ONLINE APPENDIX: "Recommending for a Multi-Sided Marketplace: A Multi-Objective Hierarchical Approach"

Contents

A Additional Technical Details for MOHR

A.1 Details for the ML models.

Table 1 summarizes the features used for predicting consumer conversion, consumer retention and basket value. Note that for the count-based features such as the number of impressions/views/orders, we include both the raw count and the *normalized count* as features, where the normalized count are divided by the average impression/view/order count at the position of the event, in order to correct for the position bias as illustrated in Fig.8 in Main Appendix B.4.1. The name/id of the source (e.g. i.e. the name of the row or "single" if the training instance is a single restaurant recommendation) is explicitly used as a feature.

For the gradient boosted trees as the predictive ML model, we use learning rate of 0.1, and maximum depth of the tree as 8, which is the same model architecture and capacity as the latest production system at the company.

The data used to train the ML models are *observational* data from randomly sampled global user logs, consisting of about 600 million impressions and 11 million orders. As an industry-wide challenge, observational data potentially brings bias in the training data due to their off-policy nature (Saito and Joachims 2021). Alternatively, one could rely on experimental data, such as the random ranking data as described in Section 5, which is free of off-policy bias. However, experimental data are usually costly to obtain (random ranking significantly hurt consumer experience). Therefore the platform is usually only willing to set aside a very small percentage of sessions as experimental data. In our case, the random ranking data we obtained from the company is only on 3% of overall sessions. As a result, they provide poor coverage of all the consumers and all the restaurants to be served as training data. Because of these challenges, we do not rely on random ranking data, but rather all available observational data as the training data for the ML models. We adopt off-policy correction techniques in training (described as part of the contextual features in Section 4.2) to alleviate the off-policy bias.

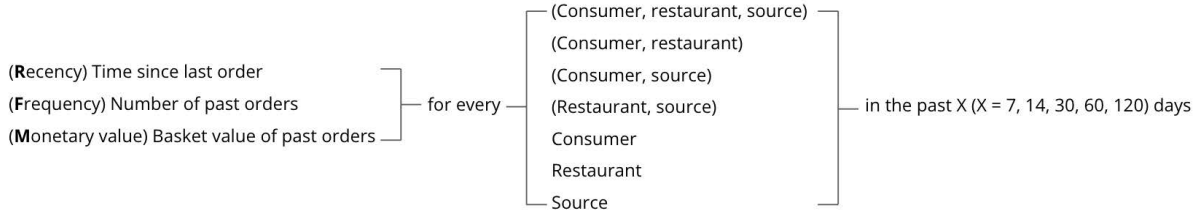


Figure A.1: RFM features in the MO-module.

Feature	Definition
a_i	(Normalized) Impression/view/order count/ratio from consumer i in the past T days
	Average basket values from consumer i in the past T days
	Consumer embedding u_i and $\tilde{u_i}$ from matrix factorization
a_j	(Normalized) Impression/view/order count/ratio on restaurant j in the past T days
	Average basket values from restaurant j in the past T days
	Percentage of consumers churned after ordering from restaurant j in the past T days
	Restaurant embedding v_j from matrix factorization
	Delivery radius of restaurant j
a_k	Source name (name of the row or "single" if appearing as single restaurant recommendation)
	(Normalized) Impression/view/order count/ratio from source k in the past T days
	Average basket values from source k in the past T days
	Source embedding $\tilde{w_k}$ from matrix factorization
a_{ij}	(Normalized) Impression/view/order count/ratio between consumer i and restaurant j in past T days
	Haversine distance between restaurant j and consumer's delivery location
	Estimated delivery time range
	Delivery fee, busy area fee, service fee
	$u_i^T v_j$, i.e. dot product of consumer embedding and restaurant embedding from matrix factorization
	$cos(u_i, v_j) = \frac{u_i^T v_j}{\|\|u_i\|\|_2 * \|\|v_j\|\|_2}$., i.e. cosine similarity between consumer embedding and restaurant embedding
a_{ik}	(Normalized) Impression/view/order count/ratio from consumer i in source k in the past T days
	Average basket values from consumer i in source k in the past T days
	$\tilde{u_i}^T \tilde{w_k}$, i.e. dot product of consumer embedding and source embedding from matrix factorization
	$cos(\tilde{u_i}, \tilde{w_k})$, i.e. cosine similarity between consumer embedding and source embedding
a_{jk}	(Normalized) Impression/view/order count/ratio from restaurant j in source k in the past T days
	Average basket values from restaurant j in source k in the past T days
a_{ijk}	(Normalized) Impression/view/order count/ratio from consumer i, restaurant j, source k in the past T days
	Average basket values from consumer i, restaurant j, source k in past T days
z	Source k (name of the row or "single restaurant")
	Vertical & horizontal position of the recommendation item
	City, geolocation, language, device
	Temporal features including day of week, local hour of day, meal period

Table 1: List of features for the ML models in the MO-module. T=7,14,30,60,120.

A.2 Proof of the concavity of the Pareto frontier.

Proof. Proof of the concavity of the Pareto frontier. We prove the case for the trade-off between the conversion objective $C(\boldsymbol{x})$ and the basket value objective $B(\boldsymbol{x})$. The cases for other objectives readily follow.

Given B^* is a fixed constant independent of $\boldsymbol{x}$, we let $\lambda := \alpha_b B^*$ and rewrite the optimization problem as

$$\begin{aligned} \max_{\boldsymbol{x}\in\mathcal{F}} \ & C(\boldsymbol{x}) \\ \text{s.t. } & B(\boldsymbol{x}) \geq \lambda \end{aligned} \tag{1}$$

where $\mathcal{F} = \{\boldsymbol{x} \in \mathcal{E} : R(\boldsymbol{x}) \geq \alpha_r R^*,\ E(\boldsymbol{x}) \geq \alpha_e E^*\}$ is the feasible region for $\boldsymbol{x}$. Let $\boldsymbol{x}^*_\lambda$ be the solution to Eq.(1) which is a function of λ. We would like to show that $C(\boldsymbol{x}^*_\lambda)$ is concave in $B(\boldsymbol{x}^*_\lambda)$ as a function of λ. We decompose the proof into two steps, which are the two claims below.

Claim 1: $z(\lambda) := C(\boldsymbol{x}^*_\lambda)$ is a concave function of λ.

