Probabilistic Simulation of Spatial Demand for Intelligent Product Allocation

Porter Jenkins
Brigham Young University
pjenkins@cs.byu.edu

Hua Wei
New Jersey Institute of Technology
hua.wei@njit.edu

J. Stockton Jenkins
Brigham Young University
sjenkin2@byu.edu

Zhenhui Li
Pennsylvania State University
zul17@psu.edu

ABSTRACT

Connecting consumers with relevant products is a very important problem in both online and offline commerce. In many offline retail settings, product distributors extend bids to place and manage product displays within a retail outlet. The distributor aims to choose a spatial allocation strategy that maximizes revenue given a preset budget constraint. Prior work shows that carefully selecting product locations within a store can minimize search costs and and induce consumers to make "impulse" purchases. Such impulse purchases are influenced by the spatial configuration of the store. However, learning important spatial patterns in offline retail is challenging due to the scarcity of data and the high cost of exploration and experimentation in the physical world. To address these challenges, we propose a stochastic model of spatial demand in physical retail, which we call the Probabilistic Spatial Demand Simulator (`PSD-sim`). `PSD-sim` is an effective mirror of the real environment because it exploits the structure of common retail datasets through a hierarchical parameter sharing structure, and is able to incorporate spatial and economic knowledge through informative priors. We show that `PSD-sim` can both recover ground truth test data better than baselines, and generate new data for unseen states. The simulator can naturally be used to train policy estimators that discover intelligent, spatial allocation strategies. Finally, we perform a preliminary study into different optimization techniques and find that Deep Q-Learning can learn an effective spatial allocation policy.

CCS CONCEPTS

• **Computing methodologies** → **Modeling and simulation**; *Bayesian network models*;

KEYWORDS

Retail Optimization, Probabilistic Simulation, Reinforcement Learning

GeoSim'21, November 2, 2021, Beijing, China

ACM ISBN 978-1-4503-9101-6/21/11…$15.00
https://doi.org/10.1145/3486184.3491078

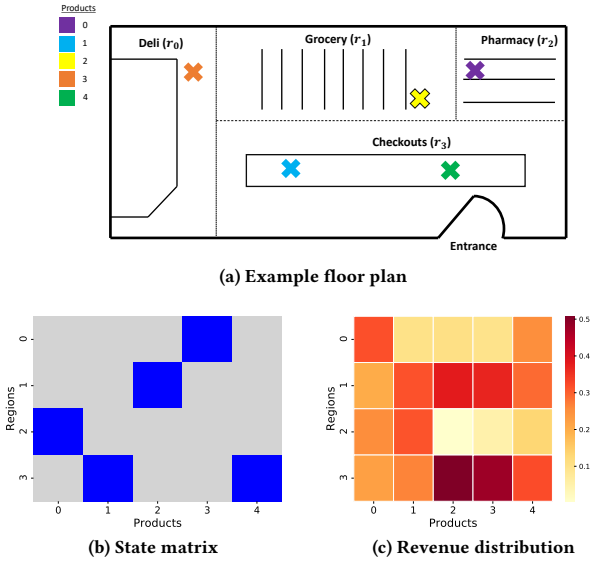
ACM Reference format:
Porter Jenkins, Hua Wei, J. Stockton Jenkins, and Zhenhui Li. 2021. Probabilistic Simulation of Spatial Demand for Intelligent Product Allocation. In *Proceedings of 4th ACM SIGSPATIAL International Workshop on GeoSpatial Simulation , Beijing, China, November 2, 2021 (GeoSim'21),* 10 pages.
https://doi.org/10.1145/3486184.3491078

1 INTRODUCTION

Over the past two and half decades the internet has transformed the nature of commerce. E-commerce platforms have thrived because of the data rich environment of the world wide web. Through data driven methods, internet retailers are able to find surpluses in demand and connect the right product with the right consumer. Simultaneously, many offline distributors and retailers have fallen behind partially due to the data poor environment of the offline world.

Consider the case of consumer packaged goods (CPG): a merchandiser for a large distributor (e.g., Coca-cola) makes bids to place product displays inside of a large retailer (e.g., Wal-mart). A large retailer may offer hundreds of product displays. The distributor historically makes allocation decisions using heuristics or intuition. For example, he or she may arbitrarily choose to allocate Coca-cola products near the deli, when the products may actually drive more revenue if placed near the checkout (see Figure 1). In an effort to close this gap and improve the effectiveness of offline retail processes, we study the product allocation problem in physical retail. In this problem, the distributor aims to allocate product across the store by placing bids on discrete locations. The key question for the distributor is, *how to choose locations for its products to maximize revenue subject to a budget constraint?*

Effective spatial allocation strategies can increase revenue by connecting products with consumers. On one hand, consumers might find it difficult to locate what they really need in a store. Proper placement can reduce the consumer's search costs and help sell more products. On the other hand, consumers often purchase products on an impulse, i.e., buying products they had not intended to buy beforehand. For example, suppose a shopper visits a supermarket intending to purchase groceries. As the shopper checks out he sees a soft drink beverage placed near the cash register, and adds it to his cart. The shopper's decision to purchase the drink was in part a function of the environmental cues and placement of the product [19]. Proper placement can maximize such "impulse" purchases[3].



(a) Example floor plan

(b) State matrix

(c) Revenue distribution

Figure 1: An example of the spatial product allocation problem in physical retail. We provide a sample floor plan of a small, retail environment (a). Each section of the store is partitioned into "regions" (e.g., r_1). The distributor has to choose the regions in which to put each of five possible products. The current product locations are plotted as colored x's. We visualize the current allocation strategy as a state matrix, where blue components denote a given region, product combination has been selected (b). We also show the historical spatial distribution of revenue as a heat map (c). Darker colors indicate more historical revenue. The figure suggests that the current configuration may be sub-optimal.

In addition to commercial applications, successfully learning spatial demand can have impact in society. For example, understanding demand for public transportation can inform placement of bus stops, thereby increasing overall utility of a city. The methods outlined in this work could be extened to other domains.

Some existing work explores domains adjacent to the spatial product allocation problem. A large body of operations research analyzes shelf space distribution. For example, early work proposed a dynamic programming algorithm to discover an optimal shelf allocation strategy [27]. Other work poses shelf space allocation as a constrained optimization problem that can be solved via simulated annealing [4]. More contemporary studies propose frequent pattern mining approaches to determine profitable product item sets [13] [1]. To the best of our knowledge, none of the existing literature has studied the spatial effects of product locations across the entire store.

However, learning a strategy for spatial product allocation is non-trivial. First, the number of candidate allocation strategies is large but the historical data usually only explores a small subset (see Figure 3). Moreover, sales are also correlated with other factors such as holidays and store promotions, which makes the search space even bigger. Because of this issue of data sparsity we cannot directly rely on historical data to learn the best strategy. Second, the cost of experimentation and exploration is high. It is not feasible to perform extensive experiments due to the potential lost revenue and the physical cost of moving products around the store. Finally, the correlation between product positions and sales is likely complex and non-linear due to the dynamic nature of the market; simple search heuristics may not provide an optimal policy. For all of these reasons, we need an approach that allows for experimentation and counterfactual exploration in a cost-effective way.

