ONLINE LEARNING AND MATCHING FOR RESOURCE ALLOCATION PROBLEMS*

ANDREA BOSKOVIC†, QINYI CHEN‡, DOMINIK KUFEL§, AND ZIJIE ZHOU¶

Abstract. In order for an e-commerce platform to maximize its revenue, it must recommend customers items they are most likely to purchase. However, the company often has business constraints on these items, such as the number of each item in stock. In this work, our goal is to recommend items to users as they arrive on a webpage sequentially, in an online manner, in order to maximize reward for a company, but also satisfy budget constraints. We first approach the simpler online problem in which the customers arrive as a stationary Poisson process, and present an integrated algorithm that performs online optimization and online learning together. We then make the model more complicated but more realistic, treating the arrival processes as non-stationary Poisson processes. To deal with heterogeneous customer arrivals, we propose a time segmentation algorithm that converts a non-stationary problem into a series of stationary problems. Experiments conducted on large-scale synthetic data demonstrate the effectiveness and efficiency of our proposed approaches on solving constrained resource allocation problems.

Key words. online algorithms, resource allocation, traffic shaping, reinforcement learning, online convex optimization, non-stationary arrivals

AMS subject classifications. 90B05, 90B50, 90B60, 90C05

1. Introduction. Resource allocation has been considered an important task by many e-commerce platforms, and it can essentially be formulated as a generalized online matching problem. In an electronic marketplace, products are placed for sale on a webpage as customers arrive sequentially, viewing the products and making purchase decisions. As each customer arrives, the platform needs to display corresponding items that the customer is likely to purchase. However, the tendency of each customer to purchase a certain product is unknown, and the revenue generated by the sale of different products varies. In a given session, we assume that customers arrive onto the webpage randomly over time. The task at hand is to find a way to match each customer to an item such that this matching maximizes the reward (i.e., the potential revenue generated through the sale of items) with respect to certain constraints, such as the stock of each item.

Resource allocation problems can be approached in either an offline or an online manner. The offline problem assumes that the sequence of customer arrivals is known in advance, while in the online problem, we consider customers arriving onto the webpage as following an unknown stochastic process. Oftentimes, customers arriving in an online manner are modeled as a stationary Poisson process, the rate of which is unknown beforehand. The offline algorithm optimizes multiple functions simultaneously [3], whereas the online problem optimizes different sequences of functions at each time. Although the offline problem is a less realistic problem, the optimal solution to

*Submitted to the editors November 17, 2019. Completed under the guidance of Anna Ma, Department of Mathematics, University of California, Irvine (anna.ma@uci.edu) and Xinshang Wang, DAMO Academy, Alibaba US (xinshang.w@alibaba-inc.com).

†Department of Statistics, Amherst College, Amherst, MA 01002 (aboskovic21@amherst.edu).

‡Department of Mathematics, University of California, Los Angeles, Los Angeles, CA 90095 (qinyichen@ucla.edu).

§Department of Physics, University College London, Gower St, Bloomsbury, London WC1E 6BT, United Kingdom (dominic.kufel@gmail.com).

¶Department of Mathematics, Purdue University, 610 Purdue Mall, West Lafayette, IN 47907 (zhou759@purdue.edu).

the offline problem is necessary for the online setting, specifically in the evaluation of regret, a measure of how well the online algorithm works in comparison to the offline algorithm and its optimal solution. Current work on the online problem [1, 2, 6, 7] mainly focuses on the theoretical aspects of the online problem and attempts to minimize the regret. Moreover, resource allocation problems are closely related to ad allocation, which is also studied in the context of online matching problems. Some notable examples include *DisplayAds* [10] and *AdWords* [9, 13].

In this paper, we propose several online algorithms for allocating products to users, which extend and improve previous work. We first approach the online stationary problem, in which the sequence of customer arrivals is unknown, but the arrival rates of customers are constant over time, by introducing an integrated algorithm that performs online learning and matching together. We then proceed to the non-stationary case, in which customer arrival rates vary over time, and propose another time segmentation algorithm that tackles the customer heterogeneity. We theoretically verify the convergence of average regret in our algorithms, and experimentally demonstrate their efficacy in providing near-optimal product recommendations. It should be noted that although we mainly consider an e-commerce problem set-up, resource allocation also have many other applications. One such application is, for example, in healthcare. In [14], it is suggested that healthcare interventions such as reducing infection spread or slowing disease progression rate may modify the optimal strategies for constrained resource allocation. This connects to the exploration-exploitation trade-off discussed in Subsection 3.2 of this paper.

The rest of the paper is organized as follows. Section 2 discusses the background of the offline and online resource allocation problems, and some existing approaches that we rely upon. Section 3 introduces an integrated algorithm that tackles the online problem with stationary customer arrivals. Section 4 extends the problem to consider heterogeneous customer arrivals, and proposes another algorithm that approximates a non-stationary problem into a series of stationary problems. Section 5 demonstrates experimentally the effectiveness of our proposed algorithms. Section 6 and Section 7 interpret the results of our work and propose future directions.

2. Background. We first review some existing approaches for the offline and online matching problems that we build upon to design our online algorithms as well as outline the framework of each approach.

2.1. Offline Problem. In the offline matching problem, we assume that the distribution of customer arrivals and the preference of customers are both known. Therefore, we can simply optimize the potential revenue by solving an optimization problem, represented by the first term in (2.1). We add a regularization term, the second term in (2.1), where μ accounts for regularization [17], ensuring that our linear program is strongly convex and therefore has only one optimal solution. Adding this regularization term is a standard way to reduce a linear programming problem into a convex optimization problem [4]. In this problem, j indexes the customers, where the total number of customers is m, and i indexes the items, where the total number of items is n. Further, r_i refers to the reward, or revenue, for the company when a particular customer purchases item i, P_{ij} is the customer preference matrix, which contains the probability of customer j purchasing item i given they were offered item i, and $\bar{P}_j$ is the maximum value of P_{ij} for each customer, i.e., $\bar{P}_j = \max_i P_{ij}$. Additionally, x_{ij} refers to the probability that customer j is recommended item i, and b_i is the budget of item i.

We formulate the following optimization problem:

$$\max_{\substack{x_{ij} \\ i\in[n] \\ j\in[m]}} \sum_{i=1}^{n}\sum_{j=1}^{m} r_i P_{ij} x_{ij} - \mu \sum_{j=1}^{m} \bar{P}_j \sum_{i=1}^{n} x_{ij} \log x_{ij}, \tag{2.1}$$

$$\text{s.t.} \sum_{j=1}^{m} P_{ij} x_{ij} \le b_i, \forall i \in [n];$$
$$\sum_{i=1}^{n} x_{ij} = 1, \forall j \in [m];$$
$$x_{ij} \ge 0, \forall i \in [n], j \in [m].$$

The objective function in (2.1) represents maximizing reward, or revenue, for the company. Specifically, the first term in the objective function represents reward, followed by a regularization term that ensures a unique optimal solution. The first constraint is a budget constraint that guarantees we do not recommend an item more of that item than we have in stock. The second constraint guarantees that each customer gets recommended exactly one item, while the last constraint ensures that the probability that a customer gets recommended a certain item is non-negative.

In order to solve our maximization problem, we convert our primal objective function into its dual formulation, and then proceed to minimize the dual function. Note that to ensure that strong duality holds, it suffices to show that our convex optimization problem satisfies the Slater's condition. Let x^* be the optimal primal solution, and additionally assume that the budget $b_i \ge \frac{m}{n}$. Under this assumption, we have that $x^*_{ij} = \frac{1}{n}$ is an interior point. For all $j \in [m]$, we have that $\sum_{i=1}^{n} x^*_{ij} = 1$, and for all $i \in [n]$, $\sum_{j=1}^{m} P_{ij} x^*_{ij} \le b_i$. Therefore, strong duality holds, i.e. solving the dual minimization problem is equivalent to solving the primal maximization problem.