Proof. Define the Lagrangian function as

$$L_\lambda(\boldsymbol{x}, \mu) = C(\boldsymbol{x}) + \mu(B(\boldsymbol{x}) - \lambda), \qquad \mu \geq 0. \tag{2}$$

Therefore the dual problem for Eq.(1) can be written as

$$\begin{aligned} D_\lambda(\mu) &= \max_{\boldsymbol{x}\in\mathcal{F}} L_\lambda(\boldsymbol{x}, \mu) = -\mu\lambda + \max_{\boldsymbol{x}\in\mathcal{F}} \left(C(\boldsymbol{x}) + \mu B(\boldsymbol{x})\right) \\ &:= -\mu\lambda + \kappa(\mu), \end{aligned} \tag{3}$$

where $\kappa(\mu) := \max_{\boldsymbol{x}\in\mathcal{F}} \left(C(\boldsymbol{x}) + \mu B(\boldsymbol{x})\right)$. Because Eq.(1) is a feasible linear optimization problem, strong duality holds, i.e.

$$z(\lambda) = \max_{\boldsymbol{x}\in\mathcal{G}_\lambda} C(\boldsymbol{x}) = \min_{\mu\geq 0} D_\lambda(\mu), \tag{4}$$

where $\mathcal{G}_\lambda = \{\boldsymbol{x} \in \mathcal{F} : B(\boldsymbol{x}) \geq \lambda\}$ is the feasible region for Eq.(1). For any positive λ_1, λ_2 and $t \in [0, 1]$, we have

$$\begin{aligned} z(t\lambda_1 + (1-t)\lambda_2) &= \min_{\mu\geq 0} \left(-\mu(t\lambda_1 + (1-t)\lambda_2) + \kappa(\mu)\right) \\ &\geq t \min_{\mu\geq 0}(-\mu\lambda_1 + \kappa(\mu)) + (1-t) \min_{\mu\geq 0}(-\mu\lambda_2 + \kappa(\mu)) \\ &= tD_{\lambda_1}(\mu) + (1-t)D_{\lambda_2}(\mu) \\ &= tz(\lambda_1) + (1-t)z(\lambda_2),\ \forall t \in [0, 1]. \end{aligned} \tag{5}$$

Therefore by definition of concavity, $z(\lambda)$ is concave in λ.

Claim 2: $B(\boldsymbol{x}^*_\lambda)$ is a piecewise linear function of λ. Specifically, $B(\boldsymbol{x}^*_\lambda) = \lambda_0$ for $\lambda \leq \lambda_0$, $B(\boldsymbol{x}^*_\lambda) = \lambda$ for $\lambda > \lambda_0$.

Proof. Let

$$\boldsymbol{x}_0 = \arg\max_{\boldsymbol{x}\in\mathcal{F}} C(\boldsymbol{x}) \tag{6}$$

be the solution to a modified version of Eq.(1) that relaxes the feasible region from $\mathcal{G}$ to $\mathcal{F}$. If there is more than one solution to Eq.(6), pick x_0 to be the one such that $B(\boldsymbol{x})$ is *maximized*. Let $\lambda_0 = B(\boldsymbol{x}_0)$. λ_0 can be bigger or smaller than λ. We discuss the two cases separately below.

If $\lambda \leq \lambda_0$, then x_0 is also the solution to the original optimization problem in Eq.(1). Therefore $B(\boldsymbol{x}^*_\lambda) = \lambda_0$.

Otherwise, if $\lambda > \lambda_0$, next we show that $B(\boldsymbol{x}^*_\lambda) = \lambda$. Because $\mathcal{G} \subseteq \mathcal{F}$ and $\boldsymbol{x}^*_\lambda = \arg\max_{\boldsymbol{x}\in\mathcal{G}} C(\boldsymbol{x})$, we have

$$C(\boldsymbol{x}^*_\lambda) \leq C(\boldsymbol{x}_0). \tag{7}$$

We know that

$$B(\boldsymbol{x}_0) = \lambda_0 < \lambda \leq B(\boldsymbol{x}^*_\lambda). \tag{8}$$

If $C(\boldsymbol{x}^*_\lambda) = C(\boldsymbol{x}_0)$, by Eq.(8) it contradicts with the assumption that $\boldsymbol{x}_0$ is picked among the optimal solutions such that $B(\boldsymbol{x})$ is maximized. Therefore the inequality in Eq.(7) is strict, i.e.

$$C(\boldsymbol{x}^*_\lambda) < C(\boldsymbol{x}_0). \tag{9}$$

Note that if $B(\boldsymbol{x}^*_\lambda) > \lambda$, we have $B(\boldsymbol{x}^*_\lambda) > \lambda > B(\boldsymbol{x}_0)$. By linearity of $B(\cdot)$, we have that there exists a $\boldsymbol{x}' = t'\boldsymbol{x}_0 + (1-t')\boldsymbol{x}^*_\lambda$ such that $t' \in (0,1)$ and $B(\boldsymbol{x}') = \lambda$. Then by linearity of $C(\cdot)$ and Eq.(9), we have

$$\begin{aligned} C(\boldsymbol{x}') &= t'C(\boldsymbol{x}_0) + (1-t')C(\boldsymbol{x}^*_\lambda) \\ &> t'C(\boldsymbol{x}^*_\lambda) + (1-t')C(\boldsymbol{x}^*_\lambda) \\ &= C(\boldsymbol{x}^*_\lambda). \end{aligned} \tag{10}$$

Because the feasible region $\mathcal{G}$ is convex and $\boldsymbol{x}'$ is a linear combination of two points within $\mathcal{G}$, we have $\boldsymbol{x}' \in \mathcal{G}$ but $C(\boldsymbol{x}') > C(\boldsymbol{x}^*_\lambda)$. This contradicts the fact that $\boldsymbol{x}^*_\lambda$ is the optimal solution for Eq.(1). So we must have $B(\boldsymbol{x}^*_\lambda) = \lambda$ for $\lambda > \lambda_0$. So $B(\boldsymbol{x}^*_\lambda)$ is a piecewise linear function in λ.

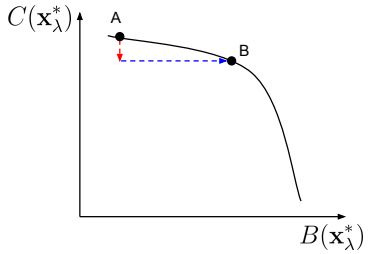


Figure A.2: An illustration of a concave trade-off curve. A small sacrifice in one of the objectives can lead to a big improvement in the other.