Therefore, we design a new framework to solve these challenges. We formulate the spatial product allocation problem as a Markov Decision Process (MDP) and propose the Probabilistic Spatial Demand Simulator (`PSD-sim`), which is a stochastic model of spatial demand for physical retail. `PSD-sim` has three model features that facilitate better learning of complex demand patterns: 1) a hierarchical parameter sharing structure; 2) spatial covariance priors to capture spatial demand; 3) product substitution effects learned from online Amazon review data.

We propose `PSD-sim` as a mechanism to study more sophisticated search algorithms such as reinforcement learning without incurring the high cost of exploration in the physical world. We calibrate `PSD-sim` using a real-world dataset of Coca-Cola products. Additionally, when deployed online, `PSD-sim` could be used to perform Monte Carlo rollouts for efficient exploration and experimentation [11]. In our experiments, we demonstrate that `PSD-sim` can effectively recover ground truth test data in two retail environments. Additionally we explore interesting features of the model. Finally, we do a preliminary study into different optimization techniques using `PSD-sim`.

In summary the key contributions of our paper are:

- We study the new problem of spatial product allocation in physical retail
- We propose a novel environment model, which we call the Probabilistic Spatial Demand Simulator (`PSD-sim`) that features hierarchical parameter sharing, spatial covariance priors, and product substitution effects.
- We train PSD-sim on real data from two different retail stores and show it is more effective than existing, discriminitive baselines.
- We do a preliminary study into various optimization methods and show that Deep Q-Learning can learn an effective allocation policy

2 PROBLEM DEFINITION

In the following section, we provide a formal definition of the spatial allocation problem. Additionally, we define the necessary components of our reinforcement learning agent: the state space, action space, reward function, and state transition function.

2.1 Spatial Allocation Problem

In a physical retail environment with a set of n spatial regions, we represent the environment with a spatial graph $\mathcal{G} = (\mathcal{R}, \mathcal{E})$, where each region $r_i \in \mathcal{R}$ is a vertex in the graph, the spatial neighboring relation between two regions r_i and r_j are represented as an edge, $e_{ij} \in \mathcal{E}$. From $\mathcal{G}$, we can construct the adjacency matrix, $\mathbf{A}$. Additionally, we observe a set of k products, $\mathcal{M} = \{m_j : 0 <$

$j <= k\}$ that are sold. For each product, m_j, we know the retail price, p_j.

The decision process faced by the retailer or manufacturer is to allocate each product in $\mathcal{M}$ across regions in $\mathcal{R}$. We define the allocation policy as a function f:

$$f : \mathcal{R} \times \mathcal{M} \rightarrow \mathcal{Z} \tag{1}$$

$$\mathcal{Z} = \{\langle r_i, p_j \rangle, ...\langle r_w, p_q \rangle\} \tag{2}$$

Where $\mathcal{Z}$ is the set of selected product region, such that $w <= n$, $q <= k$ and $\mathcal{Z} \subseteq \mathcal{R} \times \mathcal{M}$. This function is typically dynamic over time, which we denote as f^t. To simplify computation, we treat $\mathcal{Z}^t$ as an $(n \times k)$ grid and refer to it as the board configuration at time, t. An optimal retail strategy is to find the allocation policy that maximizes revenue:

$$f^* = \sum_{t}^{T} \underset{f^t}{\arg\max} \sum_{i,j \in f^t(\mathcal{R},\mathcal{M})} p_j q_i \tag{3}$$

where p_j is the price for product m_j, and q_i is the quantity sold in region r_i and T is the future time horizon of analysis. The main idea of the current work is to discover the long-term, optimal allocation policy, f^* from data.

2.2 Spatial Allocation as a Markov Decision Process

We believe that the spatial allocation problem is well suited for reinforcement learning because the RL agent is designed for sequential decision making that maximizes expected discounted reward over time. We frame the inputs as a Markov Decision Process (MDP). An MDP is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, P, r, \delta \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of possible actions, P is the (typically unkown) state transition function, r is the reward function and $\delta \in [0, 1]$ is the discount factor.

• **State** At each time, t, we observe the state of the retail environment, $\mathcal{G}$. We define the state, $s_t \in \mathcal{S}$, as the tuple of state features, $s_t = \langle \mathcal{Z}^t, d^t, \mathbf{g}^{(t-1)} \rangle$, where $\mathcal{Z}^t$ is the current board configuration, d^t is the current day of the week (e.g., Sunday $\rightarrow$ 0), and $\mathbf{g}^{(t-1)}$ is a vector denoting the revenue at the previous time, $(p_j q_i)^{(t-1)} \forall z \in \mathcal{Z}^t$

• **Action** We define the action space $\mathcal{A} = \mathcal{R} \times \mathcal{M} \times \{-1, 1\} \cup \{0\}$, indicating "to place", "take way" or "do nothing" for each product, m_j in each region, r_i.

• **Reward** The reward function in this case is the total product revenue at time t, constrained by the monetary cost, c, of placing a set of products in each region:

$$r(t) = \sum_{i=1}^{n} \sum_{j=1}^{k} p_j q_{ij}^t - c \sum_{i=1}^{n} \mathbb{1}_{\mathcal{Z}}(r_i) \tag{4}$$

The second term in the reward function accounts for the cost faced by the manufacturer who typically has to pay for each space in a retail environment.

• **State transition function**: The state transition, P is defined as $p(s^{t+1}|s^t, a^t) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, which gives the probability of moving to state, $s^{(t+1)}$ given the current state and action. In the spatial allocation problem the exact transition function, P, is unknown since the current state, s^t depends on the results of the previous time, $\mathbf{g}^{(t-1)}$. We model this transition as a stochastic process.

3 PROPOSED METHOD

In this section, we define our framework for solving the spatial allocation problem. We first outline our proposed environment model that is used to simulate spatial demand. We propose two variants of the environment model. The first assumes that spatial weights are homogeneous across products. The second allows for heterogeneous spatial weights and parameter sharing across regions.

3.1 Stochastic Model of Spatial Demand

We propose the following stochastic model of spatial demand in physical retail, which we call the Probabilistic Spatial Demand Simulator (`PSD-sim`). See Figure 2 for an overview. In the current work, the stochastic model is used as a 'simulator' to enable offline policy learning. There are many advantages of using a probabilistic model in the optimal product allocation problem. First, it allows us to exploit structure in our dataset. We are able to share parameters across spatial groups for added data efficiency. Second, we are able to incorporate prior knowledge about the data generating process, as well as prior knowledge from secondary data sets. Third, it provides a natural framework for simulating future scenarios through Monte Carlo roll-outs.

Our ultimate objective is to maximize total revenue at time, $\rho^{(t)}$, which is defined as $\rho^{(t)} = \sum_{i=1}^{n} \rho_i^{(t)}$, where $\rho_i^{(t)}$ is the revenue for region, r_i. Region-level revenue is calculated over products, m_j: $\rho_i^{(t)} = \sum_{j=1}^{k} p_j q_{ij}^{(t)}$.