We proceed to derive our dual function, where Λ denotes the dual variable. By Lagrangian duality, the primal form in (2.1) is equivalent to the following:

$$\min_{\Lambda_i} \max_{x_{ij}} \left(\sum_{i=1}^{n}\sum_{j=1}^{m} r_i P_{ij} x_{ij} - \mu \sum_{j=1}^{m} \bar{P}_j \sum_{i=1}^{n} x_{ij} \ln(x_{ij}) + \sum_{i=1}^{n} \Lambda_i \left(b_i - \sum_{j=1}^{m} x_{ij} P_{ij} \right) \right). \tag{2.2}$$

Under the Karush-Kuhn-Tucker (KKT) conditions, we can find the global maximum of the above function by calculating its gradient with respect to x_{ij}, leaving us with

$$x_{ij}(\Lambda_i) = e^{\frac{(r_i - \Lambda_i)P_{ij}}{\mu \bar{P}_j} - 1}. \tag{2.3}$$

To fulfill the equality constraint in (2.1), we have that $e^{-1} \sum_{i=1}^{n} \exp \frac{(r_i - \Lambda_i)P_{ij}}{\bar{P}_j \mu} = 1$, Additionally, let

$$Z_j = \sum_{i=1}^{n} \exp \frac{(r_i - \Lambda_i)P_{ij}}{\bar{P}_j \mu}, \tag{2.4}$$

and this implies that

$$x_{ij}(\Lambda_i) = \frac{1}{Z_j} \exp \frac{(r_i - \Lambda_i)P_{ij}}{\bar{P}_j \mu}. \tag{2.5}$$

If we now substitute x_{ij} into (2.2), we have that

$$f(\Lambda) := \mu \sum_{j=1}^{m} \bar{P}_j \log Z_j + \langle \Lambda, b \rangle, \tag{2.6}$$

where $Z_j = \sum_{i=1}^{n} \exp\left(\frac{(r_i - \Lambda_i)P_{ij}}{\bar{P}_j \mu}\right)$ is known as normalization factor. To summarize, μ accounts for regularization, as described before (2.1). Given optimal dual variable Λ^*, the optimal primal solution x^* is then

$$x^*_{ij} = \frac{1}{Z_j} \exp \frac{(r_i - \Lambda^*_i)P_{ij}}{\bar{P}_j \mu}. \tag{2.7}$$

The primal formulation (2.1) is thus converted to its dual form (2.6) by means of Lagrangian duality. This is a well-studied topic in optimization, and more details can be found in [11]. To obtain the optimal solution to the offline matching problem, various first-order optimization algorithms can be applied, such as gradient descent (GD) and stochastic gradient descent (SGD). In this work, the objective function is minimized via GD with lingering radius (GD^{lin}), a less computationally expensive, state-of-the-art method [4] well-suited for solving resource allocation problems. Due to our problem formulation, the gradients of our objective functions would not change significantly in a sufficiently small gradient descent step, and therefore GD^{lin} in general leads to better performance.

2.2. Online Stationary Problem. The goal of an online matching algorithm is to recommend products to customers as they arrive sequentially onto the webpage in a way that not only maximizes reward, but also satisfies budget constraints. One difficulty of the online problem is that as each customer arrives, their preference for any particular item is unknown and must be learned in real time. To obtain a prediction of the customer preferences in advance, e-commerce platforms often divide the customers into different types, according to their demographics or other information. In the most simplified online problems, each type of customer is assumed to arrive as a stationary Poisson process. To learn their preferences P_{ij}, which correspond to the likelihood that a customer from type j buys item i, we apply reinforcement learning techniques. By utilizing knowledge about the purchases of previous customers, we make product allocation decisions for future arriving customers.

One commonly used technique to take the best possible action to maximize reward, or to determine the best product to recommend to each customer type, is the Upper Confidence Bound (UCB) algorithm [5]. The UCB algorithm is considered ideal for our purposes mainly because it is not greedy, i.e., it does not always recommend an item to a specific customer type if that item maximizes reward at a particular time. The algorithm exemplifies the principle of optimism in face of uncertainty, recommending items to each customer type until it exceeds some upper bound of certainty of the expected reward of that item's recommendation. This property allows us to obtain an accurate estimate of P_{ij} fairly early on, thus enabling us to achieve a more accurate solution to the optimization problem. This approach is discussed in more detail in Section 3.

In addition to using the UCB algorithm to recommend products to users, we use online gradient descent, an online convex optimization method, to compute the gradient of the objective function of each arriving customer, which is then used to update the value of our dual variable Λ [18]. The online convex optimization component of the algorithm is crucial in measuring the performance of our online integrated

algorithm. We aim to compare our online solution to the optimal solution from the offline problem, denoted by $f_t(\Lambda^*, P^*)$. In particular, Λ^* refers to the optimal dual variable and P^* refers to the ground truth preference matrix. Specifically, we seek to minimize regret, which is defined as follows:

$$\min_{\Lambda_t, t\in[T]} \text{regret}_T = \sum_{t=1}^{T} \min(f_t(\Lambda_t, P^{(t)})) - \sum_{t=1}^{T} f_t(\Lambda^*, P^*). \tag{2.8}$$

The regret function essentially compares the online problem for each arriving customer t to the optimal solution to the offline problem, where we know the sequence of functions $\{f_1, f_2, ..., f_T\}$ in advance [15]. In other words, regret acts as a metric that uses the offline problem as a benchmark for the online problem. The goal in solving the online problem is to minimize this regret function, thus minimizing the loss incurred due to error in optimization.

2.3. Online Non-stationary Problem. In the online non-stationary problem, we consider a more realistic case: different types of customers arrive according to non-stationary Poisson processes, in which their arrival rates are functions of time. As in our integrated algorithm, we consider the regret of the online non-stationary algorithm, defined in (2.8), and we again aim to minimize this regret function. Although minimal literature exists on problems with non-stationary stochastic customer arrivals, [16] discusses a non-stationary stochastic demand problem.

3. Online Integrated Algorithm. We now consider customer arrivals onto a webpage in an online, or sequential, manner. Additionally, we assume no previous knowledge of customer preferences P_{ij}, and learn this value as customers arrive. The customers are assumed to arrive following a stationary Poisson process, where the Poisson arrival rates are known. In this section, we describe the formulation of the online stationary problem, and introduce an online integrated algorithm that combines the Upper Confidence Bound (UCB) algorithm, which learns customer preferences, with Online Gradient Descent (online GD), which tackles the optimization component of the problem. By performing online learning and optimization together, the integrated algorithm thus allows us to recommend the optimal product to each customer, and study their purchasing behaviors at the same time.

3.1. Mathematical Formulation. In the online stationary problem, we assume that a total of T customers arrive over the entire time period. The customers arrive in a sequential manner, and when the t^{th} customer arrives, the only information we have is the information about the previous customers. Our objective is to maximize the total expected reward for all customers by maximizing reward for any given t^{th} customer, where $t \in [T]$.

However, it is oftentimes too computationally expensive to learn the purchasing behavior of every single customer and minimize the dual variable for each of them separately. We therefore group the customers into different types based on their demographics—as e-commerce platforms tend to do in practice—since customers from the same background tend to display similar shopping behaviors. We assume that there are m types of customers, and the customer preferences in each type are i.i.d. We let the preference matrix P_{ij} represent the probability that any customer of type j buys item i, instead of the preference of a single customer. Additionally, we assume the customers of type j arrive as a stationary Poisson process of rate λ_j. Therefore by the superposition property of Poisson processes, we know that the probability that the customer arrival is of type j is $\frac{\lambda_j}{\sum_{s=1}^{m} \lambda_s}$.

The primal objective for the t^{th} customer now becomes:

$$\max_{x_{ij}} \sum_{i=1}^{n}\sum_{j=1}^{m} r_i P_{ij} x_{ij} \frac{\lambda_j}{\sum_{s=1}^{m}\lambda_s}, \tag{3.1}$$

$$\begin{aligned} \text{s.t.} \sum_{j=1}^{m} \frac{\lambda_j}{\sum_{s=1}^{m}\lambda_s} P_{ij} x_{ij} &\le \frac{b_i}{T}, \quad \forall i \in [n]; \\ \sum_{i=1}^{n} x_{ij} &= 1, \quad \forall j \in [m]; \\ x_{ij} &\ge 0, \quad \forall i \in [n],\ j \in [m]. \end{aligned}$$

Note that (3.1) now reflects the expected revenue we would obtain from the t^{th} customer arrival.

The above primal problem with the regularization term leads to the following:

$$\begin{aligned} \min_{\Lambda_i}\max_{x_{ij}} \Bigg(\sum_{i=1}^{n}\sum_{j=1}^{m} r_i P_{ij} x_{ij} \frac{\lambda_j}{\sum_{s=1}^{m}\lambda_s} - \mu \sum_{j=1}^{m} \frac{\lambda_j}{\sum_{s=1}^{m}\lambda_s} \bar{P}_j \sum_{i=1}^{n} x_{ij}\log(x_{ij}) \\ + \sum_{i=1}^{n} \Lambda_i \left(\frac{b_i}{T} - \sum_{j=1}^{m} \frac{\lambda_j}{\sum_{s=1}^{m}\lambda_s} P_{ij} x_{ij} \right) \Bigg), \end{aligned} \tag{3.2}$$

Similarly to the offline problem, assuming the Karush-Kuhn-Tucker (KKT) conditions hold, we can globally maximize the Lagrangian (3.2) by calculating the gradient with respect to x_{ij}, which implies:

$$x_{ij}(\Lambda_i) = \frac{1}{Z_j} e^{\frac{(r_i - \Lambda_i)P_{ij}}{\mu \bar{P}_j}}, \tag{3.3}$$

where $Z_j = \sum_{i=1}^{n} e^{\frac{(r_i-\Lambda_i)P_{ij}}{\mu\bar{P}_j}}$ ensures the fulfillment of the equality constraint in (3.1). Plugging in the form of x_{ij} from (3.3) to the Lagrangian (3.2) leads to the following dual minimization problem:

$$\min_{\Lambda} f_t(\Lambda, P) = \min_{\Lambda} \left(\mu \sum_{j=1}^{m} \frac{\lambda_j}{\sum_{s=1}^{m}\lambda_s} \bar{P}_j \log(Z_j) + \frac{1}{T}\langle \Lambda, b\rangle \right). \tag{3.4}$$

Note that while f_t denotes the objective function related to the t^{th} customer, f_t does not depend on t. Similar to the offline case, after obtaining the optimal dual variable Λ^*, we can obtain x^* by applying (3.3).