Finally, combining Claim 1 and 2, we arrive at the conclusion that $C(\boldsymbol{x}_\lambda^*)$ is concave in $B(\boldsymbol{x}_\lambda^*)$. In other words, the trade-off curve between $C(\boldsymbol{x}_\lambda^*)$ and $B(\boldsymbol{x}_\lambda^*)$ with varying λ is a concave curve. The benefit of a concave trade-off curve is illustrated in Fig.A.2. Comparing with point A on the trade-off curve, point B achieves a big boost in $B(\boldsymbol{x}_\lambda^*)$ with only a small sacrifice in $C(\boldsymbol{x}_\lambda^*)$. □ □

B Additional Details on Offline Evaluation

B.1 Stability of ML Model Performance

Figure B.3 shows the AUC (on next day's test data) of the consumer conversion model in the MO-module over consecutive 21 days. For a training data of 30 days, the initial 29 days are utilized for training, while the final day is reserved for validation. This temporal split of the training and validation datasets replicates the online setting, where models trained using data up until the previous day are deployed to handle the current day's traffic. Therefore the offline AUC can be viewed as a proxy metric for model online performance (on future data that are potentially off-policy). The next-day AUC in Fig.B.3 is stable across the 21-day period, with a maximum of 0.8640, and a minimum of 0.8488, all within 1% change of the mean value 0.8562. In other words, the off-policy model performance is stable over time, demonstrating that the techniques described in Section 5.1 are effective in enabling the ML models to consistently capture the evolving dynamics of the multi-sided marketplace.

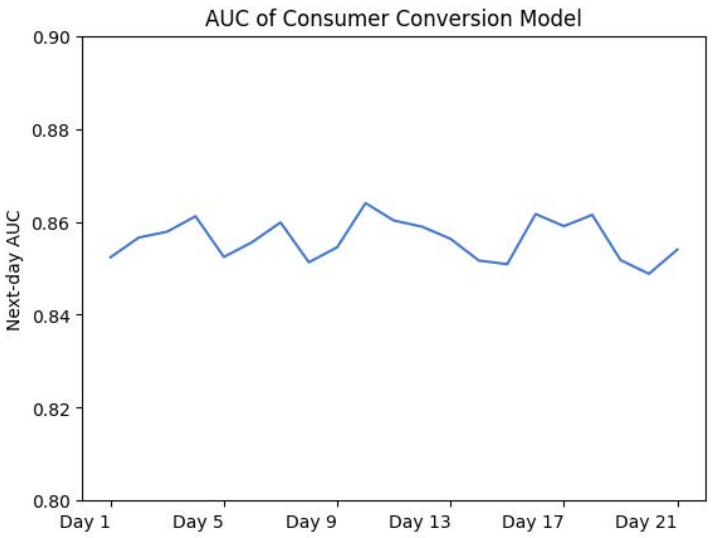


Figure B.3: AUC of the consumer conversion model (on next-day test data) over 21 days.

B.2 Analysis of Short-Term Session-Level Bookings

B.2.1 Offline Analysis

Using the random ranking data and the offline replay methodology described in Section 5.2.1, we generated the offline predictions for conversion rates and basket values for different values of the basket value objective weight (λ_b). In addition to the offline Pareto frontiers presented in Section 5.2.2, we also present the offline results on short-term session-level bookings here, which is computed by multiplying the conversion rate by the basket value at each λ_b. Figure B.4 shows the result of the short-term session-level bookings. We see that as a result of the trade-off between conversion rate and basket value presented in Fig.4a, the short-term session-level bookings, which equals the product of the two values, remains relatively flat for different λ_b values. Therefore, the analysis of offline short-term profit is not particularly helpful in selecting candidate values for λ_b for online experiments. Instead, given that conversion is the most important business metric for the company, we select candidate values for λ_b so that the trade-offs lie in the "flat region" of the Pareto frontiers in Fig.4, i.e. the drop in conversion rate is minimal.

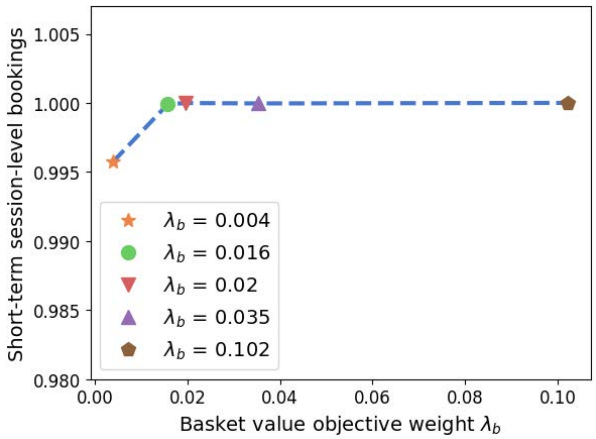


Figure B.4: Short-term session-level bookings from offline replay analysis. To preserve confidentiality, the tick values on the y-axis represent the percentage over the maximum values.

B.2.2 Online Analysis

Figure B.5 shows the short-term session-level bookings for varying λ_b values in the online experiments described in Section 6.3.2. We see that apart from the extreme weight $\lambda_b = 0.1$ under which the basket value objective dominates the final ranking, the short-term session-level bookings for the other λ_b's are statistically indistinguishable. In particular, they are all within 0.5% difference of each other as predicted by the offline analysis in Fig.B.4. However, at the extreme boost $\lambda_b = 0.1$, there is a significant drop in short-term session-level bookings, which is different from the offline prediction. The reason for this

is the "backfire" effect that causes consumers losing trust in the recommender system and abandoning homepage, as described in Section 6.3.2 and Web Appendix C.4 below.

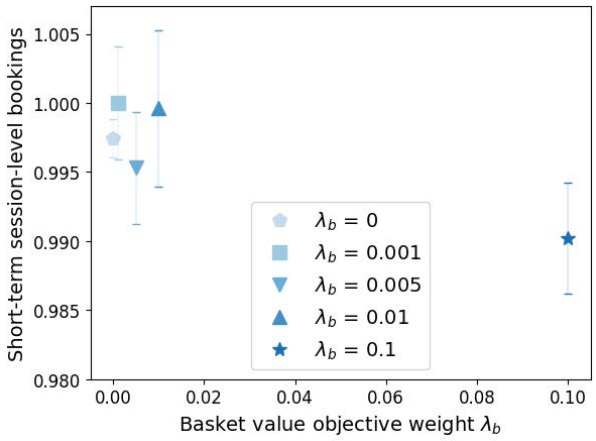


Figure B.5: Online short-term session-level bookings with varying λ_b. To preserve confidentiality, the tick values on the y-axis represent the percentage over the maximum values.