The key variable of interest is, $q_{ij}^{(t)}$, the quantity sold for product, m_j, region, r_i, at time, t. We model $q_{ij}^{(t)}$ as a truncated normal random variable, $q_{ij}^{(t)} \sim \psi(\mu, \sigma, a, b)$

where, $\psi(\mu, \sigma, a, b)$ is the pdf of the truncated normal distribution:

$$\psi(x; \mu_q, \sigma_q, a, b) = \begin{cases} 0 & \text{if} x \leq a \\ \frac{\phi(\mu, \sigma^2; x)}{\Phi(\mu, \sigma^2; b) - \Phi(\mu, \sigma^2; a)} & \text{if} a < x < b \\ 0 & \text{if} b \leq x \end{cases} \tag{5}$$

The term, $\phi(z)$ is the standard normal pdf, and $\Phi(z)$ is its cumulative distribution function:

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z}{2}^2} \tag{6}$$

$$\Phi(z) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{z}{\sqrt{(2}}\right)\right) \tag{7}$$

See [5] for more details. We set $a = 0$ and $b = +\infty$, which forces $\Phi(\mu, \sigma^2; b) = 1$ and constrains the quantity, $q_{ij}^{(t)} \in \mathbb{R}^+$. The prior for $q_{ij}^{(t)}$ is characterized by the mean, μ_q, which is a linear function of environment features, $\mathbf{x}$ and learned weights, $\mathbf{w}$, and the inverse gamma distribution for the variance, σ_q:

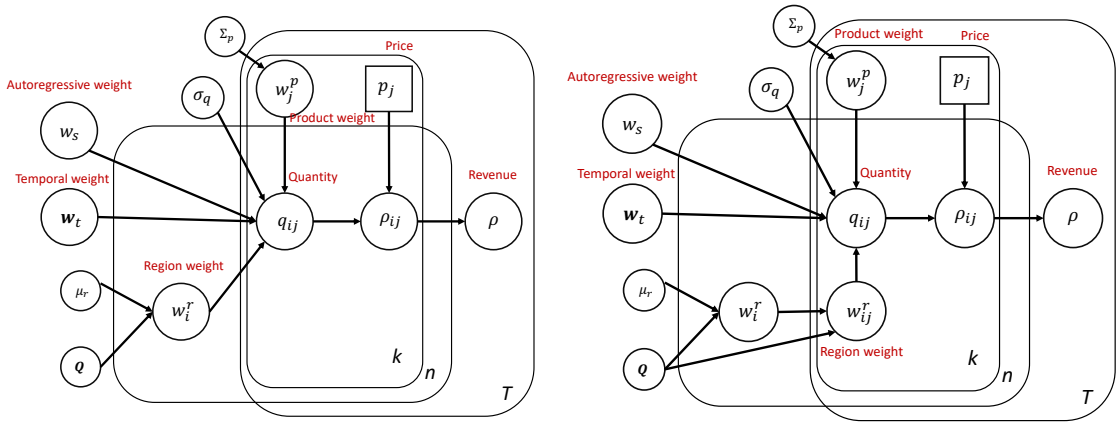


Figure 2: An overview of both `PSD-sim-linear` (left) and `PSD-sim` (right) each as a Bayesian network. The boxes are "plates" representing structures in the data. The plates marked by k, n and T represent products, regions, and time, respectively. Circles denote random variables and squares are deterministic quantities. Both models decompose quantity as a function of region, product, time, and auto-regressive weights. The two differ in the structure of the region-level weights, $\mathbf{w}_r$. The `PSD-sim-linear` assumes a single weight vector for all regions, while `PSD-sim` has a hierarchical structure that also allows region-level weights to vary by product.

$$\mu_q = \mathbf{x}^\top \mathbf{w} + b,\ \sigma_q \sim \text{IG}(\alpha_q, \beta_q) \tag{8}$$

In our environment, we observe temporal features, $\mathbf{x}_t$, region features, $\mathbf{x}_r$, product features, $\mathbf{x}_p$, and autoregressive features, $\mathbf{x}_s$: $\mathbf{x} = [\mathbf{x}_t, \mathbf{x}_r, \mathbf{x}_p, \mathbf{x}_s]^\top$. We discuss our feature extraction approach more in Section 3.5

3.1.1 Region-level Weights. We initially model the weights for each spatial region with a multivariate normal distribution, with mean vector, $\boldsymbol{\mu}_r$ and precision matrix, $\mathbf{Q}_r$:

$$\mathbf{w}_r \sim \mathcal{N}(\boldsymbol{\mu}_r, \mathbf{Q}_r) \tag{9}$$

$$\mathbf{Q}_r = \gamma \tilde{L} = \gamma(\mathbf{I} - \hat{\mathbf{D}}^{-\frac{1}{2}} \hat{\mathbf{A}} \hat{\mathbf{D}}^{-\frac{1}{2}}) \tag{10}$$

We are able to encode graph information into our prior for $\mathbf{w}_r$ by computing the adjacency matrix, $\mathbf{A}$ from $\mathcal{G}$ and computing the normalized graph Laplacian, $\tilde{L}$. Doing so allows us to put a spatial prior over the variance of the region weights. We use this as our precision matrix [7] times a scale factor, γ. The normalized graph Laplacian ensures that the precision matrix, $\mathbf{Q}$ is symmetric and positive semidefinite. We treat the mean vector, $\boldsymbol{\mu}_r$ as a hyperparameter.

3.1.2 Product-level Weights. We also define weights for each product, m_j, as follows:

$$\mathbf{w}_p \sim \mathcal{N}(\boldsymbol{\mu}_p, \Sigma_p) \tag{11}$$

We assume the product weights have mean, $\boldsymbol{\mu}_p$ with a structured covariance prior Σ_p that models product substitution effects. Substitution effects are discussed more in Section 3.3.

3.1.3 Temporal weights. The temporal features capture the long-term and short-term seasonality of the environment. The temporal weights are defined similar to the product weights. Namely, the temporal weights, $\mathbf{w}_t$, follow a multivariate normal distribution:

$$\mathbf{w}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t),\ \boldsymbol{\mu}_t \sim \mathcal{N}(\boldsymbol{\delta}_t, \Gamma_t),\ \Sigma_t = \mathbf{L}\mathbf{L}^\top \sim \text{LKJ}(\sigma_t) \tag{12}$$

We put an LKJ prior over the covariance matrix and reparameterize Σ_t as its cholesky decomposition, $\mathbf{L}\mathbf{L}^\top$, so that the underlying correlation matrices follows an LKJ distribution [15]. The standard deviations, σ_p, follow a half-cauchy distribution. The advantage of the LKJ prior is that it is more computationally tractable than other covariance priors [15].