In order to evaluate the performance of our online algorithm, we define the regret function [18], which compares our online dual objective against the optimal dual objective obtained in the corresponding offline problem:

Definition 3.1. *Given an online algorithm and online minimization problem* (3.4)*, the regret of the algorithm at time T is:*

$$\text{regret}_T = \sum_{t=1}^{T} f_t(\Lambda_t, P^{(t)}) - \sum_{t=1}^{T} f_t(\Lambda^*, P^*),$$

where Λ^* denotes the optimal dual variable in the offline problem and P^* is the underlying ground truth customer preference matrix. At each iteration, we obtain the preference matrix $P^{(t)}$ and dual variable Λ_t. Note that here the $\sum_{t=1}^{T} f_t(\Lambda^*, P^*)$ is simply the optimal offline dual when the ground truth preference matrix is known. Additionally, we define the average regret to be $\frac{\text{regret}_T}{T}$.

Our goal of solving the online problem is to minimize this regret function, thus minimizing the loss incurred due to error in optimization. Note that

$$\min_{\Lambda_t, t\in[T]} \text{regret}_T = \sum_{t=1}^{T} \min(f_t(\Lambda_t, P^{(t)})) - \sum_{t=1}^{T} f_t(\Lambda^*, P^*),$$

which matches (2.8) as minimizing the regret_T does not change the second term because the values in the summation are fixed.

Note that in our problem set-up, customer preference P is initially unknown. Not only do we need to solve the online stationary problem, we also need to gradually learn P and to keep updating it as customers arrive. Therefore, when solving the minimization problem in (3.4), the variable P_{ij} will change as customers continue arriving. In the following sections, we describe an integrated algorithm that allows us to learn P and solve the optimization problem simultaneously.

3.2. Upper Confidence Bound (UCB) Algorithm. In Algorithm 3.1, we introduce the UCB algorithm [12] used as part of our integrated algorithm. The UCB algorithm, typically used in the context of multi-armed bandit problems, is relevant to the online problem proposed here. It efficiently manages the trade-off between *exploration*—learning of the customer preference matrix P, and *exploitation*—optimizing $f_t(\Lambda, P)$ during online matching given the current knowledge of P.

Here, we let $D \in \{0,1\}^{n\times T}$ denote a binary reward matrix, where each entry D_{it} denotes whether or not the t^{th} customer buys item i. By the time of the t^{th} customer arrival, we let $N_i(t)$ denote the number of times item i has been selected and $R_i(t)$ be the amount of rewards we have already collected by assigning item i. The average reward is denoted as $\bar{r}_i(t) = R_i(t)/N_i(t)$. We define our UCB function as follows:

$$\text{UCB}_i(t-1) = \begin{cases} \infty & \text{if } N_i(t-1) = 0; \\ \bar{r}_i(t-1) + \sqrt{\frac{3\log(t)}{2N_i(t-1)}} & \text{otherwise.} \end{cases}$$

Note that the parameters in the definition of the upper confidence bound can in fact be tuned depending on how much importance we place on the exploration component.

Algorithm 3.1 Upper Confidence Bound (UCB) Algorithm

Input: number of customer arrivals T, reward matrix D
Output: item assignments $\{i^{(t)}\}_{t=1,\ldots,T}$
for t = 1, ..., T **do**
  Choose the item to assign: $i^{(t)} = \text{argmax}_i \text{UCB}_i(t-1)$
  Observe reward $D[i^{(t)}, t]$
  $N_i(t) = N_i(t-1) + 1; R_i(t) = R_i(t-1) + D[i^{(t)}, t]$
end for

In a typical setting, Algorithm 3.1 initially favors the exploration component, due to small $N_i(t-1)$, but over time the algorithm would transition to a predominantly

exploitatory phase, and $\bar{r}_i(t-1)$ would not be prone to large fluctuations as it was already estimated with large number of selections $N_i(t)$.

3.3. Online Gradient Descent (Online GD). Online GD [11] is an algorithm similar to offline gradient descent. In the offline problem, since all the data is known at the start of the matching process, we can compute the gradient of the full objective function. However, in the online problem, since customers arrive one by one, our data set grows over time as we learn more about the item preferences of each customer type. Therefore, we can only use the data we have at a particular time to compute gradients. Thus, we only iterate through the data set once, unlike in the offline GD, where we loop through the data many times.

When applying online GD, we start from an initial $\Lambda_0 \in \kappa$, where κ is a convex set. Then we iterate through $t = 1, ..., T$, and at each iteration, we update Λ_t in the following way:

$$y_{t+1} = \Lambda_t - \eta_t \nabla_{\Lambda_t} f_t(\Lambda_t),$$

$$\Lambda_{t+1} = \text{proj}_\kappa(y_{t+1}) = \text{argmin}_{s\in\kappa}\|y_{t+1} - s\|.$$

Here, $\eta_t \in \mathbb{R}$ is the step size, and $\text{proj}_\kappa(y_{t+1})$ is the projection of y_{t+1} onto a convex set κ. Note that if $y_{t+1} \in \kappa$, then evidently $\Lambda_{t+1} = y_{t+1}$.

3.4. Integrated Algorithm. Combining the UCB algorithm and online GD, we create an efficient integrated algorithm that solves the online stationary problem. Our integrated algorithm relies on a learning component that updates customer preference P_{ij}, as well as an optimization component that finds the optimal assignment of items that results in the highest expected reward. We assume that when the t^{th} customer arrives, we will first observe their type j and assign them to an item i using the UCB algorithm. Then, based on whether the customer of type j purchases the item i or not, we update $P^{(t)}$ to reflect a more accurate customer preference matrix: each entry $P_{ij}^{(t)} = R_{ij}^{(t)}/N_{ij}^{(t)}$, where $R_{ij}^{(t)}$ is the total number of times customers of type j purchase item i, and $N_{ij}^{(t)}$ is the total number of times item i gets assigned to customers of type j until time t. We then use this $P^{(t)}$ in (3.4) and apply online GD to get the solution for the dual variable. If the $P^{(t)}$ that we get at each iteration converges, we can halt the UCB algorithm and only run online GD until the dual variables Λ_t also converge. Our algorithm is summarized in Algorithm 3.2.

Note that since $P^{(t)}$ does not necessarily reflect the true preference matrix, the optimization problem that we solve changes at each iteration as we update $P^{(t)}$. In the following section, we theoretically show that as long as the number of customer arrivals are sufficient, $P^{(t)}$ eventually converges to the true P_{ij}.

3.5. Upper Bound of Average Regret. Recall that in the online model, our goal is to minimize the regret, as defined in Definition 3.1. Here, we show that using the integrated algorithm, the average regret converges to zero as the number of customer arrivals approaches infinity.

THEOREM 3.2. *Consider Algorithm 3.2, we have:*

$$\limsup_{T\to\infty} \frac{\text{regret}_T}{T} = 0.$$

Proof. Recall that the optimal offline solution is the minimizer of (2.6), which can be re-written as $\Lambda^* = \text{argmin}_{\Lambda\in\kappa}\sum_{t=1}^{T} f_t(\Lambda, P^*)$. Assuming $\Lambda^* \in \kappa$, when projecting

Algorithm 3.2 Online Integrated Algorithm

Input: Customer arrivals $t = 1, ..., T$, number of customer types m, number of items n, budgets $b \in \mathbb{R}^n$, rewards $r \in \mathbb{R}^n$, initial preference matrix $P^{(0)} \in \mathbb{R}^{m\times n}$, initial dual variable $\Lambda_0 \in \mathbb{R}^n$, maximum number rounds of UCB $R_{\max}$.

Output: item assignments for $t = 1, ..., T$.

for t = 1, ..., T **do**
  Observe the type of this customer: $j = 1, ..., m$.
  if $\left\|P^{(t)} - P^{(t-1)}\right\| > \epsilon$ **and** $t \le R_{\max}$ **then**
    Assign item i with maximum UCB value for customer type j.
  else
    Assign item i with Λ_{t-1}.
  end if
  Update $P_{ij}^{(t)} = R_{ij}^{(t)}/N_{ij}^{(t)}$.
  Define $f_t(\Lambda) = f_t(\Lambda, P^{(t)})$ according to (3.4).
  $\Lambda_t = \text{proj}_\kappa\{\Lambda_{t-1} - \eta_{t-1}\nabla f_t(\Lambda_{t-1})\}$.
end for

y_{t+1} onto κ, we must have $\|\Lambda_{t+1} - \Lambda^*\| = \|\text{proj}_\kappa(y_{t+1}) - \Lambda^*\| \le \|y_{t+1} - \Lambda^*\|$, where $y_{t+1} = \Lambda_t - \eta_t\nabla f_{t+1}(\Lambda_t)$. Define $\bigtriangledown_t = \bigtriangledown_\Lambda f_t(\Lambda_t, P^{(t)})$. We have that

$$\|y_{t+1} - \Lambda^*\|^2 = \|\Lambda_t - \eta_t \bigtriangledown_t - \Lambda^*\|^2 = \|\Lambda_t - \Lambda^*\|^2 + \eta_t^2\| \bigtriangledown_t \|^2 - 2\eta_t\langle\bigtriangledown_t, \Lambda_t - \Lambda^*\rangle.$$