C Additional Details on Online Experiments

C.1 Results on the H-module

Table 2 and Fig.C.6 presents the estimated scrolling factors in the experiment. Note that there are at most 7 restaurants presented in every row. To see more restaurants within the row, there is a "see all" button at the top right corner of every row. The H-module is only applied to the top 7 positions within each row.

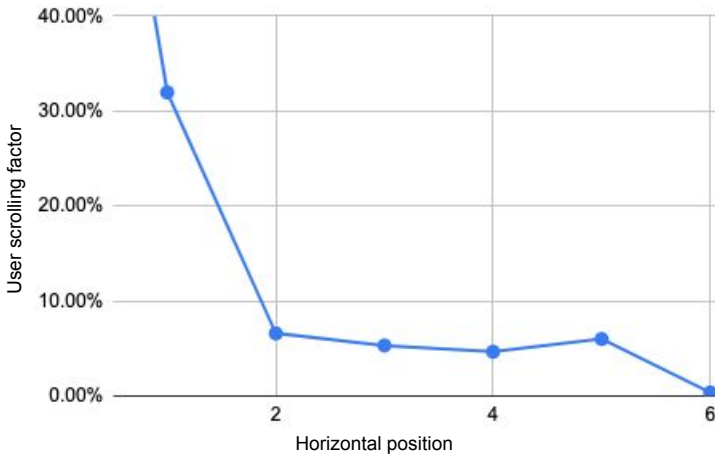


Figure C.6: Estimated scrolling probabilities in the H-module.

Horizontal position	Consumer scrolling factor $\hat{p}_{l,l+1}$
0	100.00%
1	31.96%
2	6.58%
3	5.32%
4	4.65%
5	5.99%
6	0.36%

Table 2: Estimated scrolling probabilities in the H-module.

C.2 Comparison with LinUCB as an Additional Baseline

We compare the performance of MOHR against an additional baseline, LinUCB. LinUCB (Linear Upper Confidence Bound), proposed by Li et al. (2010), is one of the most popular algorithms used in bandit problems and belongs to the family of UCB (Upper Confidence Bound) algorithms. We were unable to conduct additional live experiments due to the constraints of the company, but luckily we were able to conduct *unbiased* offline evaluation using the same random ranking data and offline replay method outlined in Section 5.2.1.

LinUCB is designed to address the exploration-exploitation trade-off by maintaining estimates of the expected rewards associated with each arm a. In particular, these rewards, $r_{t,a}$, are *linear* functions of the features $x_{t,a}$:

$$\mathbb{E}[r_{t,a}|x_{t,a}] = x_{t,a}^T \theta_a^*,$$

with some unknown coefficient vector θ_a^*. The estimates of θ_a^*, denoted as $\hat{\theta}_a$, are updated based on the agent's interactions with the environment. The algorithm assigns a confidence interval to each arm's estimated reward, and the arm with the highest upper confidence bound is chosen:

$$a_t \stackrel{\text{def}}{=} \arg\max_{a \in \mathcal{A}_t} \left(x_{t,a}^T \hat{\theta}_a + \alpha \sqrt{x_{t,a}^T A_a^{-1} x_{t,a}} \right), \tag{11}$$

where $\mathcal{A}_t$ is the set of actions at trial t. In the context of restaurant recommendation, $\mathcal{A}$ is the set of restaurant candidates. A_a tracks the covariance matrix of the d-dimensional feature $x_{t,a}$ and is updated when action a is picked. α controls the exploration-exploitation trade-off and higher α values lead to more exploration. For more details of the LinUCB algorithm see Algorithm 1 of Li et al. (2010). As LinUCB is only designed for one-dimensional ranking, in the experiments below we compare MOHR against LinUCB with fixed within-row ranking. Note that this does *not* offer MOHR more advantage, as the with-row rankings are fixed for both MOHR and LinUCB in the offline experiments.

In our experiments, we consider LinUCB with the following objectives: (1) consumer

conversion; (2) basket value; (3) restaurant exploration. We were unable to conduct offline evaluation for the consumer retention objective for the same reason mentioned in Section 5.2.2.

An important input to the LinUCB algorithm is the feature vector $x_{t,a}$ for every trial t and every action a. Due to the limitation of the engineering infrastructure at the company, we were unable to extract the full feature set used by the ML models in the MO-module. In addition, as the complexity of the LinUCB algorithm scales with d^3 where d is the number of features (this is due to the inversion of A_a in Eq.(11)), an increasing number of features will rapidly make LinUCB slower to run. Therefore, we aim to replicate features from the feature pipeline to the maximum extent possible, while ensuring that the running time of the LinUCB algorithm remains within acceptable time limits. As a result, we ended up with 12 features from the top 20 features used by the ML models in the MO-module, which are: delivery fees, average restaurant ratings in past 30 days, restaurant order count in the past 120 days, restaurant order ratio in the past 14 and 30 days, (consumer, restaurant) impression-to-order ratio in the past 14 and 60 days, (consumer, restaurant) impression-to-click ratio in the past 14 and 60 days, whether the consumer has membership, whether the restaurant belongs to a chain, weekday vs. weekend, and consumer order counts in the past 120 days.

C.2.1 LinUCB with conversion as the reward

Figure 6 in Section 6.4.1 of the main text shows the results from offline replay comparing MOHR against LinUCB with various α values, on the same offline random ranking data as described in Section 5.2.1. There are about 140k matched sessions under the offline replay methodology for both methods. Note that the MOHR trade-off curves (in blue dashed lines) in Fig.6a and 6b are exactly those in Fig.4, but plotted on a much wider x-axis and y-axis.

There are several observations from these plots. First, Fig.6a clearly illustrates that the counterfactual consumer conversions and basket values achieved using the LinUCB approach are consistently dominated by those attained through MOHR, indicating the superiority of MOHR. In particular, the objective values under LinUCB only achieves 50%-93% of those under MOHR. This can be mostly attributed to the fact that linear reward models are expected to perform worse than ML models in their predictive powers. LinUCB consists of two components: a reward model ($x_{t,a}^T \theta_a^*$) for "exploitation", and an uncertainty component ($\sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}$) for "exploration". Both components are derived based on the assumption that the true reward model is *linear* in the covariates. The theoretical guarantees of LinUCB are also established based on the assumption that the linear reward model is correctly specified. However, the linear assumption is quite limiting and often unrealistic in practical scenarios, such as in real-time conversion rate prediction,

leading to potential *model misspecification*. Consequently, since the exploration mechanism relies on the accuracy of the linear reward model, which is likely misspecified in real-world contexts, its ability to conduct effective exploration is compromised due to the inaccuracies in the uncertainty component. On the contrary, MOHR's conversion model utilizes a machine learning framework capable of identifying non-linear relationships and interaction effects across a comprehensive set of features. Therefore, it is expected that MOHR will more accurately estimate conversion rate than LinUCB, contributing to the superiority of MOHR over LinUCB approaches.

