \text{ and } \delta^j_\pi(\{\tau_i, \tau_j\}) \leq \delta_{max}$$

Only the feasible ordering for $\pi(\cdot,\ \delta_{max})$ that minimizes the total travelled distance is relevant. We note that minimizing the total travelled distance is equivalent to maximizing the cost savings $l_{\pi(\cdot,\ \delta_{max})}(\{\tau_i, \tau_j\}) - l(\tau_i) - l(\tau_j)$

The Clairvoyant offline policy minimizes the total distance travelled for a set of shared trips $\mathscr{T}_s$ by finding a maximum weighted matching in a graph where the nodes correspond to the trip and the edge weights are the cost saving realized by the match of the two corresponding end-nodes.

B.3 Restriction on feasibility of matches for Greedy batching and Vertex additive

In this Subsection, π refer to a Greedy batching or a Vertex additive policy interchangeably.

Greedy batching and Vertex additive are not clairvoyant policies, hence matches have to be made dynamically with only partial information of the set of trip requests. Both policies only have knowledge of past requests and the available waiting time of passengers.

For simplification of notation, we focus on the match $\{\tau_i, \tau_j\}$, where passenger i is picked up first and the driver has to pass by O_i, O_j, D_i and D_j in that order. All other cases (O_i, O_j, D_j, D_i), (O_j, O_i, D_i, D_j) and (O_j, O_i, D_j, D_i) are defined analogously.

Once the waiting time of a passenger has elapsed, should she remain unmatched, she is picked up by a driver and starts moving from her pickup location to her drop-off location along the shortest path $\mathbf{p}$ in $\mathscr{N}$. By a slight abuse of notation, we denote $\mathbf{p}_i$ the shortest path associated with trip τ_i

In our example, passenger i starts moving from the pickup location O_i toward the drop-off location D_i along the shortest path $\mathbf{p}_i$ in $\mathscr{N}$ at time $t_i + w$. The earliest time passenger i can be routed toward passenger j's pickup location is when passenger j requests at t_j. At that time, passenger i has travelled $(t_j - t_i - w)$.

We denote by $U_i(t_j)$ the first node passenger i can reach along the shortest path after passenger j has requested, and $\kappa_i(t_j)$ the distance left to cover to reach this node. Figure B.1 provides an example of such situation, it represents a snapshot at t_j with passenger i on route and passenger j at its pickup location. The match $\{\tau_i, \tau_j\}$ is feasible only if passenger j is still in her waiting window when passenger i reaches O_i: $\kappa_i(t_j) + l(U_i(t_j), O_j) \leq t_j + w$. $U_i(t)$ and $\kappa_i(t)$ are computed as follows:

$$U_i(t) = \underset{u \in \mathbf{p}_i}{\arg\min} \big\{ l(O_i, u) | l(O_i, u) \geq (t - t_i - w)^+ \big\}$$

$$\kappa_i(t) = t_i + w + l\big(O_i, U_i(t)\big) - t$$

We can now derive the total distance travelled by passenger i when matched with passenger j. This distance has been called $l^i_\pi(\{\tau_i, \tau_j\})$. In our example, when passenger j requests at time t_j, passenger i can be routed toward D_j after traveling from O_i to $U_i(t_j)$.

$$l^i_\pi(\{\tau_i, \tau_j\}) = l(O_i, U_i(t_j)) + l(U_i(t_j), O_j) + l(O_j, D_i)$$

Passenger i experiences a longer trip due to the match $\{\tau_i, \tau_j\}$. Her trip is delayed by $l(O_i, U_i(t_j)) + l(U_i(t_j), O_j) + l(O_j, D_i) - l(O_i, D_i)$. This quantity is captured by the ratio of additional travel distance for i denoted $\delta^i_\pi(\{\tau_i, \tau_j\})$:

$$\delta^i_\pi(\{\tau_i, \tau_j\}) = \frac{l(O_i, U_i(t_j)) + l(U_i(t_j), O_j) + l(O_j, D_i) - l(O_i, D_i)}{l(O_i, D_i)}$$