By the convexity of f_t, which is ensured by appropriately tuning the regularization parameter μ (see [17] for details), we get

$$\begin{aligned} f_t(\Lambda_t, P^*) - f_t(\Lambda^*, P^*) &\le \langle\bigtriangledown_t, \Lambda_t - \Lambda^*\rangle \\ &\le \frac{1}{2\eta_t}(\|\Lambda_t - \Lambda^*\|^2 - \|\Lambda_{t+1} - \Lambda^*\|^2) + \frac{\eta_t}{2}\| \bigtriangledown_t \|^2. \end{aligned} \tag{3.5}$$

Let D be the upper bound on diameter of the convex set κ; that is, $\forall x, y \in \kappa, |x-y| \le D$. In addition, as f_t is assumed to be Lipschitz continuous, we let G be such that $\| \bigtriangledown_t \| \le G$ for all $1 \le t \le T$ and for all $\Lambda \in \kappa$, We define $\eta = \eta_t = \frac{D}{a\sqrt{T}}$ for $1 \le t \le T$. If we sum (3.5) over t we obtain

$$\begin{aligned} \sum_{t=1}^{T}(f_t(\Lambda_t, P^*) - f_t(\Lambda^*, P^*)) &\le \frac{1}{2\eta}\|\Lambda_1 - \Lambda^*\|^2 + \frac{\eta}{2}\sum_{t=1}^{T}\| \bigtriangledown_t \|^2 \\ &\le \frac{1}{2\eta}D^2 + \frac{\eta}{2}TG^2 \\ &= GD\sqrt{T}. \end{aligned}$$

Note that the function f_t is different for the online and offline problems because in the online problem, the preference P gets updated at each iteration. We have:

$$f_t(\Lambda_t, P^{(t)}) - f_t(\Lambda^*, P^*) \le |f_t(\Lambda_t, P^{(t)}) - f_t(\Lambda_t, P^*)| + |f_t(\Lambda_t, P^*) - f_t(\Lambda^*, P^*)|. \tag{3.6}$$

If we sum over (3.6) for $1 \le t \le T$, we get the following:

$$\text{regret}_T \le \sum_{t=1}^{T}|f_t(\Lambda_t, P^{(t)}) - f_t(\Lambda_t, P^*)| + GD\sqrt{T}.$$

Here, $|f_t(\Lambda_t, P^{(t)}) - f_t(\Lambda_t, P^*)|$ represents the regret resulting from approximating P^* with $P^{(t)}$. If we apply the UCB algorithm to obtain the approximations $P^{(t)}$ for $t = 1, ..., T$, the total regret after T iterations is $O(T \log T)$ [12]. That is, there exists $C > 0$ such that

$$\sum_{t=1}^{T} |f_t(\Lambda_t, P^{(t)}) - f_t(\Lambda_t, P^*)| \leq C\sqrt{T \log T}.$$

Thus, we have

$$\limsup_{T\to\infty} \frac{\text{regret}_T}{T} \leq \limsup_{T\to\infty} C\sqrt{\frac{\log T}{T}} + GD\sqrt{\frac{1}{T}} = 0$$

Thus, the average regret converges to 0 when $T \to \infty$. ∎

4. Online Non-Stationary Problem. We now extend our previous discussion to consider a more practical setting in which the customers arrive following non-stationary Poisson processes. Removing the assumption of the stationary Poisson processes leads to a more complex formulation of the online convex optimization problem that cannot simply be solved using Algorithm 3.2. In existing literature of the online optimization problem with heterogeneous customer arrivals, if the arrival rates and customer preferences P_{ij} are both known, we can apply methods such as the Large-or-Small Algorithm [16]. However, there is no existing algorithm that can perform online optimization without previous knowledge of arrival rates or customer preferences P_{ij}. The difficulty lies in that if we perform online learning for P_{ij}, the dual variables do not converge. Moreover, if the arrival processes are modeled as non-stationary Poisson processes, then the probability that the next customer arrival comes from type j is almost impossible to calculate.

In this section, we propose a time segmentation algorithm that can approximate this probability and convert the non-stationary arrival problem into a series of stationary problems. We can then solve each stationary problem using the method discussed in Section 3. We give a detailed description of this algorithm in Subsection 4.1, and provide an upper bound of the average regret of this approach in Subsection 4.2.

4.1. Algorithm Description. In the online stationary problem, we obtain the probability that the next customer arrival is of type j by directly invoking the superposition property of Poisson processes. However, in a non-stationary Poisson process, this probability continuously depends on time and thus cannot be simply computed as a constant. To tackle this difficulty that arises, we assume that the Poisson rate function $\lambda_j(t)$ changes slowly inside a sufficiently small time interval. This is a realistic assumption since within a short time period—for example, 10 minutes—it is unlikely that the density of customer arrivals would change drastically. The following discussion thus relies on the assumption that the amount of the change of arrival rate function $\lambda_j(t)$ in a specific time segment I, which is defined by $\max_{t\in I} \lambda_j(t) - \min_{t\in I} \lambda_j(t)$, is bounded by some constant. As we shall see later, to perform an accurate and computationally feasible approximation, we would need both this constant to be sufficiently small, and the length of the time interval I to be reasonably large. We make a further assumption that the rate function $\lambda_j(t)$ is bounded in any given time interval for all $1 \leq j \leq m$. Note that the rate functions may still be discontinuous. An example rate

function that satisfies the above assumptions would be:

$$\lambda(t) = \begin{cases} 0.5\sin(t) + 30 & 0 \le t \le 1 \\ 0.01t + 5 & 1 \le t \le 2 \end{cases}.$$

This represents a realistic setting when a website experiences heavier traffic during the first hour, while the customer arrivals slow down in the second hour; however, arrival rates within one hour do not change drastically.

We proceed to describe the main ideas behind the algorithm for the online non-stationary problem. Recall the Piecewise Constant Approximation Theorem:

THEOREM 4.1. *If f is a continuous function defined on a compact set $D \in \mathbb{R}$, it can be uniformly approximated by a piecewise constant function.*

Theorem 4.1 implies that we can use a piecewise constant function to approximate each arrival rate function. In other words, there exists a series of time segments $\{I\}$ in which

$$\max_{t \in I} \lambda_j(t) - \min_{t \in I} \lambda_j(t) \le \epsilon,$$

for a sufficiently small number ϵ. Inside each time segment, the rate functions can be approximated as constants. In doing so, the non-stationary processes can be approximated with multiple stationary Poisson processes in small time segments. However, it is not computationally feasible to entirely rely on this approach. If we divide the time span into small time segments based on the approximation in Theorem 4.1, the length of time segments has to be extremely small in some cases in order to achieve the desired accuracy. In such cases, we end up dealing with too many time segments. Solving online stationary problems in a large number of time segments in these instances leads to excessive computational cost. Therefore, we can only apply piecewise constant approximation when the rate functions $\lambda_j(t)$'s all change extremely slowly in a time segment of reasonable length. In our algorithm, we would refer to these time segments as type A.

If the rate functions change moderately slowly, and we are unable to find a time segment of sufficient length on which to perform piecewise constant approximation, we turn to a different approach. Instead of approximating the arrival rates as constants, we instead approximate the probability that the next customer arrival is of type j directly. In particular, we find time segments in which the difference between the upper and lower bounds of this probability is small. We then pick a random value between the upper and lower bounds to be the approximated probability in that time segment, without incurring significant loss in accuracy. In our algorithm, we refer to these time segments as type B. To identify such a time segment, we first introduce Lemma 4.2, which can be proved by contradiction.

LEMMA 4.2. *If f is a continuous function defined in some domain D, one can divide D into disjoint segments $I_1, I_2, ...$, such that in each segment, the amount that f changes is bounded by a given threshold v.*

By choosing an appropriate threshold $v \gg \epsilon$ that bounds the amount of the change of rate functions, one can control the amount of inaccuracy incurred by the approximation of the probability of next customer arrival being of type j. This is captured in the following Theorem:

THEOREM 4.3. *Assume that during time period $[t_1, t_2]$, for some $v \gg \epsilon > 0$,*

$$\max_{t \in [t_1, t_2]} \lambda_j(t) - \min_{t \in [t_1, t_2]} \lambda_j(t) \le v \quad \forall 1 \le j \le m.$$

Let $U(j)$, $L(j)$ denote the upper and lower bound of probability that the arriving customer is of type j during the time period $[t_1, t_2]$. Let $Y = \sum_{j=1}^{m} \max_{s\in[t_1,t_2]} \lambda_j(s)$ and $y = \sum_{j=1}^{m} \min_{s\in[t_1,t_2]} \lambda_j(s)$. Then,

$$U(j) \le \frac{\max_{s\in[t_1,t_2]} \lambda_j(s)}{y}, \tag{4.1}$$

$$L(j) \ge \frac{\min_{s\in[t_1,t_2]} \lambda_j(s)}{Y}. \tag{4.2}$$

It follows that if $\delta(j) = U(j) - L(j)$, we have:

$$\delta(j) \le \frac{mv^2 + (y + m\min_{s\in[t_1,t_2]} \lambda_j(s))v}{y^2 + mvy}. \tag{4.3}$$

Proof. Note that $\max_{s\in[t_1,t_2]} \lambda_j(s) \le \min_{s\in[t_1,t_2]} \lambda_j(s) + v$, $Y \le y + mv$, then by (4.1) and (4.2), we get

$$\delta(j) \le \frac{\max_{s\in[t_1,t_2]} \lambda_j(s)}{y} - \frac{\min_{s\in[t_1,t_2]} \lambda_j(s)}{Y} \le \frac{mv^2 + (y + m\min_{s\in[t_1,t_2]} \lambda_j(s))v}{y^2 + mvy} \tag{4.4}$$

∎

(4.3) demonstrates the relationship between the chosen threshold v and the difference between the upper and lower bound $\delta(j)$. In the case when the rate functions change moderately slowly, one can obtain an approximation of the probability that the next customer arrival is of type j by controlling the changes of the arrival rate functions inside each time segment. It is noteworthy that $\delta(j)$ not only depends on v, but also depends on rate functions inside specific time segments. Therefore, even if we seek a constant confidence bound $\delta(j)$, the desired values of v will vary at different times.