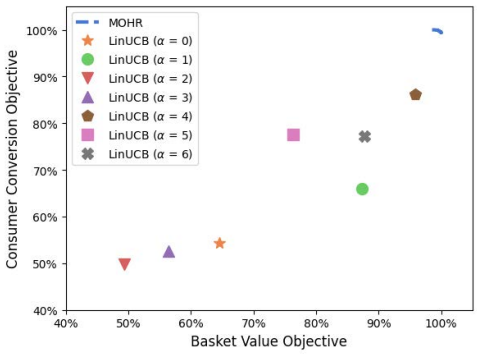
Second, the trade-offs between conversion and restaurant exploration under LinUCB is better (the colored dots in Fig.6b are closer to the blue curve compared with Fig.6a) although still mostly dominated by MOHR. This is expected as LinUCB is designed for optimizing the trade-off between exploration (y-axis in Fig.6b) and exploitation (x-axis in Fig.6b) for improving long-term conversion, but *not* the trade-off between conversion and basket value. Interestingly, the restaurant exploration objective under LinUCB can potentially achieve even better values than those under MOHR ($\alpha = 2$ and $\alpha = 3$ in Fig.6b result in exploration objective values exceeding 100%), probably because the restaurant exploration objective is positively correlated with the uncertainty component in LinUCB. However, these aggressive improvements in the exploration objective result in a 40% drop in the consumer conversion objective. Given that consumer conversion stands as the top business goal within the company, our focus remains only on segments of the trade-off curve where any potential sacrifice in conversion is limited to a mere fraction (e.g., less than 1%). Thus, in the regions where the trade-offs are practical to the company (top left region of the Pareto frontiers), MOHR still strictly dominates all LinUCB approaches.

Third, we notice that unlike MOHR where the objective weights control the trade-off in a monotone way (Fig.4), there is no monotone relationship between the α values and the trade-off curve. This is expected as the objective weights in MOHR are designed to control the importance of multiple objectives, while α values in LinUCB aims at controlling the trade-off between exploration and exploitation in optimizing a single objective (Even in Fig.6b where α is exactly expected to control the degree of exploration, the restaurant exploration objective value we defined is on restaurant level and not the same as the uncertainty component in the LinUCB framework, therefore we do not expect these values to align.).

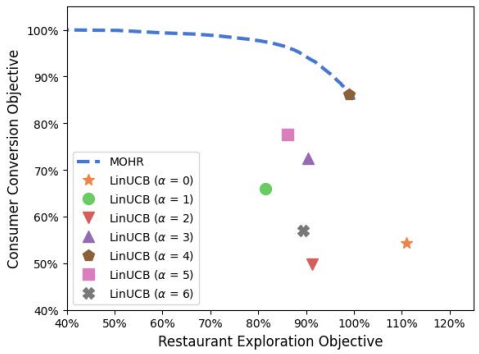
Lastly, we would like to call out the running time differences between LinUCB and MOHR in our experiments. Assuming there are T sessions and N restaurants in total, the complexity of MOHR over these sessions is $\mathcal{O}(NT \log N)$, while LinUCB is $\mathcal{O}(NT \log N + NTd^3)$. When $d^3 \gg \log N$, which holds true in our case, the running time for LinUCB is much longer than that of MOHR.

C.2.2 LinUCB with basket value or restaurant exploration as the reward

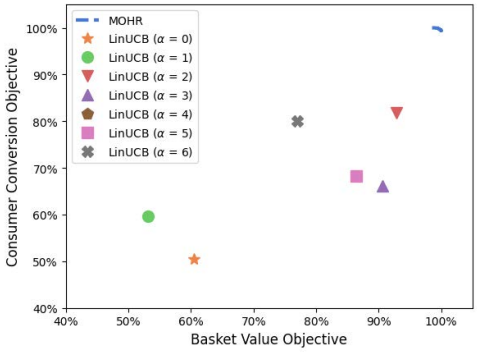
While this might not be a typical configuration for LinUCB, we also conducted experiments with basket value or restaurant exploration as the single objective for LinUCB. Figure C.7 shows the trade-off plots from these additional experiments. We see that there are minor improvements in the corresponding objectives that are used as the LinUCB objectives. Specifically, the basket values are improved when basket value is used as the LinUCB objective (Fig.C.7a); The restaurant exploration values are improved when restaurant exploration is used as the LinUCB objective (Fig.C.7b). This is expected as LinUCB is designed to optimze the single objective of interest. However, MOHR still dominates LinUCB in these trade-offs. The observations are similar to those from when using conversion as the single objective (Fig.6) and confirms the superior performance of MOHR.



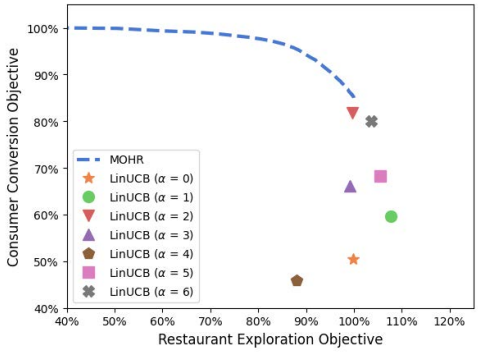
(a) Trade-off between consumer conversion and basket value; Basket value is used as the single objective for LinUCB.



(b) Trade-off between consumer conversion and restaurant exploration; Basket value is used as the single objective for LinUCB.



(c) Trade-off between consumer conversion and basket value; Restaurant exploration is used as the single objective for LinUCB.



(d) Trade-off between consumer conversion and restaurant exploration; Restaurant exploration is used as the single objective for LinUCB.

Figure C.7: Additional experiments comparing MOHR and LinUCB, which uses basket value or restaurant exploration as the single objective for LinUCB. Tick values are presented as percentages relative to the maximum values of the objectives under MOHR, while actual values are omitted due to business compliance reasons.

C.3 Additional Baselines for Hierarchical Ranking

The hierarchical context we are dealing with involves ranking rows of products along with single products. To the best of our knowledge, there are no publicly available benchmarks in these settings. Despite this, similar hierarchical settings are prevalent in prominent content recommendation platforms like Spotify and YouTube. Unfortunately, these platforms haven't yet disclosed their benchmarks to the public or academia.