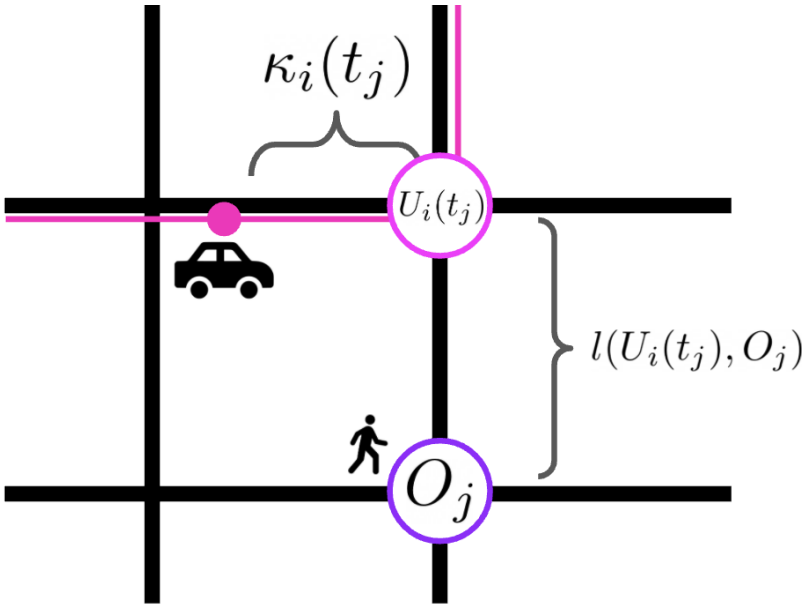


Figure B.1: Example of a passenger j requesting while passenger i is on route

Similarly, passenger j stops at D_i, hence she travels $l(O_j, D_i) + l(D_i, D_j)$. Since j is the second pickup, there is no additional detour:

$$l^j_\pi(\{\tau_i, \tau_j\}) = l(O_j, D_i) + l(D_i, D_j)$$

Passenger j experiences a longer trip due to the match $\{\tau_i, \tau_j\}$. The trip is delayed by $l^j_\pi(\{\tau_i, \tau_j\}) = l(O_j, D_i) + l(D_i, D_j) - l(O_j, D_j)$ and the ratio of additional travel distance is computed as follows:

$$\delta^j_\pi = \frac{l^j_\pi(\{\tau_i, \tau_j\}) = l(O_j, D_i) + l(D_i, D_j) - l(O_j, D_j)}{l(O_j, D_j)}$$

We have called $l_\pi(\{\tau_i, \tau_j\})$ the total distance travelled by the driver. In our example, it is computed as follows:

$$l_\pi(\{\tau_i, \tau_j\}) = l(O_i, U_i(t_j)) + l(U_i(t_j), O_j) + l(O_j, D_i) + l(D_i, D_j)$$

For a maximum detour value δ_{max}, the matching policy subject to such detour constraint is denoted $\pi(\cdot,\ \delta_{max})$. Under this matching policy, the match $\{\tau_i, \tau_j\}$ is feasible only if

$$\delta^i_\pi(\{\tau_i, \tau_j\}) \leq \delta_{max} \text{ and } \delta^j_\pi(\{\tau_i, \tau_j\}) \leq \delta_{max}$$

Added to these constraints, the Vertex Additive policy has an additional constraint per match on the cost savings. The trips are clustered in trip types and a minimum cost savings parameter is learnt with respect to each trip type's arrival rate. The detail of this computation can be found in [1].

Greedy batching and Vertex additive policies are both heuristics to minimize the total operating costs. These policy batch demand for a short period of time and then find a maximum weighted matching on a graph where the nodes correspond to past unmatched trip requests and the edges weight are the cost savings realized by the match of the two corresponding end-nodes.