Algorithm 4.1 Time Segmentation

Input: rate functions $\lambda_j(t)$, time span $[t_0, t_{\text{end}}]$, parameters $\epsilon, \delta, d > 0$
Output: time segments $I_1, I_2, ..., I_l$

while $t < t_{\text{end}}$ **do**
  for $j = 1, ..., m$ **do**
    compute the largest t_j s.t. $\max_{s\in[t,t_j]} \lambda_j(s) - \min_{s\in[t,t_j]} \lambda_j(s) \le \epsilon$
  end for
  $t^* \leftarrow \min_{j\in[m]} t_j$
  if $t^* - t \ge d$ **then**
    Add $[t, t^*]$ as one of the time segments, and mark it as type A; $t \leftarrow t^*$
    break
  else
    Solve $mv^2 + (\sum_j \lambda_j(t) + m\lambda_j(t) - \delta m \sum_j \lambda_j(t))v - \delta(\sum_j \lambda_j(t))^2 = 0$ for v
    for $j = 1, ..., m$ **do**
      compute the largest t'_j s.t. $\max_{s\in[t,t'_j]} \lambda_j(s) - \min_{s\in[t,t'_j]} \lambda_j(s) \le v$
    end for
    $t^* \leftarrow \min_{j\in[m]} t'_j$
    Add $[t, t^*]$ as one of the time segments, and mark it as type B; $t \leftarrow t^*$
  end if
end while

In Algorithm 4.1, we propose a time segmentation algorithm that divides the entire time span into small time segments. Note that we always first look for time segments of type A, in which we can perform piecewise constant approximation. However, if such time segments do not have sufficient length, we turn to look for time segments of type B, for which we can approximate the probability of the next customer arrival being a particular type. For the time segments $\{I^{(A)}\}$ that are marked as type A, we select a random $t \in I^{(A)}$ and approximate the arrival rates as a fixed number: $\lambda_j(t) \approx \lambda_j$. We can then solve the non-stationary problem in this time segment as an online stationary problem. On the other hand, for the time segments $\{I^{(B)}\}$ that are marked as type B, we compute $U(j)$ and $L(j)$ as in (4.1) and (4.2). We then approximate $\frac{\lambda_j}{\sum_{s=1}^{m}\lambda_s}$ as a random number between $U(j)$ and $L(j)$. Note that these approximated numbers do not necessarily sum up to one, but their sum would not deviate too much from one as long as each $\delta(j) = U(j) - L(j)$ is sufficiently small, which can be controlled by threshold v. In this way, we again approximate the non-stationary problem as a stationary problem in this time segment. To solve the online stationary problem in each time segment, we simply apply the integrated Algorithm 3.2, using both UCB and online GD. In practice, when the number of customer arrivals in each time segment is large, the dual variables should converge before reaching the end of the time segment.

4.2. Upper Bound of Average Regret. To solve an online non-stationary problem, we first apply Algorithm 4.1 to divide the time span into small time segments, and then apply Algorithm 3.2 to solve the online stationary problem within each time segment. As in Section 3, we now provide an analysis of the average regret bound of this approach. Since the online non-stationary algorithm involves both the time segmentation algorithm and the integrated algorithm, the regret computation here requires our previous analysis of UCB and online GD. Our analysis over the average regret bound here will be focused on a single time segment.

THEOREM 4.4. *Consider the online non-stationary algorithm described in Subsection 4.1, in a specific time segment I, we have:*

$$\limsup_{T\to\infty} \frac{\text{regret}_T}{T} \leq R\delta \tag{4.5}$$

where T is total number of customer arrivals in I, δ is the confidence bound used for type B time segments and $R = m(\mu \log n + r^*)$, where $r^* = \max_{i\in[n]} r_i$.

Proof. As in the proof of Theorem 3.2, we define $f_t(\Lambda, P)$ to be the dual objective at time t. We let $\Lambda^* = \text{argmin}\sum_{t=1}^{T} f_t(\Lambda, P^*)$, where P^* is the ground truth preference matrix. We denote Λ_t and $P^{(t)}$ as the dual variable and preference matrix obtained by our algorithm at time t. Here, we additionally define $f_t^{'}(\Lambda_t, P^{(t)})$ to be the dual objective function of the online non-stationary problem:

$$f_t^{'}(\Lambda_t, P^{(t)}) = \mu \sum_{j=1}^{m} \bar{P}_j^{(t)} \log(Z_j(\Lambda_t, P^{(t)}))\phi_j(t) + \frac{1}{T}\langle \Lambda_t, b\rangle,$$

where $\bar{P}_j^{(t)} = \max_i P_{ij}^{(t)}$, $Z_j(\Lambda_t, P^{(t)}) = \sum_{i\in[n]} \exp\left(\frac{(r_i - \Lambda_{t,i})P_{ij}^{(t)}}{\bar{P}_j^{(t)}\mu}\right)$, and $\phi_j(t)$ represents the ground truth probability that the next customer arrival is of type j at time t. The

regret achieved at time t is thus:

$$\begin{aligned}\text{regret}_T^t &= f_t^{'}(\Lambda_t, P^{(t)}) - f_t(\Lambda^*, P^*) \\ &\leq |f_t^{'}(\Lambda_t, P^{(t)}) - f_t(\Lambda_t, P^{(t)})| + |f_t(\Lambda_t, P^{(t)}) - f_t(\Lambda_t, P^*)| \\ &\quad + |f_t(\Lambda_t, P^*) - f_t(\Lambda^*, P^*)|,\end{aligned} \tag{4.6}$$

As before, $P^{(t)}$ is the preference matrix we have at time t while P^* is the ground truth preference matrix.

Using the same techniques as in the proof of Theorem 3.2, we can show that the average of the last two terms in (4.6) will converge to zero as $T \to \infty$. Hence, it suffices to show the convergence of the average of the first term, which captures the regret from approximating the non-stationary problem with a stationary problem. Recall that the dual objective of the online-stationary problem is defined as follows in (3.4):

$$f_t(\Lambda_t, P^{(t)}) = \mu \sum_{j=1}^{m} \frac{\lambda_j}{\sum_{s=1}^m \lambda_s} \bar{P}_j^{(t)} \log(Z_j(\Lambda_t, P^{(t)})) + \frac{1}{T}\langle \Lambda_t, b\rangle,$$

where

$$Z_j(\Lambda_t, P^{(t)}) = \sum_{i=1}^{n} \exp\left(\frac{(r_i - \Lambda_{t,i})P_{ij}^{(t)}}{\mu \bar{P}_j^{(t)}}\right) \leq n \exp\left(\frac{r^*}{\mu}\right),$$

since $P_{ij}^{(t)} \leq \bar{P}_j^{(t)}$. Hence, we must have

$$\log(Z_j) \leq \log(n) + \frac{r^*}{\mu} \qquad \forall j \in [m].$$

Additionally, note that

$$\left|\frac{\lambda_j}{\sum_{s=1}^m \lambda_s} - \phi_j(t)\right| \leq \max\left(\frac{\epsilon}{m \min_s \lambda_s}, \delta\right),$$

where the first term in the maximum on the right hand side results from time segments of type A, and the second term results from the confidence bound δ in time segments of type B. Realistically, the number of customers who arrive at an online marketplace per second are of the order of thousands or millions. Therefore, the customer arrival rates λ_s's are of substantial magnitude. We can choose ϵ sufficiently small such that $\frac{\epsilon}{m \min_s \lambda_s} \ll \delta$. Now, if we define $R = m(\mu \log n + r^*)$, we can bound the difference between $f_t(\Lambda_t, P^{(t)})$ and $f_t^{'}(\Lambda_t, P^{(t)})$ with the following:

$$|f_t^{'}(\Lambda_t, P^{(t)}) - f_t(\Lambda_t, P^{(t)})| \leq R\delta,$$

It follows that:

$$\limsup_{T\to\infty} \frac{\text{regret}_T}{T} \leq \limsup_{T\to\infty} \frac{\sum_{t=1}^T |f_t^{'}(\Lambda_t, P^{(t)}) - f_t(\Lambda_t, P^{(t)})|}{T} \leq R\delta. \tag{4.7}$$ ∎

From Theorem 4.4, we can see that the average regret does not converge to 0, but instead converges to a constant. In particular, δ corresponds to the confidence bound that we use in the type B time segments. Theoretically, by setting δ sufficiently small, we can control the average regret to converge to a number close to 0. Another trade-off certainly needs to be taken into account: as we decrease the regret, computational complexity will increase. However, as our numerical experiments later demonstrates, as long as we keep the value of confidence bound reasonably small, the average regret tends to become negligible as the number of customers get larger.