However, drawing from anecdotal insights, we can summarize a few practices employed within the industry for hierarchical ranking. The main challenge is the calibration among different levels of aggregation of products (e.g. comparing the attractiveness of a row of products vs. individual products). To address this calibration challenge, the industry has mainly adopted the following three approaches for ranking in hierarchical settings:

- **Heuristic-Based Approach**: Rows are assigned fixed positions in the feed (e.g., positions 2, 8, and 14 for rows), with other positions reserved for single products.
- **Disjoined Systems**: Rows and single products are ranked independently using separate ML systems, with another model deciding the number of rows to display. The production system at the company adopts this approach as described in main Appendix C.2, which is used as the baseline in our experiments.
- **Dimension Reduction**: Only the first product within each row is used in determining the ranking score for that row, as opposed to using all products available in the row. This effectively reduces the two-dimensional ranking problem into a one-dimensional ranking problem.

Disjoined Systems were previously shown to outperform Heuristic-Based Approaches by the company, leading the company to adopt them as their production recommender system. In our online experiments, we showed that MOHR outperformed the Disjoined Systems (Section 6.2), leading us to believe that it also outperforms the Heuristic-Based Approaches. The first ablation study in the H-module (Appendix C.3) belongs to the Dimension Reduction approach, where the ranking score for a row equals to that of the first restaurant within the row. Remarkably, MOHR also outperforms such Dimension Reduction approaches. In summary, our work demonstrates that MOHR achieves superior performances to all three recognized approaches for hierarchical ranking.

C.4 "Backfire" effects with extreme λ values.

In this section, we provide more insights on the "backfire" effect observed in Fig.5b at $\lambda_b = 0.1$, and why the offline Pareto frontier in Fig.4a failed to capture such events.

When the weight on the basket value objective is increasing in a reasonable range ($\lambda_b \in [0.001, 0.01]$), the consumers are placing more expensive orders but less likely

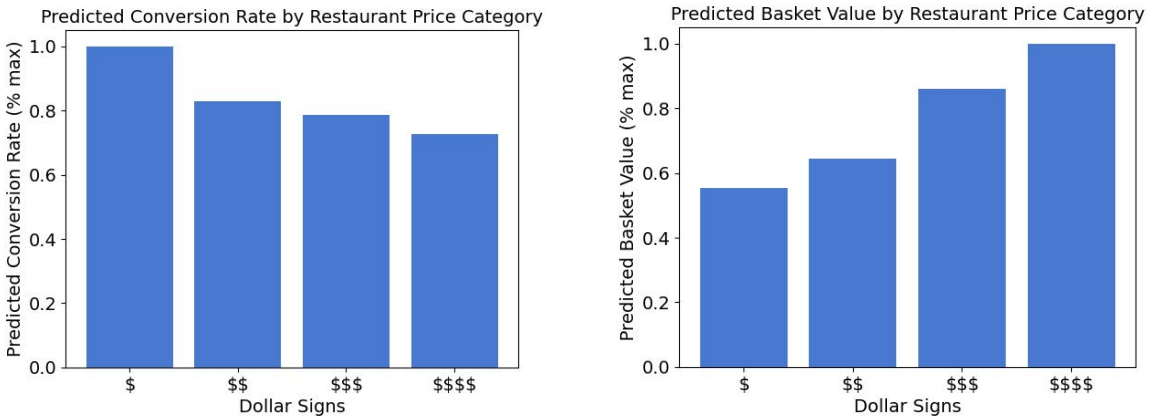
to order (shown as the trade-off between "actual basket value per order" and "actual conversion rate" in Fig.5a). When the weight on the basket value objective is extreme such that the restaurants are essentially ranked by the predicted basket value, the consumers lose trust in the recommender system. This is evidenced by the increase in search rate and order position, and decrease in new consumer retention.

For the consumers who went to the Search tab, the search ranking algorithm used by the platform is a single-objective ranking function based on conversion (MOHR is only applied to the homepage), so the basket value of those orders placed from the Search tab will look similar to when $\lambda_b = 0$. For the consumers who remained on the homepage, we looked into the past experiment results and found that there is also a significant 3.8% increase in the vertical order position, which suggests that the consumers were scrolling deeper down the feed and ordering from cheaper restaurants.

The fact that conversion rate does not drop under $\lambda_b = 0.1$ should be because of the switching cost in the current session: the consumer is hungry and they will place an order even if the homepage recommendations are unsatisfactory. This explains the observation that the data point for $\lambda_b = 0.1$ has a similar conversion rate and basket value as the data point for $\lambda_b = 0$ in the short term. But the fact that retention drops for new consumers is a sign for consumers losing trust on the recommender system and the platform overall.

Note that the observed "backfire" effect does *not* suggest inaccuracies in the ML models predicting basket value or conversion rates. On the contrary, both models generate reasonable predictions, particularly in scenarios involving pricier restaurants. For instance, as illustrated in Fig.C.8, with restaurants categorized by increasing price levels (indicated by the number of dollar signs), there's a notable trend: the average predicted conversion rate decreases (Fig.C.8a), while the average predicted basket value—assuming an order is placed—increases (Fig.C.8b). This confirms that both models are capturing the correct trend for pricier restaurants: consumers are less likely to order from them, but when they do, the average basket value is higher.

Instead, we would like to call out that this "backfire effect" comes from the fact that the recommendation framework is not explicitly modeling the "trust" component for consumers. This is a limitation that is not unique to our MOHR framework, but rather, all recommender systems. Specifically, recommender system literature assumes that the consumers will make a decision based on characteristics of the individual products, *not* how much they trust the recommender system or enjoyed the homepage overall. Therefore, they are not able to capture extreme scenarios where the aggregated recommendation results lead to a diminished trust in the system. There are mainly two reasons why trust is not explicitly modeled in current recommender systems. First, accurately modeling such rare and extreme events would require an extensive dataset capturing these events across diverse contexts. For food delivery platforms, this would involve presenting exclusively high-priced restaurants to a wide-ranging sample of consumers under various



(a) (Normalized) Predicted conversion rate.

(b) (Normalized) Predicted basket value.