3.1.4 Autoregressive weight. Finally, we specify the weight of previously observed revenue values on $q_{ij}^{(t)}$. The feature, $\mathbf{x}_s$ is an autoregressive feature denoting the previous k values of product-level revenue, $\rho_j^t = \sum_{j=i}^{n} p_j q_{ij}^{(t)}$. We assume truncated normal prior for w_s, and half cauchy priors for the location, μ_s and scale, σ_s:

$$w_s \sim \psi(\mu_s, \sigma_s, a, b) \tag{13}$$

$$\mu_s \sim \text{HalfCauchy}(\phi_s) \tag{14}$$

$$\sigma_s \sim \text{HalfCauchy}(\psi_s) \tag{15}$$

We again set $a = 0$ and $b = +\infty$ such that $\mathbf{w}_s \in \mathbb{R}^+$.

3.2 Hierarchical Model of Spatial Demand

One major advantage of Bayesian modeling is the natural ability to exploit the structure inherent in many datasets for parameter sharing [2]. Sharing parameters across groups in a hierarchical way, can enable better learning and more predictive power. It can also allow for discovery of more fine-grained spatial effects of the environment. Below we present a hierarchical variant of the environment model presented in Section 3.1.

3.2.1 Hierarchical Region-level Weights. The key difference between the hierarchical environment model and the one presented above lies in the definition of $\mathbf{w}_r$ in Equation 9. Previously, we assumed that the spatial effects of each region on quantity demanded, $q_{ij}^{(t)}$, was homogeneous across products. The weight for product $m_{j=1}$ and $m_{j=2}$ are the same, given they are both placed in region, r_i.

We also test the hypothesis that this homogeneity assumption is too strong in Section 4. We allow for heterogeneous region weights, w_{ij}^r, which denotes the impact of region, r_i, given product, m_j, on $q_{ij}^{(t)}$.

$$\mathbf{w}_{ij}^r \sim \mathcal{N}(\mathbf{w}_r, \mathbf{Q}_r),\ \mathbf{w}_r \sim \mathcal{N}(\boldsymbol{\mu}_i, \mathbf{Q}_r) \tag{16}$$

Note that both $\mathbf{w}_r$ and $\mathbf{w}_{ij}^r$ share the same same covariance structure: the precision matrix defined by the graph laplacian (Equation 10). Thus, the region weights are only hierarchical in their means. Additionally, we treat the upper-level mean vector, $\boldsymbol{\mu}_r$ as a hyperparameter. In Section 4 we test which environment model is more effective at predicting revenue on a test set.

3.3 Learning Product Substitution Effects

One important challenge in the allocation problem is accurately capturing product relationships, or substitution effects. Both theoretical and empirical economics demonstrates that products tend to exhibit either complementary or supplementary effects [18, 22]. In the case of complements, two goods "go together" if the presence of one increases the demand for the second. For supplements, the presence of one good decreases the demand for the other.

In order to properly learn the patterns of product substitution, an ideal data set would consist of product co-occurences. Intuitively, products that frequently co-occur in purchase data are likely to be complements (e.g., peanut butter and jelly). However, many common retail datasets, like the one described in section 4.1.1, are aggregated and do not reveal individual transactions. Consequently, learning substitution effects from aggregated data can be challenging.

To solve this problem, we propose to incorporate information from both offline retail data, and publicly available online review data. From the online data we observe product reviews within a user. We use this information to learn general product substitution effects, which is then used to seed the covariance matrix for the product weights, Σ_p. Given a set of of users $\mathcal{U} = \{u_1, ...u_U\}$, we define our measure of product correlation:

$$\alpha_{ij} = \frac{\sum_{u=1}^{U} s_i s_j \mathbb{1}_u(m_i, m_j)}{\sqrt{\sum_{i=1}^{k} s_i^2}\sqrt{\sum_{i=j}^{k} s_j^2}} \tag{17}$$

where $\mathbb{1}_u(m_i, m_j)$ is the indicator that m_i and m_j are both reviewed by user u, s_i and s_j are the review scores for m_i and m_j from user u. If two products, m_i and m_j are both reviewed by user u, then we multiply the review scores, s_i and s_j together to get a weighted co-occurence. We normalize the scores so that each score falls between 0 and 1. The normalization constant is the product of the square root of the sum of rating for m_i and m_j.

Intuitively, when two items have high scores and co-occur often, α_{ij} approaches 1. Conversely, for two products that never co-occur $\alpha_{ij} = 0$. We then scale α_{ij} by λ to give a more meaningful covariance and use σ_{ij} as the off-diagonal elements of the product covariance matrix Σ_p in Equation 11.

$$\sigma_{ij}^p = \lambda \alpha_{ij} \tag{18}$$

This procedure gives larger, positive covariance to products that co-occur frequently and have high reviews in the Amazon dataset. These will be considered complements in our environment model `PSD-sim`. Moreover, products that are complements have larger, positive impact on demand q_{ij}.

3.4 Training

We train `PSD-sim` using the No U-Turn Sampler (NUTS) algorithm [6]. This allows us to draw samples from the posterior distribution of model weights, $\mathbf{W}$, as well as the posterior predictive distribution of quantity, $q_{ij}^{(t)}$, and revenue $\rho^{(t)}$. We use Automatic Differention Variational Inference (ADVI) [9] as an initialization point for the sampling procedure. All models are implemented in PyMC3 [24]

We initialize with ADVI using 200,000 iterations. Once initialized we sample the posterior using NUTS using a tuning period of 5,000 draws followed by 5,000 samples across four chains.

3.5 Feature Extraction

In order to train `PSD-Sim`, we extract environment-level features, $\mathbf{x}$, which is composed of temporal features, $\mathbf{x}_t$, region features, $\mathbf{x}_r$, product features, $\mathbf{x}_p$, previous sales features and $\mathbf{x}_s$.

• **Temporal features.** We use a one-hot vector denoting the day of the week for, $\mathbf{x}_t$. This feature vector captures the short-term temporality commmon in physical retail settings. For example, weekends tend to be busier shopping days than weekdays.

• **Region features.** We again use a one-hot vector for spatial regions, $\mathbf{x}_r$. This feature vector 'turns on' the weight that each region has on quantity via the weight vector, $\mathbf{w}_r$.

• **Product features.** We use a one-hot vector to represent each product, $\mathbf{x}_p$.

• **Previous sales features.** Finally, we construct an autoregressive sales feature that represents the sales at time, $t-1$. We use the previous sales for product m_j, summed across all regions, $w_s = \rho_j^{(t-1)} = \sum_{i=1}^{k} p_j q_{ij}^{(t-1)}$. This feature captures micro-fluctuations in demand for each product.

4 EXPERIMENTS

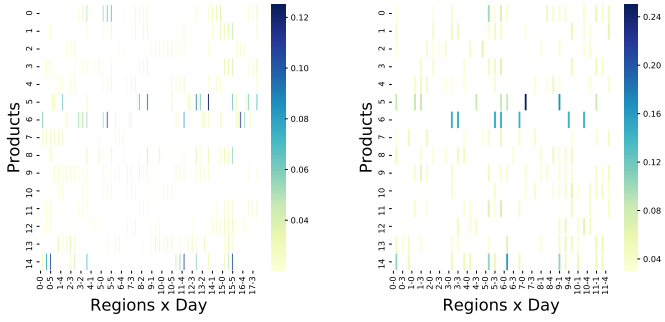
In the following section we first describe the dataset and discuss interesting features of the problem. Next, we perform empirical evaluations of `PSD-sim` across two large retail environments by showing that it can more accurately predict revenue better than more elementary baselines. We explore the model by discussing the estimation of region weights, and show that it is robust to previously unseen states. Finally, we do a preliminary inquiry into effective methods for optimization.