5. Numerical Study. In this section, we performed a number of numerical experiments to test the efficacy of our proposed algorithms, and all experiments were run on macOS Sierra with a 2.7 GHz Intel Core i5 processor and 8GB memory, using code written in Python. We create different synthetic datasets that simulate customer preferences and arrivals following stationary or non-stationary Poisson processes. In order to make our problem non-trivial, we choose set-ups which guarantee that certain products will be sold out, while other products will have remaining budget in the optimal offline solution. We first apply Algorithm 3.2 to solve a set of online stationary problems and compare its empirical performance against a greedy heuristic. We then present experiments with non-stationary Poisson customer arrival processes and apply Algorithm 4.1 along with the integrated algorithm. In all of the experiments, we examine the convergences of the average regrets and check whether they match the theoretical results in Section 3 and Section 4. The results of these numerical experiments confirm the effectiveness of our approach in tackling resource allocation problems in different settings.

5.1. Online Stationary Experiments. We first apply Algorithm 3.2 to a series of online stationary problems with varying numbers of customer arrivals. We compare its performance with a greedy heuristic, which simply matches each incoming customer to the product with largest reward available. The metric for evaluating algorithm performance is the average regret, as defined in Definition 3.1.

We test our online stationary problem with four different sizes of total customer arrivals: $1,000, 10,000, 100,000, 1,000,000$. Each of them has the same initial set-up:

- There are $m = 10$ types of customers and $d = 10$ products to be assigned.
- The j^{th} type of customer arrives as a stationary Poisson process with constant arrival rate $\lambda_j = 0.1j$.
- Each customer's ground truth preference is drawn from a normalized beta distribution $\text{Beta}(\alpha, \beta)$, where $1 \leq \alpha \leq 3$ and $3 \leq \beta \leq 6$, so that the buying behavior of different types of customers differ from each other. Our algorithm does not have any previous knowledge of the ground truth preference matrix P^*; instead, it learns preferences as the customers arrive.
- The budgets of products range between 10% and 30% of the total population, while their rewards range from 0.1 to 1. The products with higher rewards tend to have lower budgets, and there exists a trade-off between the number of products that can be assigned and the reward value it can generate.

With the above set-up, we demonstrate the potential of integrated algorithm in solving a challenging resource allocation problem.

We first examine the algorithm's ability in learning customer preferences. Recall that in the beginning of the algorithm, we do not have any past knowledge of P_{ij}. As each customer arrives, we assign them to a product either by UCB or by the solution we reach from online GD. After a type j customer gets assigned a product i, they will accept or decline the item based on their buying preferences. Based on this new outcome, we update the entry P_{ij} to more accurately reflect this customer's preferences. We expect that as more customers arrive and get assigned to different types of products, our preference matrix P will eventually converge to the ground truth matrix P^*. In the experiment with 100,000 customers, we perform the UCB algorithm for the first 20,000 incoming customers and rely on the solution from online GD afterwards. Figure 1 shows the convergence of P under this setting, in which we can see that P hits the convergence horizon, approaching P^* in the first 5,000 customer arrivals. After we stop applying the UCB algorithm, the value of $||P - P^*||_F$ remains

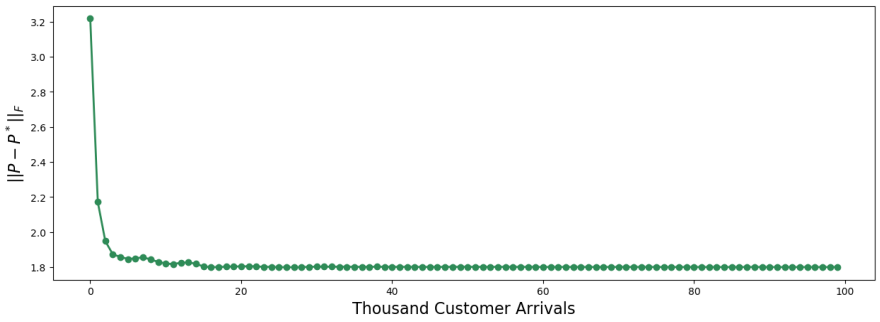


Fig. 1: The Frobenius norm between the ground truth P^* and our preference matrix P with every thousand customer arrivals.

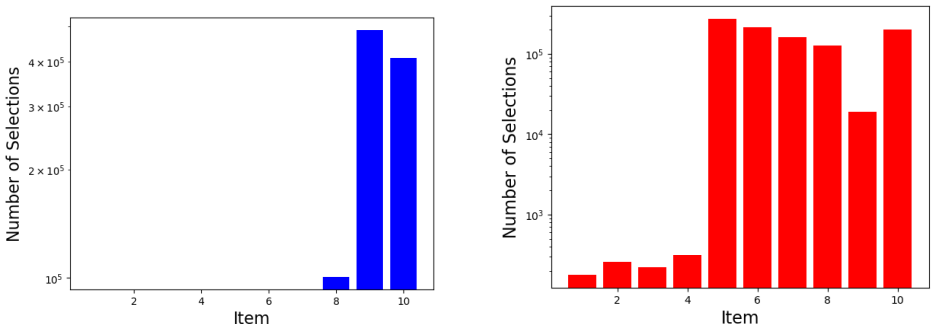
stable because the customer preferences that we have learned closely matches the actual purchasing behavior. We additionally note that P no longer approaches P^* quickly after the first few thousand arrivals. This is because after the application of the UCB algorithm in the beginning, the algorithm develops a good understanding of which customers have higher probabilities of purchasing certain products, thus avoiding matching those customers to products they are unlikely to buy. Therefore, it is difficult for the customer preferences for those products to approach extreme accuracy. However, since most entries of P and P^* are sufficiently close, the remaining inaccuracy will not prevent the algorithm from making an optimal product allocation, and the regret introduced is minimal.

Number of Customers	Offline Dual Optimal Objective Value	Online Dual Objective Value	Average Regret	Runtime
1,000	324.93	861.18	0.536	0.7s
10,000	3251.91	4671.50	0.142	7.5s
100,000	32480.30	29947.76	0.051	120s
1,000,000	325171.14	295788.51	0.032	300s

Table 1: Results of online stationary experiments. We obtain the optimal offline dual objective by applying the first-order method GD^{lin} [4].

In Table 1, we record the average regret and runtime obtained by Algorithm 3.2. We can clearly see that the average regret decreases as the size of data gets larger. While the performance of the integrated algorithm is far from optimal in the first 1,000 customer arrivals, the dual variable already converges to the near-optimal solution when the size of the customer reaches 100,000, thus leading to a much smaller average regret. To further reduce the computational costs, we additionally choose to apply UCB and online GD sufficient number of times such that the dual variable no longer experiences drastic changes, i.e., when $||\Lambda_t - \Lambda_{t-1}|| < \gamma$ for some small threshold γ. When this threshold condition is reached at time $t = T$, we would simply assign

products based on Λ_T from then on. We observe that in the experiment with 100,000 customer arrivals, after applying around 20,000 rounds of UCB and around 40,000 rounds of online GD, the dual variable already converges and requires no further computation. Therefore, we expect the runtime of the algorithm to remain at a considerably small value, as shown in the last column.



(a) Item selection by the greedy algorithm.

(b) Item selection by Algorithm 3.2.

Fig. 2: Number of times each product is selected.

We have compared the results of Algorithm 3.2 with those of the greedy heuristic by directly comparing the revenue generated from the two approaches. In the greedy heuristic, each incoming customer is shown the product available with the highest reward until that product is fully consumed. In Figure 2, we show the number of times that each product is presented to customers by the greedy algorithm and the integrated algorithm, respectively. We can tell that the integrated algorithm is not greedy since it does not select an item solely based on its reward value. However, each customer might have a different preference for the product with highest reward, so intuitively we expect this approach to be somewhat naive and not necessarily lead to the optimal outcome.

Number of Customers	Offline Revenue	Greedy Algorithm Revenue	Integrated Algorithm Revenue
1,000	324.93	198.00	154.00
10,000	3251.81	2024.80	2841.40
100,000	32413.36	20365.60	31438.10
1,000,000	324240.92	203584.80	315435.10

Table 2: Revenues generated with the offline approach, the greedy algorithm and Algorithm 3.2.