Figure C.8: Predicted conversion rate and basket value by restaurant price category. Values are represented as percentage over the maximum values for each quantity for compliance reasons.

conditions—a dataset that would be prohibitively expensive to obtain. Second, there is little practical meaning in modeling these extreme events where consumers lose trust in recommender systems, as they rarely happen. Instead, a more beneficial approach for platforms is to implement safeguards to ensure the overall quality of recommendations. For instance, the food delivery platform we collaborate with has established a (personalized) conversion rate threshold, and restaurants falling below this threshold are removed from the ranking process. This measure effectively guarantees that consumers are unlikely to see a recommendation list dominated by the most expensive restaurants (unless their previous orders were exclusively from such restaurants), making specific scenarios like the extreme test group ($\lambda_b = 0.1$) unlikely to happen in the real world.

Now we dive into why the offline evaluation in Fig.4a failed to capture such backfire effects at extreme λ values. On a high level, the mismatch between offline and online results at the extreme weight $\lambda_b \approx 0.1$ is because of the fundamental limitation of offline evaluation on out-of-distribution (OOD) data. Specifically, at $\lambda = 0.1$, the final ranking function is dominated by the basket value objective (the basket value objective is about 2 to 3 orders of magnitude larger than the other three objectives) and the recommended restaurants are exclusively more expensive ones. The offline random ranking data used by the offline evaluation is extremely unlikely to cover such extreme ranking outcomes. Although we cannot disclose the accurate number of restaurants available to each consumer, we provide a hypothetical example here to illustrate this point. Suppose there are 300 restaurants in total and the consumers usually browse the top 20 recommended restaurants. In the random ranking data, all 300 restaurants are ranked completely at random. Then the probability that the top 20 restaurants are all from the top 30 most

expensive restaurants (i.e. top 10 percentile in terms of basket value) is equal to

$$\begin{aligned}&\mathbf{P}(\text{seeing exclusively expensive restaurants in top 20 positions} \mid \text{random ranking})\\ =&\frac{\binom{30}{20} \cdot 20! \cdot 280!}{300!} \approx 4 \times 10^{-24},\end{aligned} \tag{12}$$

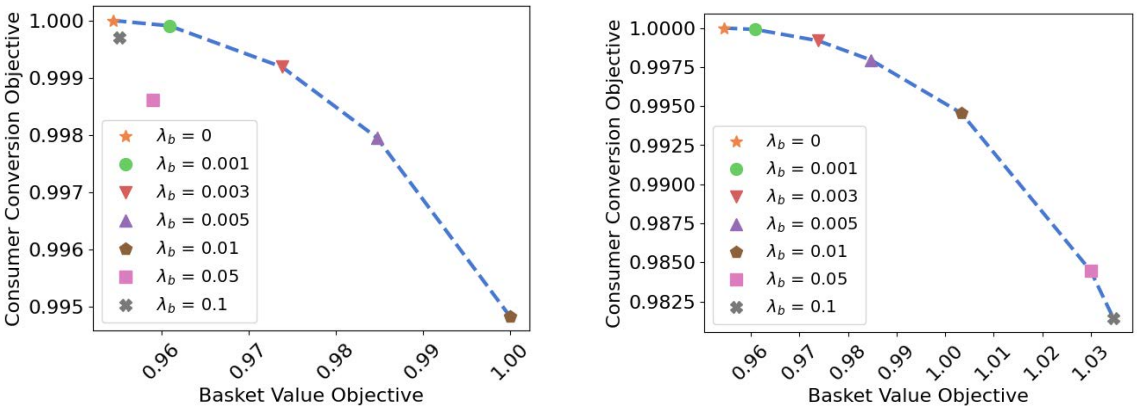
which is implying that such scenarios would basically never appear in the offline evaluation data, i.e. they are considered as out-of-distribution. Without such data, the offline replay method we used is unable to capture the phenomenon where consumers lose trust and subsequently abandon the homepage when seeing an exclusive list of expensive restaurants in the recommendation results. And this is simply because this scenario was almost never observed in the random ranking data. As a result, the offline Pareto frontier merely *extrapolates* to out-of-distribution scenarios based on existing trends. It inaccurately predicts that a huge weight on the basket value objective will lead to a huge boost in the actual basket value per order (Fig.4a).

To further illustrate this point, we conducted additional simulations. We simulated $I = 300$ consumers and $J = 500$ restaurants, and generated the conversion matrix $C = (c_{ij}) \in \mathbf{R}^{I \times J}$ and the basket value matrix $B = (b_{ij}) \in \mathbf{R}^{I \times J}$, where c_{ij} and b_{ij} represent the ground truth conversion and basket value between consumer i and restaurant j, and are randomly sampled from lognormal distributions with ranges similar to those from the ML estimates (we unfortunately cannot disclose those values due to business compliance reasons). We assume the system has *perfect* knowledge of the conversion rate and basket value for every $(\text{consumer}, \text{restaurant})$ pair, i.e. the ML conversion model and the ML basket value model have *zero* prediction errors. To explicitly incorporate the extreme case of consumer losing trust in the homepage when seeing extreme recommendations in the offline evaluation, we generate the consumer's conversion behavior according to the following mechanism: If all of the top N recommended restaurants for a consumer are from the top x percentile in terms of the basket value, then she will abandon the homepage and go to the Search tab, which uses the default ranking function with conversion as the single objective (i.e. all λ values are zero). In order words, consumers will lose trust in the recommender systems when they see exclusively expensive recommendations, and they will abandon the homepage and behave as if there were no multi-objective recommendation.

By explicitly incorporating such out-of-distribution data of losing trust, we show that the offline evaluation framework will now be able to predict the "backfire" effect as observed in the online experiments. Specifically, we regenerate the offline Pareto frontier using the simulation data with $N = 10$ and $x = 20$ (i.e. consumers are simulated to abandon the homepage if *all* top 10 recommended restaurants are from the top 20% expensive restaurants).

Fig.C.9a shows that the "backfire" effect indeed emerged when $\lambda_b = 0.1$, effectively closing the gap between the offline Pareto frontier and the online results originally observed in Fig.4 and Fig.5. As a sanity check, we conducted an ablation study with the trust component *removed*, assuming that consumers would not abandon the homepage even when presented with an exclusive list of expensive recommendations. The results are shown in Fig.C.9b. We see that the "backfire" effect disappears in this scenario, replicating the original offline Pareto frontier result shown in Fig.4a. This reaffirms that trust is indeed the underlying mechanism contributing to the observed discrepancy between the offline Pareto frontier (Figure 4a) and the online Pareto frontier (Figure 5b) at $\lambda_b = 0.1$: Given the ML models in the simulation are designed to have perfect predictions of the objectives, the only difference between the data used to generate Fig.C.9a and Fig.C.9b is the presence of the trust component.