4.1 Dataset Decription

In the following section we describe in detail our offline retail dataset, and the online, product review dataset.

Table 1: Retail Dataset Summary

Store Id	Regions	Products	Train	Test
#1	17	15	8/2017 - 3/2019	4/2019 - 8/2019
#2	12	15	8/2017 - 3/2019	4/2019 - 8/2019



(a) Store 1 (b) Store 2

Figure 3: Spatiotemporal frequency table for store 1 (a) and store 2 (b). For each product, region, day combination we count the number of samples observed in the dataset. The rows represent products, and the columns are region, day pairs. The counts are normalized for each product. The heatmaps above shed light into the size of the allocation strategy space, and its coverage in the training data. We can see that the number of candidate allocation strategies is large but the historical data only explores a small subset. Thus, we cannot directly rely on historical data to learn the best strategy.

4.1.1 Retail Data. Our retail dataset is comprised of three primary entities: stores, products, and regions.

• **Stores**: We collect data from Swire Coca-Cola, a large Coca-Cola distributor in the western United States. The data is comprised of two large supermarket and retail stores in Salt Lake City, UT, USA. Each store primarily sells groceries, household goods and clothing.

• **Products**: We observe quantities sold for a set of 15 products, as well as each product's average price over the year. All of the products in our dataset are Coca-cola brand beverage products. The product set includes items such as "Coca-Cola 8 oz - 12 pack", and "Sprite 2 Liter bottle". See Figure 3 for a visualization of the spatiotemporal frequency table of regions and products.

• **Regions:** The data provides daily counts of quantities at the region-product level. Additionally, the locations of the products are varied in product "displays". These displays are small groups of products intended to catch the eye of the shopper. See Figure 1 for an example of a product display layout. Store 1 is comprised of 17 regions, and store 2 has 12. Each region represents a section of the store. In general regions tend to be constructed based on the function of each space (e.g., pharmacy, deli, etc.). We construct a spatial graph of these regions.

4.1.2 Online Review Data. In addition to the retail dataset, we use the publicly available Amazon 18 dataset [20] to learn product substitution effects and structure our covariance matrix. The

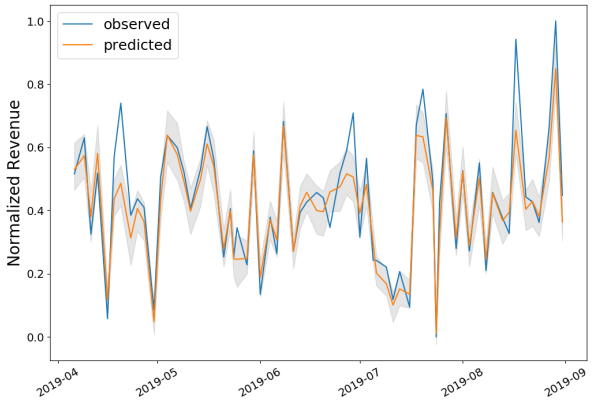


Figure 4: Predictions and observed revenue during the test period (April - August 2019). Revenue from store 1 is aggregated to the store-level. We show the posterior distribution for revenue by plotting the mean (blue line) and inner 95% credible interval (gray shaded area). In general, the predicted revenue mirrors the behavior of the ground truth data. `PSD-sim` correctly predicts directional changes (i.e., positive or negative) 82% of the time.

Amazon 18 dataset consists of timestamped user ratings of items, in addition to other rich metadata. In raw form it is very large and has over 233.1M samples across all categories and is typically used for recommendation research. Because all of the products in the retail data are Coca-Cola products, we filter the dataset to three categories for our experiments: grocery and gourmet food, prime pantry, and home and kitchen. We then match the universal product code (UPC) of the 15 products in the retail dataset to the Amazon Standard Identification Number (ASIN) and filter again to the user-item interactions involving only the retail product set. This allows us to compute pair-wise item similarities from ratings data with equation (17).

4.2 Model Evaluation

We first evaluate the effectiveness of `PSD-sim` in predicting revenue on a test dataset. Specifically, we partition the time series into a training period from August 1, 2017 - March 31, 2019 , and a test period of April 1, 2019 to August 31, 2019. We compare the proposed `PSD-sim` to a variety of discriminitive baselines and evaluate all models in terms of the following error metrics:

$$\text{MSE} = \frac{1}{nkT}\sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{k}(\rho_{ij}^{(t)} - \hat{\rho}_{ij}^{(t)})^2 \tag{19}$$

$$\text{MAE} = \frac{1}{nkT}\sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{i=j}^{k}|\rho_{ij}^{(t)} - \hat{\rho}_{ij}^{(t)}| \tag{20}$$

$$\text{MAPE} = \frac{1}{nkT}\sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{i=j}^{k}\left|\frac{\rho_{ij}^{(t)} - \hat{\rho}_{ij}^{(t)}}{\rho_{ij}^{(t)}}\right| \tag{21}$$

where the predicted revenue is equal to quantity times price for the i^{th} product, in the j^{th} region, at time, t: $\hat{\rho}_{ij}^{(t)} = \hat{q}_{ij}^{(t)} p_j$. To

compare PSD-sim to the discriminitive models, we obtain a point estimate for $\hat{q}_{ij}^{(t)}$ by computing the mean of the samples taken from posterior predictive distribution.

Table 2: Evaluation of PSD-Sim. The best scores are presented in bold, and the best two for each metric are underlined.

	Store 1			Store 2		
Model	MSE	MAE	MAPE	MSE	MAE	MAPE
OLS	1663.33	29.10	0.3932	2955.76	33.16	0.3546
KNN	1287.78	26.53	0.3709	2884.82	33.32	0.3361
SVR	1519.10	29.00	0.3637	3107.20	35.97	0.3446
RF	**1152.65**	24.33	0.3265	2631.41	31.06	0.2986
GBRT	1383.54	27.24	0.3641	2620.85	31.66	0.3133
MLP	1193.32	25.10	0.3350	2628.38	31.60	0.2962
PSD-sim	1170.68	**19.39**	**0.3218**	**1331.30**	**19.92**	**0.2862**

4.2.1 Baseline Approaches. The proposed PSD-sim is a generative environment model and is able to draw samples from the full posterior distribution of revenue, $\rho^{(t)}$. We also compare to the following discriminative prediction models [23]:

- **Ordinary Least Squares (OLS)**: Classical least squares regression that decomposes predicted quantity as a linear function of weights: $\hat{q}_{ij}^{(t)} = \mathbf{X}\mathbf{w} + b$.
- **K-nearest Neighbors (KNN)**: an instance-based learning method than finds the closest k neighbors in the feature space and predicts the label as the average of the nieghbor labels [25]
- **Support Vector Regression (SVR)**: Support Vector Machines extended to regression. We select an RBF kernel [8].
- **Random Forest (RF)**: An ensemble regressor [25] that learns many decisions trees and averages over the labels in each terminal node to compute, $\hat{q}_{ij}^{(t)}$. We use 100 trees.
- **Gradient Boosted Regression Trees (GBRT)**: An ensemble of regression trees recursively trained on sample residuals [25].
- **Multilayer Perceptron (MLP)**: A simple neural network with two hidden layers of dimensions 256, and 128 with ReLU activations, MSE loss, and the Adam optimizer [14].