This disparity in customer preferences is also indicated by our experiment results, which are shown in Table 2. The first column records the optimal revenue that we

can achieve if we are to solve the corresponding offline problem. When the number of customer arrivals is small (e.g., 1,000), the greedy approach gives a higher revenue than the integrated algorithm; this is because the integrated algorithm has not gone through a sufficient number of online GD iterations for the dual variable to converge. As the sizes of data later increases, we observe that the revenue generated by the integrated algorithm is closer to the optimal revenue achieved in the offline problem and also exceeds that of the greedy heuristics. We additionally note that there can be cases where the greedy heuristics might give better performance. For instance, when the customer preferences for each product are close to each other, choosing the product with the highest reward is essentially the optimal solution. However, since customer preferences in realistic settings tend to have more variance, the integrated algorithm would almost always allocate the better product.

5.2. Online Non-Stationary Experiments. We now move on to test the performance of the proposed online non-stationary algorithm, which combines Algorithm 3.2 and Algorithm 4.1. Recall that we do this by converting the non-stationary problem into a series of stationary problems and then solving each of the stationary problems accordingly. In this subsection, we present two representative experiments, each having initial set-ups that make the problem non-trivial: the first experiment comes with extreme budget constraints, while the second is closer to a realistic setting, where the products have diverse rewards.

5.2.1. Experiments with Extreme Budget Constraints. In the first set of experiments for the non-stationary problem, the following set up is considered:

- There are $m = 10$ types of customers and $d = 10$ products to be assigned.
- Each type of customer is associated with an arrival rate function that changes fairly slowly. Figure 3 shows some example rate functions that we consider.
- We draw the ground truth customer preference matrix P^* randomly from a Uniform$(0, 1)$ distribution, sampling with replacement, with probability weights on 0.9 and 0.1, as we want to create scenarios where the probability of a customer getting recommended each item is low or high.
- One product has infinite budget, two have small budgets (10% of the population) and the rest have minimal budgets (1% of the population).
- The reward of each product is set to be uniformly 1.

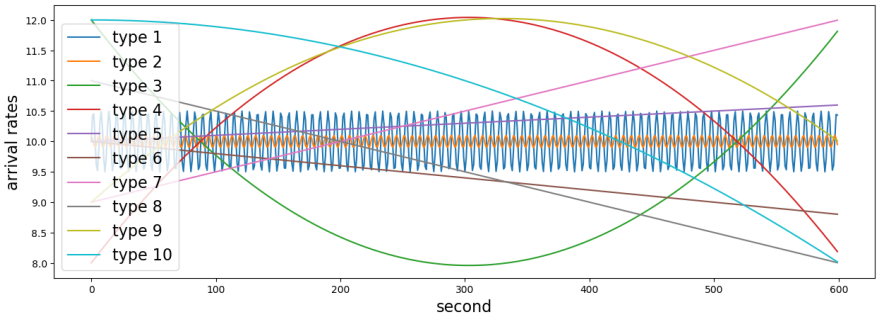
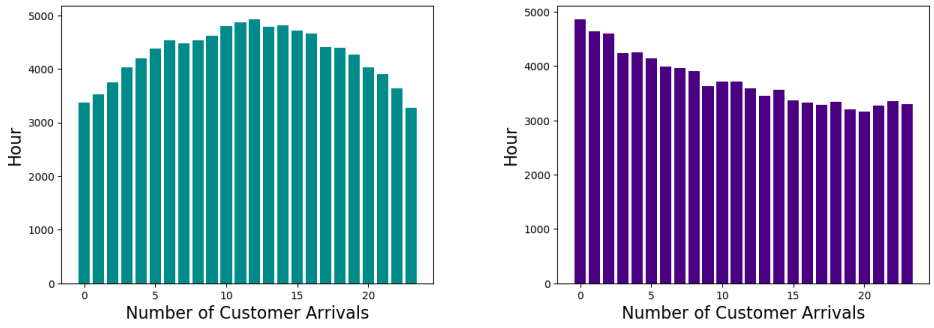


Fig. 3: The rate functions $\lambda_j(t)$ of each type of customer used in the experiment with 60,000 customer arrivals. There are two trigonometric functions, four linear functions and four quadratic functions.



(a) Customers Type 4. (b) Customer Type 9.

Fig. 4: Number of customer arrivals each hour in the 1M experiment.

We performed experiments using three different sizes of data, which includes: (1) a population of 6,000 people arriving in an hour, (2) a population of 60,000 people arriving in 10 hours, and (3) a population of 1,000,000 people arriving in 24 hours. We scaled the rate functions in accordance with the length of the time span to keep the experimental set-ups consistent. In Figure 4, we plot the number of customer arrivals each hour in the experiment with 1,000,000 customers for two different types of customers. We can clearly observe that the numbers vary with the hours, and that different types of customers have different arrival patterns.

In Figure 5, we plot the number of each product assigned to the customers in the experiment with 1,000,000 customer arrivals. The result is as expected: all the products with minimal or large budgets have been sold to customers who have higher preference for those products, and the only product that has remaining budget is the one with infinite budget. As before, we not only care about the assignment of items, but also how close our dual variable is to the optimal solution. We thus move on to compute the online dual objective and evaluate its performance via average regret.

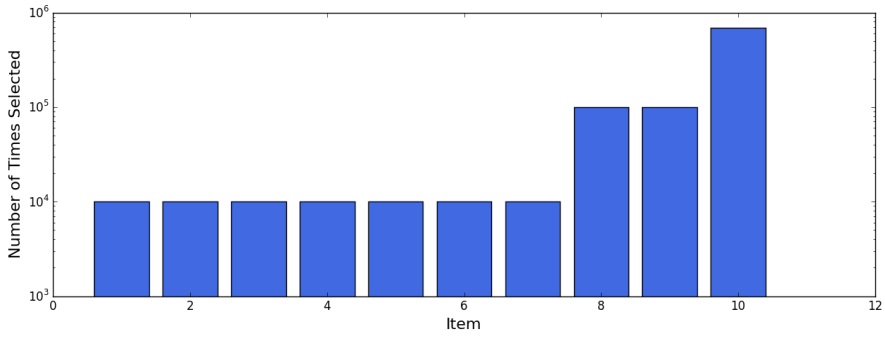


Fig. 5: The item assignment of the 1M experiment with extreme budget constraints. Items are sorted in ascending order by values of their budget constraints.

Table 3 reports the results of the experiments across all three different sizes of data. We computed the optimal offline dual objective mentioned in Subsection 2.1.

We computed the online dual objective using (3.4), and the total regret using Definition 3.1. We observe that average regret decreases as we increase the size of data, which confirms the theoretical result in Theorem 4.4. When applying the non-stationary algorithm, we set the number of rounds of UCB and online GD we wish to apply inside each time segment, which mainly determines the runtime of the algorithm. Here, as we make the size of the data larger, the number of rounds of UCB and gradient computations needed to achieve the convergence of the dual variable increases accordingly, and the runtime scales up roughly linearly.

Number of Customers	Offline Dual Optimal Objective Value	Online Dual Objective Value	Average Regret	Runtime
6,000	612.54	900.58	0.0454	5s
60,000	6222.43	6843.15	0.0101	40s
1,000,000	98547.03	105854.78	0.0076	940s

Table 3: Results of experiment with extreme budget constraints. We obtain the optimal offline dual objective by applying $\mathrm{GD}^{\mathrm{lin}}$.

5.2.2. Experiments with Varying Rewards. We also performed another experiment with varying reward values such that the assignment of the items are not solely based on their budget constraints and customer preferences, which reflects a more realistic setting. The set-ups of this experiment remain the same as the previous experiment, with the following exceptions:

- Instead of applying the extreme budgets constraints as before, we select one product to have fairly large budget (66.67% of the population) and let the rest of the products have fairly small budgets (10% of the population).
- The rewards of products vary from 0.2 to 1. In particular, the product with the most budget is associated with a reward value of 0.2, so this product should be the least favorable to most customers.

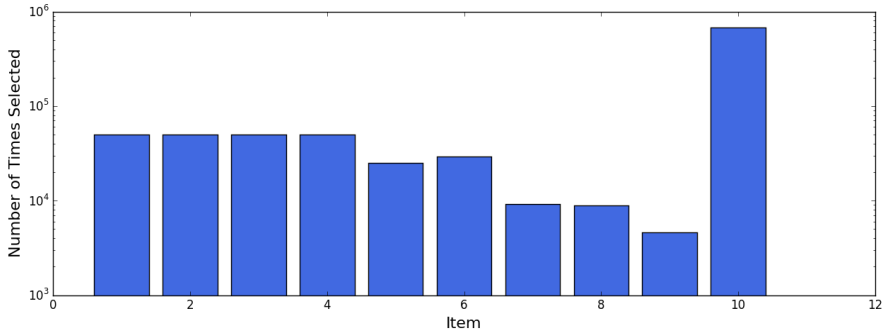


Fig. 6: The item assignment of the experiment with 1,000,000 customer arrivals and varying rewards. Items are sorted in descending order by values of their rewards.

The result of the experiment with 1,000,000 customer arrivals can be seen in

Figure 6. Observe that our algorithm ensures that the four items with highest rewards have been fully sold out, while the rest still have remaining budgets in the end. It is noteworthy that even the items with low rewards have been sold to some extent, due to the exploration component of the UCB algorithm, as explained in Subsection 3.2. Overall, such a solution matches our expectation of a near-optimal solution.