Lastly, we emphasize that the observed "backfire effect" rarely happens in practice and is out of managerial interest. Specifically, given that conversion is the top important business metric for the company, *only* the top left regions of the Pareto frontiers (where the drop in conversion is minimal) in Fig.4 and Fig.5 are of practical importance to the platform. In other words, the cases when there are aggressive boosts on other objectives are not practically meaningful for the platform, as they would never be comfortable to sacrifice that amount of loss in conversion in exchange for gains in other objectives.



(a) With the trust mechanism.

(b) Without the trust mechanism.

Figure C.9: Simulation of the offline Pareto frontier with (Fig.C.9a) and without (Fig.C.9b) the trust mechanism.

C.5 Service Quality Objective

Approximately one year after the initial launch of MOHR, the company experimented with an additional objective, called service quality. It models the probability that an order will be successfully fulfilled in time, which is a binary outcome capturing delays,

cancellations, courier unfulfilled orders and any customer reported issues:

$$s(i,j,k) = \mathbf{E}[S(i,j,k,z)|O(i,j,k,z)=1] = f_s(a_i, a_j, a_k, a_{ij}, a_{ik}, a_{jk}, a_{ijk}, z), \tag{13}$$

where $S(i,j,k,z)$ is a binary random variable taking value 1 if the order from consumer i on restaurant j from source k is fulfilled in time without a support ticket afterwards, and 0 otherwise. Similar to the basket value and consumer retention objective, it is a *counterfactual* estimation conditioning on the consumer placing an order in the current session. The offline AUC for the service quality model is 0.61.

The inclusion of the service quality outcome (with optimal objective weight equal to 1.5) resulted in a 0.7% decrease in the order unfulfillment rate, without significantly impacting other core business metrics. Considering the cost of customer support at the company, this reduction translate to a $2.9 million decrease in support staffing cost.

C.6 Robustness Checks

C.6.1 Randomization Check.

To check if the random assignment for the online experiments truly holds, we inspect the treatment and control consumers *before* the experiment start date, when they were all receiving recommendations from the same production recommender system. This is called A/A test in the industry. Specifically, we collect 472 metrics related to different aspects of consumer behaviors[1] including the key business metrics in the results above, and conduct hypothesis testing on whether the differences between treatment and control groups are statistically significant before the experiment start date. Specifically, we compute the p-values for the metric differences between treatment and control group 28 days *before* the experiment start date, when both treatment and control consumers are expected to receive recommendations generated by the same algorithm. Table 3 shows that the A/A testing p-values are all greater than 0.05 (or 0.10 depending on the significance level of choice), suggesting that there is no significant difference in the treatment and control group in terms of the key business metrics, before the experiment start date.

Metric	Conversion rate	Basket value per order	Retention rate	Orders per consumer	Search rate
A/A testing p-value	0.326	0.452	0.947	0.853	0.286

Table 3: p-values for the A/A testing on key business metrics.

We further collected a comprehensive set of 472 metrics capturing various aspects of consumer behavior on the platform and across different surfaces, and computed the p-values for the 472 metric differences. Under the null hypothesis that treatment and control group consumers are not statistically different, the p-values should follow a uniform

distribution. We conduct the Kolmogorov-Smirnov (KS) test on the empirical distribution of those 472 p-values, and could *not* reject the null that they follow a uniform distribution on [0,1] (Fig.C.10), suggesting that our randomization holds true.

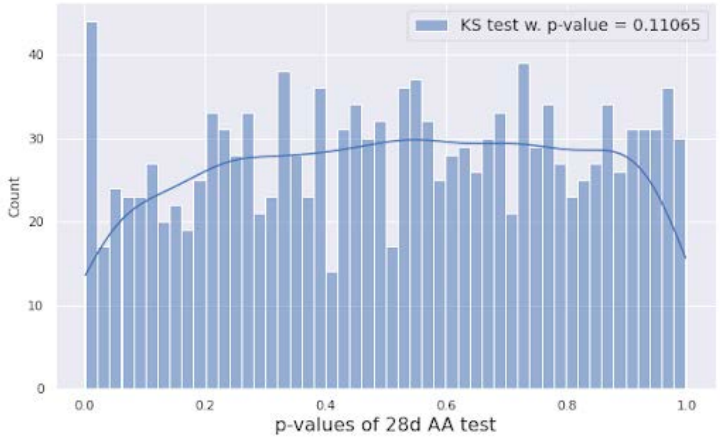


Figure C.10: Histogram of the p-values for the 472 metric differences for A/A test. Kolmogorov-Smirnov (KS) test which compares the empirical distribution of the p-values against the uniform distribution on [0,1] has p-value of 0.11, which fails to reject the null hypotheses that these metrics are not statistically significantly different during the A/A testing period.

C.6.2 Eliminating the Novelty Effects.

With the MOHR framework, rows and single restaurants are mixed together and the homepage appears as a heterogeneous arrangement of items. One might argue that the new display may introduce a novelty effect (Koch et al. 2018) and consumers' engagement levels with the platform might be higher in the beginning than when they become familiar with the new design. We identified three pieces of evidence to counter this argument. First, the first three days of experiment data is discarded in computing the metrics reported in the previous sections, which eliminates part of the novelty effect. Second, there is an improvement in long-term consumer retention (+0.7%) which by definition measures consumers' future engagement with the platform after finishing the current session. This means that the treatment effect of MOHR persists for at least 28 days. Lastly, we measured the metrics for the new consumers during the experiment, whose *first* interaction with the platform is either always under the current production system (control consumers) or always under the new display under the MOHR framework (treatment consumers). Therefore, there is no novelty effect at play for those consumers. We observe a significant 5.5% increase in the retention of new consumers which is even larger than the overall retention increase. This suggests that the novelty effect, if it exists, actually impacts the effects of the MOHR framework negatively as it introduces a "shock" to the existing consumers with a new display, so that the positive effects on them are actually smaller than the new consumers who have no prior experience with the platform.

References

Koch M, von Luck K, Schwarzer J, Draheim S (2018) The novelty effect in large display deployments–experiences and lessons-learned for evaluating prototypes. *ECSCW'18* (European Society for Socially Embedded Technologies (EUSSET)).

Li L, Chu W, Langford J, Schapire RE (2010) A contextual-bandit approach to personalized news article recommendation. *wWW'10*, 661–670.

Saito Y, Joachims T (2021) Counterfactual learning and evaluation for recommender systems: Foundations, implementations, and recent advances. *Recsys'21*, 828–830.