We use the same features for all baselines. The features used in the experiment are described in Section 3.5.

4.2.2 Results. We report the results in Table 2. Additionally, predicted and actual revenue are plotted in Figure 4. The proposed method, PSD-sim, is overall more accurate at predicting future states than baselines. We observe that error is minimized across all metrics in store 2. In the case of store 1 the proposed model offers superior performance under MAE and MAPE. In particular, the reduction in MAE offered by PSD-sim is significant. In store 1, we see a 20% decrease in MAE from the next best model; for store 2 we similarly see a 36% decrease. Additionally, in both stores the average error falls within approximately $19 of the true value. It appears that a combination of the informative priors from spatial and product similarities, in addition to the hierarchical parameter sharing structure, allow the model to better learn the underlying demand.

Table 3: Ablation Study. The MSE is reported for both stores using different model configurations. The goal is to isolate the impact that each modeling choice has on performance. In general, we observe that the hierarchical model component has a large effect on model performance.

Model	Spatial	Substitution	Store 1	Store 2
Linear	✗	✗	1293.03	1399.97
Linear	✗	✓	1294.45	1398.07
Linear	✓	✓	1291.77	1396.52
Hierarchical	✗	✗	1173.50	1335.88
Hierarchical	✗	✓	1171.14	1333.88
Hierarchical	✓	✓	**1170.68**	**1331.30**

In the case of Store 1, Random Forest does appear to offer slight improvement over PSD-sim under MSE. However, the MAE of PSD-sim is significantly lower than that of the RF. This seems to suggest PSD-sim makes a handful of larger errors that are penalized harshly by the squared error term.

Overall, PSD-sim achieves top performance in nearly all cases, and top two in all settings. This is likely due to that fact that it is designed specifically for the task of spatial demand prediction and is therefore more effective than more general regression methods.

4.3 Ablation Study

In the following section we analyze PSD-sim to diagnose the effect of each modeling choice on test performance. In our ablation study, we isolate the hierarchical component, the spatial component and the substitution effects (e.g., product covariance) and observe changes in test performance. Our results are reported in Table 3.

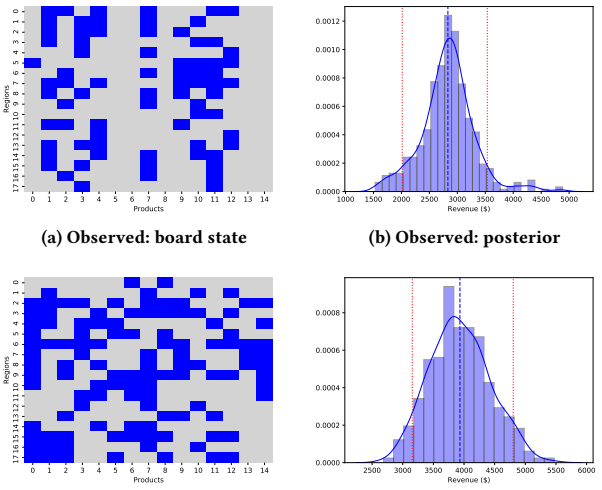
We note that the inclusion of the hierarchical component gives the biggest boost to test performance. The reduction in error due to the hierarchical model is between 119 an 123 for store 1, and 64-65 for store 2. Intuitively, it makes sense that different products have varying effects on demand in different regions. Such heterogeneity captures interactions between products and other items in their surroundings. Additionally, store 1 has more regions than store 2. It's likely that as the number of regions increases, the hierarchical model can better identify more granular signal in the data.

It is also interesting to note that the inclusion of the substitution and spatial effects do have positive effects on performance in both store 1 and store 2. However, these effects are more incremental than the hierarchical model component. In sum, we consistently see the best performance when hierarchical, substitution, and spatial components are all used.

4.4 Environment Model Analysis

Next we provide a discussion of key features of the environment that PSD-sim has learned. First we show a heatmap of the posterior means of the hierarchical region weights learned in each store. We then show that PSD-sim is robust to *new* states that it did not see in the training process.

4.4.1 Analysis of region weights. Figure 4 also contains a heatmap of the observed spatial distribution of revenue in both stores. This



(a) Observed: board state (b) Observed: posterior

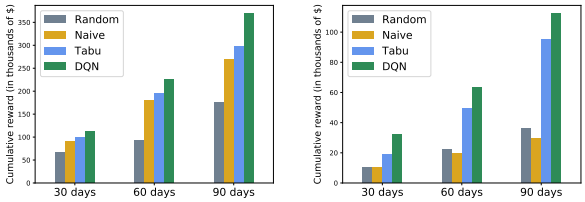
(c) Unseen: board state (d) Unseen: posterior

Figure 5: Two possible board state configurations and their corresponding posterior predictive distributions. The means are plotted as blue dotted lines and the inner 95% credible interval as red dotted lines. Along the top (a, b) row we show a state that was observed in the training data. The bottom row contains a randomly generated *unseen* board configuration (c, d). In both cases `PSD-sim` yields a posterior distribution that is approximately Gaussian. The model behaves similarly for both seen and unseen states. This demonstrates `PSD-sim` is robust to unseen states and is useful for simulation.

figure suggests clear patterns related to how products are allocated throughout the store in the observed data. We can see some similarities between the summary matrices of (a) and (c), and the learned weights of (b) and (d). For instance, within store 1 both the raw data (a) and estimated weights (b), predict a high expected revenue for placing product 5 in region 12. Additionally, in store 2, placing product 10 in region 3 yields a higher expected revenue in the summary matrix (c) and the model environment (d). However, the learned weights from the model and those from raw data are not identical because `PSD-sim` also accounts for other factors such as temporal effects.

4.4.2 Seen and unseen states. Because `PSD-sim` is an environment model intended for efficient training or exploration of a variety of search methods, it is important to demonstrate that it is robust to unseen states. The dataset used to train `PSD-sim` consists of two stores with daily snapshots over a period of more than a year. For just the board configuration, $\mathcal{Z}^t$ alone, there are $2^{r \times p}$ possible states. In the case of store 1, this corresponds to $2^{r \times p} \approx 5 \times 10^{76}$ possible states, and $2^{r \times p} \approx 1.6 \times 10^{63}$ for store 2. There is no feasible way that in a year-and-a-half period we could observe all possible board states.