We again compared the results across three different sizes of data, which can be seen in Table 4. As before, we can observe that average regret decreases as the number of customer arrivals increase. The runtime is similar to the first set of experiments, which are reasonable in terms of the size of data. Overall, our experiments have shown that the non-stationary algorithm has a good potential of being applied to real-world online product allocation problems.

Number of Customers	Offline Dual Optimal Objective Value	Online Dual Objective Value	Average Regret	Runtime
6,000	608.97	549.35	0.0199	4s
60,000	6217.12	5539.41	0.0114	36s
1,000,000	97060.83	95482.46	0.0025	957s

Table 4: Results of experiment with varying rewards.

6. Conclusion. In this work, we propose algorithms that tackle the online resource allocation problem, in which we aim to recommend each customer with an item in ways that not only maximize potential reward, but also satisfy budget constraints. In order to find the optimal solution to our online objective function, we first must learn the preferences, or P_{ij}, for each customer type. To learn the probability that a customer in a certain type purchases a given item, we use the Upper Confidence Bound (UCB) algorithm, which decides which item to recommend to a customer. When the customer arrives, we observe whether or not they have purchased the item recommended to them and update our customer preference variable, P_{ij}. We incorporate this value into our objective function, and apply online Gradient Descent to minimize our dual function. Over time, as more customers arrive, the estimations for the P_{ij} values become more accurate, and the UCB algorithm is able to make better recommendations, ones that have a higher probability of reward. Overall, our online stationary algorithm combines reinforcement learning with online optimization to minimize our dual function and find the optimal solution. Our tests on this novel algorithm support our theory that regret of this algorithm approaches zero when the number of customers is sufficiently large, and that our algorithm produces a better solution than greedy heuristics.

Although our online stationary algorithm performs well, customers do not always arrive following a stationary Poisson process. In a more realistic scenario, the rate at which customers arrive varies over time. This motivated us to consider the online non-stationary problem, in which customers arrive onto the webpage following a non-stationary Poisson process. When we remove the assumption of customer arrival following a stationary Poisson processes, we are met with complexity in formulating the online convex optimization problem, as this type of problem cannot simply be solved using the proposed online stationary algorithm. An additional difficulty lies in

that if we do online learning for P_{ij}, the dual variables will not necessarily converge. Moreover, if the arrival processes are modeled as non-stationary Poisson processes, then the probability that the new arriving customer comes from type j is almost impossible to compute. Our non-stationary algorithm approaches these difficulties by dividing the non-stationary problem into several stationary problems, under the assumption that the arrival rate functions of different customers change fairly slowly in a small time segment. We have shown theoretically that the regret of the online non-stationary algorithm should approach a small value near zero when the number of customers is sufficiently large. Our empirical results with both extreme budget constraints and non-trivial budget constraints also support this theoretical result.

7. Future Work. There are many rich, exciting directions that one can pursue with this work. In the product recommendation model we propose above, we have considered a rather simplified scenario, matching each customer with one product at a time and aiming to maximize the expected profit brought by this assignment. However, one can in fact make the current model more realistic by introducing more complications: (1) When each customer arrives, an e-commerce platform can in fact display a set of products to the customer at the same time. (2) The user engagement that a website wishes to maximize is not necessarily the reward values, but the number of clicks or the dwell time that a user spend on the webpages. (3) Sometimes there are more business contraints to consider, e.g., one needs to guarantee a fixed number of selections for a certain product.

One potential future direction of this work is to take the additional settings above into the model construction, and develop variants of the proposed algorithms that can deal with these more complicated situations. Specifically, it would be useful to improve the practicality of the algorithm by considering a dynamic segmentation approach, such as recursive partitioning. Since we group customers by buying preferences, which is static, we run into the problem of a positive feedback loop: by recommending the same products to each customer type, we may reinforce our recommendations over time. Dynamic segmentation approaches, therefore, would be useful in circumventing this issue.

Further, the budget constraints considered in the non-stationary experiments are mostly extreme constraints, in which one product has a much larger budget than the others. However, existing work like [8] suggests that inventory levels may also have an impact a firm's strategies. It would be interesting to consider other types of budget constraints and examine how the algorithm adapts to different budget scenarios.

In addition, improvements can also be made towards the performance of the proposed online stationary and non-stationary algorithms. While the runtime of the algorithms are reasonable considering the large scale of the data, one might further decrease the runtime of these algorithms with the application of parallel computing. This would enable our algorithms to have the potential of being applied in real-world settings, where e-commerce companies oftentimes need to deal with even larger scale of customer arrivals in a shorter period of time, e.g., a million customer arrivals within a second. Throughout our analysis of algorithm performance, we have only tested our algorithms with synthetic data; therefore, we are also interested in understanding how they perform when dealing with real-world datasets.

Appendix A. Notations. Table 5 records symbols used throughout the paper.

n	Number of items
m	Number of customer types
i	Item indices, $i \in [1, n]$
j	Customer type indices , $j \in [1, m]$
r_i	Reward in terms of revenue for the company for a given customer buying certain item i
b_i	Budget constraint of item i, $b \in \mathbb{R}^n$
x_{ij}	Probability that a customer of type j gets recommended item i
P_{ij}	Probability that a customer of type j will buy item i given that they are offered item i
P^*	Ground truth preference matrix
Λ^*	Optimal dual variable
$\bar{P}_j$	For a given customer of type j, his highest possibility of buying any particular product, i.e., $\bar{P}_j = \max_i P_{ij}$
μ	Regularization parameter
Λ	Dual variable vector of dimension n
η_t	Step size in optimization algorithm at an iteration

Table 5: Table of Notations

Acknowledgments. This project was completed during the Research in Industrial Projects for Students (RIPS) 2019, under the sponsorship of the Institute for Pure and Applied Math (IPAM) at UCLA and the Alibaba Group. We would like to thank our academic and industry mentors Anna Ma, Xinshang Wang, and Wotao Yin for their help discussions. We would also like to thank Susana Serna, Dima Shlyakhtenko and all of the IPAM staff who made RIPS 2019 possible.

REFERENCES

[1] S. Agrawal and N. R. Devanur, *Bandits with concave rewards and convex knapsacks*, in Proceedings of the fifteenth ACM conference on Economics and computation, ACM, 2014, pp. 989–1006.

[2] S. Agrawal and N. R. Devanur, *Fast algorithms for online stochastic convex programming*, in Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 2015, pp. 1405–1424.

[3] S. Agrawal, Z. Wang, and Y. Ye, *A dynamic near-optimal algorithm for online linear programming*, Operations Research, 62 (2014), pp. 876–890.

[4] Z. Allen-Zhu, D. Simchi-Levi, and X. Wang, *The lingering of gradients: how to reuse gradients over time*, in Advances in Neural Information Processing Systems, 2018, pp. 1244–1253.

[5] P. Auer, *Using confidence bounds for exploitation-exploration trade-offs*, J. Mach. Learn. Res., 3 (2003), pp. 397–422.

[6] A. Badanidiyuru, R. Kleinberg, and A. Slivkins, *Bandits with knapsacks*, in 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, IEEE, 2013, pp. 207–216.

[7] W. C. Cheung, W. Ma, D. Simchi-Levi, and X. Wang, *Inventory balancing with online learning*, arXiv preprint arXiv:1810.05640, (2018).

[8] T. Dai and K. Jerath, *Salesforce compensation with inventory considerations*, Management Science, 59 (2013), pp. 2490–2501.

[9] N. R. Devanur and T. P. Hayes, *The adwords problem: online keyword matching with budgeted bidders under random permutations*, in Proceedings of the 10th ACM conference on Electronic commerce, ACM, 2009, pp. 71–78.
[10] J. Feldman, A. Mehta, V. Mirrokni, and S. Muthukrishnan, *Online stochastic matching: Beating 1-1/e*, in 2009 50th Annual IEEE Symposium on Foundations of Computer Science, IEEE, 2009, pp. 117–126.
[11] E. Hazan et al., *Introduction to online convex optimization*, Foundations and Trends® in Optimization, 2 (2016), pp. 157–325.
[12] T. Lattimore and C. Szepesvári, *Bandit algorithms*, preprint, (2018).
[13] A. Mehta et al., *Online matching and ad allocation*, Foundations and Trends® in Theoretical Computer Science, 8 (2013), pp. 265–368.
[14] K. V. Natarajan and J. M. Swaminathan, *Global health*, Handbook of Healthcare Analytics: Theoretical Minimum for Conducting 21st Century Research on Healthcare Operations, (2018), pp. 137–158.
[15] S. Shalev-Shwartz et al., *Online learning and online convex optimization*, Foundations and Trends® in Machine Learning, 4 (2012), pp. 107–194.
[16] C. Stein, V.-A. Truong, and X. Wang, *Advance service reservations with heterogeneous customers.*, 2018.
[17] W. Zhong, R. Jin, C. Yang, X. Yan, Q. Zhang, and Q. Li, *Stock constrained recommendation in tmall*, in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 2287–2296.
[18] M. Zinkevich, *Online convex programming and generalized infinitesimal gradient ascent*, in Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 928–936.