Therefore, we need to demonstrate that `PSD-sim` can generalize to new, unseen states. In Figure 5 we visualize two possible board



(a) Store 1 (b) Store 2

Figure 6: The cumulative reward of three search algorithms across store 1 and store 2 (in thousands of dollars). We vary the episode length in 30 day increments (i.e., 30, 60, and 90 days in the future). The DQN algorithm is superior in all cases. Additionally, we observe that as the episode length increases so does the relative effectiveness of the DQN. The DQN agent excels in the longer episode settings because it is able to learn important, longer term strategies. On average, DQN offers an improvement of 27% over Tabu search in terms of cumulative test reward.

configurations, $\mathcal{Z}^t$, with the corresponding model output: the posterior distribution of revenue at time t, $\hat{\rho}^{(t)}$. On the top row are the board state and posterior of an example board configuration observed in our training data (a), along with the predicted posterior distribution (b) of revenue. Along the bottom row, are the same figures for a randomly generated, unseen state (c) and its predicted posterior (d). Both states have the exact same number of product-region pairs and all other state features are held constant (e.g., store, day, previous sales, etc...).

In Figure 5(b) and 5(d) we also plot the mean (dotted blue line) and 95% credible interval (dotted red line), to characterize the shape of the distributions. For both the seen and unseen states, `PSD-sim` yields a posterior predictive distribution that is well-behaved and nearly Gaussian. The posterior mean of the observed state is approximately, $3,000 and the mean of the unobserved state is just below $4,000. The mean from the unobserved state is likely higher due to the selection of better product-region pairs. Notwithstanding, the shapes of the two distributions are quite similar. The width of the 95% credible interval is approximately $1,500 for both the seen and unseen board configurations. The observation that the posteriors for both states are very similar suggests that `PSD-sim` is robust to new, unobserved states and is therefore a useful generative model for simulation.

4.5 Optimization Techniques

In this section we perform a preliminary study into various search algorithms to solve the optimal product allocation problem with `PSD-sim` as a simulator. Because exploration and experimentation in the physical world is costly, it is often preferable to design an agent that can learn a policy offline before deploying into the online environment [11].

In our experiments, the authority of the algorithm is to allocate a set of products (e.g., Coca-Cola 12 oz) within a category (e.g., beverage). Used in this way, the search algorithm will select high-reward locations for the target products. Other products may also be in a given location, but such product selections are made independent

of the algorithm. We have designed the experiment in this way after consulting with industry experts at Coca-Cola. In this paper, there are 15 products in total, but our approach can be easily be extended to a larger set of products.

4.5.1 Search Algorithms. To this end we compare four methods to search the problem space: **1) Random Search:** A search algorithm that relies on a totally random policy: at each time step, t choose a random action; **2) Naive Search:** The naive strategy in this case is simply "do nothing." At each time step, we do not move any products and do not deviate from the initialized allocation policy. This baseline allows us to assess whether searching and exploration is useful at all; **3) Tabu Search:** A local neighborhood search algorithm that maintains a memory structure called a "Tabu" list. The "Tabu" list is comprised of recent actions to encourage exploration and avoid getting trapped in local maxima. We implement the Tabu algorithm with a "Tabu" list of the previous 50 actions. We treat the local neighborhood search as the enumeration over the set of feasible actions given the current state, s_t; **4) Deep Q-Learning (DQN):** A reinforcement learning algorithm that utilizes a neural network to approximate the state-action function, $Q(s, a)$. The DQN typically employs an ϵ-greedy strategy for exploration. The exploration probability, ϵ, is typically annealed throughout training. We train our DQN using 50,000 training iterations prior to the test period. Note that this approach can scale to a much larger set of products with minor modifications to the Q-network architecture [16].

4.5.2 Policy Evaluation. In this section we conduct a policy evaluation experiment. We randomly fix the initial environment state and allow each of the search algorithms listed above to interact with the environment according to its corresponding strategy in a test period of one episode. The state in store 1 is initialized with 96 product-region pairs, while the state in store 2 has 30. We record the total reward accumulated by each agent during the entire episode. For each store, we vary the episode length in 30 day increments: 30, 60, and 90 days in the future. This allows us to evaluate whether longer rollouts have an effect on the policy of each agent. The results of the policy evaluation experiment are reported in table 6.

In general, we see that DQN is the most effective search algorithm in both stores, and across all three episode settings. In each case, it accumulates the most total reward in the test episode. On average, DQN is 27.3% better than Tabu, in terms of cumulative test reward. Tabu is the second most effective search strategy, beating out the random and naive search heuristics in all cases. Interestingly, the naive search baseline of "do nothing" is more effective than random searching in store 1, but not in store 2.

Additionally, it appears that as the episode length is increased, so too does the relative effectiveness of DQN as compared to Tabu. In the store 1, 30 day episode setting, DQN exceeds Tabu by $11k. This difference increases to $30k for 60 days and $71k for 90 days. In store 2 we see a similar effect. The difference between DQN and Tabu increases from $13k to $13.5k to $17k in the 30, 60, and 90 day settings respectively. Not only is DQN more effective, but its performance relative to other baselines gets better with longer episodes.

DQN excels as episode length increases in large part because the underlying Q-function is an approximation of discounted, expected reward over time. This allows the agent to potentially think multiple steps ahead and take a set of actions that yield low immediate reward, but higher reward in later steps. Conversely, the random and Tabu search baselines are short-term or greedy search algorithms. Especially in the case of Tabu; at each time step, an action is solely selected based on what will maximize short-term reward. These results suggest that the correlations between spatial allocation and sales is complex and dynamic. Thus both of the two baselines achieve sub-optimal policies.

It is also interesting to note the behavior of the naive search compared to the random strategies across the two stores. In store 1, the environment is initialized with an allocation strategy that already has many product placements (96). We see that the naive strategy is a strong baseline, and is superior to the random policy in each of the 30, 60 and 90 day settings. However, in store 2 where the initial allocation is more sparse (30 placements), the random policy is better than or equal to the naive search. This suggest that as more products are placed it is more difficult to find incremental improvements in the allocation strategy.

5 RELATED WORK

There are two major streams of literature that intersect with our problem: 1) shelf space allocation and 2) spatial demand estimation.

Shelf Space Allocation: Some classical work approaches the shelf space allocation problem by proposing a dynamic programming algorithm to allocate limited shelf space among a finite set of products [27]. Later work proposed a simulated annealing optimization approach that accounts for two primary decisions variables: product assortment and allocated space for each product [4]. Additionally, Murray et al. propose an optimization algorithm whose decision variables are product location and orientation within a shelf [21]. More recently, frequent pattern mining algorithms have been proposed to allocate product shelf space. Brijs et al. [13] propose the PROFSET algorithm, which is an association rule algorithm that mines customer basket sets to identify profitable product pairings. Aloysius and Binu propose a PrefixSpan algorithm for shelf allocation that first identifies complementary categories from historical purchase data before identifying product mix strategies within categories [1]. These existing studies all focus on micro-regions (shelves) within the retail environment rather than macro-level patterns across the store.

Spatial Demand Estimation: Recent studies have proposed RL algorithms as a mechanism for spatiotemporal demand estimation. Significant attention has been paid to the order dispatching problem in ride sharing systems. For example, Lin et al. [17] tackle the dispatch problem by proposing a contextual multi-agent reinforcement learning framework that coordinates strategies among a large number of agents to improve driver allocation in physical space. Additionally, Li et al. [12] also approach the order dispatching problem with multi-agent reinforcement learning (MARL).

Recent effort to optimize traffic control systems via reinforcement learning has shown encouraging results. These systems seek to adjust traffic lights to real-time fluctuations in traffic volume and road demand. Wei et al [26] propose IntelliLight, which is a

phase-gated deep neural network that approximates state-action values. More recently [10] proposes a graph attentional network to facilitate cooperation between many traffic signals.

While these reinforcement learning methods deal with the large-scale optimization of spatial resource, they cannot be directly applied to the product allocation problem because they all rely on domain-specific simulators. We propose `PSD-sim` in an effort to extend these state-of-the-art optimization techniques to our problem.

6 CONCLUSION

In this paper, we proposed a new problem called spatial product allocation in physical retail. The problem is motivated by the fact that well placed products can maximize impulse buys and minimize search costs for consumers. To solve this problem we propose a probabilistic environment model called `PSD-sim` that allows for search, simulation and exploration of new product allocation strategies. We calibrate `PSD-sim` on real data collected from two large retail environments. We show that `PSD-sim` can make both accurate predictions on test data and that it is robust to unseen states. Additionally, we do a preliminary study into various optimization methods using `PSD-sim`. We discover that Deep Q-learning techniques can learn a more effective policy than baselines. On average, DQN offers an improvement of 27% over Tabu search in terms of cumulative test reward.

REFERENCES

[1] George Aloysius and D. Binu. 2011. An approach to products placement in supermarkets using PrefixSpan algorithm. *Journal of King Saud University - Computer and Information Sciences* (2011).

[2] Hal S. Stern David B. Dunson Aki Vehtari Andrew Gelman, John B. Carlin and Donald B. Rubin. 2013. *Bayesian Data Analysis: Third Edition.* CRC Press, Boca Raton, FL.

[3] Anant Badgaiyan and Anshul Verma. 2015. Does urge to buy impulsively differ from impulsive buying behaviour? Assessing the impact of situational factors. *Journal of Retailing and Consumer Services* (2015).

[4] Norm Borin, Paul W. Farris, and James R. Freeland. 1994. A model for determining retail product category assortment and shelf space allocation. *Decision Sciences* (1994).

[5] John Burkardt. 2014. The Truncated Normal Distribution. (2014). https://people.sc.fsu.edu/~jburkardt/presentations/truncated_normal.pdf

[6] Hoffman Matthew D. and Andrew Gelman. 2011. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research* (2011).

[7] Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst. 2016. Learning Laplacian Matrix in Smooth Graph Signal Representations. In *IEEE Transactions on Signal Processing.*

[8] Harris Drucker, Christopher J. C. Burges, Linda Kaufman, Alex J. Smola, and Vladimir Vapnik. 1997. Support Vector Regression Machines. In *Advances in Neural Information Processing Systems 9,* M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.). MIT Press, 155–161.

[9] Alp Kucukelbir et al. 2017. Automatic Differentiation Variational Inference. *Journal of Machine Learning Research* (2017).

[10] Hua Wei et al. 2019. CoLight: Learning Network-level Cooperation for Traffic Signal Control. In *in Proceedings of the 2019 ACM on Conference on Information and Knowledge Management (CIKM'19).*

[11] Lukasz Kaiser et al. 2019. Model-Based Reinforcement Learning for Atari. In *arXiv.* https://arxiv.org/pdf/1903.00374.pdf

[12] Minne Li et al. 2019. Efficient Ridesharing Order Dispatching with Mean Field Multi-Agent Reinforcement Learning. In *Proceedings of the World Wide Web Conference, TheWebConf'19.*

[13] Tom Brijs et al. 2001. A Data Mining Framework for Optimal Product Selection in Retail Supermarket Data: The Generalized PROFSET Model. In *arXiv.* https://arxiv.org/pdf/cs/0112013.pdf

[14] Diederik P. Kingma and Jimmy Lei Ba. 2014. Adam: A method for stochastic optimization. In *International Conference of Learning Representations.*

[15] Daniel Lewandowski, Dorota Kurowicka, and Harry Joe. 2009. Generating random correlation matrices based on vines and extended onion method. *Journal of Multivariate Analysis* (2009).

[16] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. (2015). arXiv:cs.LG/1509.02971

[17] Kaixing Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. 2018. Effecient large-scale Fleet Management via Multi-agent Deep Reinforcement Learning. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, KDD'18.*

[18] Andreu Mas-Colell and Michael D. Whinston. 1995. *Microeconomic Theory.* Oxford University Press.

[19] Anna Mattila and Jochen Wirtz. 2008. The role of store environmental stimulation and social factors on impulse purchasing. *Journal of Services Marketing* (2008).

[20] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based Recommendation on Styles and Substitutes. In *SIGIR.*

[21] Chase C. Murray, Debabrata Talukdar, and Abhijit Gosavi. 2010. Joint Optimization of Product Price, Display Orientation and Shelf-Space Allocation in Retail Category Management. *Journal of Retailing* (2010).

[22] Walter Nicholson and Christopher Snyder. 2012. *Microeconomic Theory: Basic Principles and Extensions, 11th edition.* South-Western.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[24] J Salvatier, T.V. Wiecki, and C Fonnesbeck. 2016. The Truncated Normal Distribution. *PeerJ Computer Scienc* (2016).

[25] Robert Tibshirani Trevor Hastie and Jerome Friedman. 2009. *The Elements of Statistical Learning.* Springer, New York, NY.

[26] Guanjie Wei, Hua anad Zheng, Huaxiu Yao, and Zhenhui Li. 2018. IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control. In *in Proceedings of the 2018 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'18).*

[27] Fred S. Zufryden. 1986. A Dynamic Programming Approach for Product Selection and Supermarket Shelf-Space Allocation. *The Journal of Operational Research Society* (1986).

A SUPPLEMENTARY MATERIALS

A.1 Hyperparameters

To assist in reproducing our experiments, we detail the hyperparameter settings used in our implementations of `PSD-sim` and DQN.

A.1.1 PSD-sim Priors. Below we provide the hyperparameters of prior distributions used in our experiments:

- $\lambda = 1.5$
- $\boldsymbol{\mu_r} = [0, ..0]^\top$
- $\gamma = 25$
- $\boldsymbol{\mu_p} = [2.5, ..2.5]^\top$
- $\phi_s = 1$
- $\psi_s = 2.5$
- $\boldsymbol{\delta}_t = [5, 5, 5, 5, 10, 15, 0]^\top$
- $\boldsymbol{\Gamma}_t = \mathbf{I} * 10$
- $\sigma_t = 2.5$
- $\alpha_q = 1$
- $\beta_q = 1$

A.1.2 DQN. Finally, we provide details related to training our implementation of DQN:

- Discount factor: .2
- Learning starts: 1,500
- Batch size: 32
- Learning rate: 5e-4
- Training iterations: 50,000
- Exploration fraction: 99% annealed to 5% over the first 35% of training iterations