Learning User Preferences to Incentivize Exploration in the Sharing Economy

Christoph Hirnschall
ETH Zurich
Zurich, Switzerland
chirnsch@gmail.com

Adish Singla
MPI-SWS
Saarbrücken, Germany
adishs@mpi-sws.org

Sebastian Tschiatschek
Microsoft Research*
Cambridge, United Kingdom
setschia@microsoft.com

Andreas Krause
ETH Zurich
Zurich, Switzerland
krausea@ethz.ch

Abstract
We study platforms in the sharing economy and discuss the need for incentivizing users to explore options that otherwise would not be chosen. For instance, rental platforms such as Airbnb typically rely on customer reviews to provide users with relevant information about different options. Yet, often a large fraction of options does not have any reviews available. Such options are frequently neglected as viable choices, and in turn are unlikely to be evaluated, creating a vicious cycle. Platforms can engage users to deviate from their preferred choice by offering monetary incentives for choosing a different option instead. To efficiently learn the optimal incentives to offer, we consider structural information in user preferences and introduce a novel algorithm - Coordinated Online Learning (CoOL) - for learning with structural information modeled as convex constraints. We provide formal guarantees on the performance of our algorithm and test the viability of our approach in a user study with data of apartments on Airbnb. Our findings suggest that our approach is well-suited to learn appropriate incentives and increase exploration on the investigated platform.

Introduction
In recent years, numerous sharing economy platforms with a variety of goods and services have emerged. These platforms are shaped by users that primarily act in their own interest to maximize their utility. However, such behavior might interfere with the usefulness of the platforms. For example, users of mobility sharing systems typically prefer to drop off rentals at the location in closest proximity, while a more balanced distribution would allow the mobility sharing service to operate more efficiently.

Undesirable user behavior in the sharing economy is in many cases self-reinforcing. For example, users in the apartment rental marketplace Airbnb are less likely to select infrequently reviewed apartments and are therefore unlikely to provide reviews for these apartments (Fradkin 2014). This is also reflected in the distribution of reviews, where in many cities 20% of apartments account for more than 80% of customer reviews1.

Such dynamics create a need for platforms in the sharing economy to actively engage users to shape demand and improve efficiency. Several previous papers have proposed the idea of using monetary incentives to encourage desirable behavior in such systems. One example is (Frazier et al. 2014), who studied the problem in a multi-armed bandit setting, where a principal (e.g. a marketplace) attempts to maximize utility by incentivizing agents to explore arms other than the myopically preferred one. In their setting, the optimal amount is known to the system, and the main goal is to quantify the required payments to achieve an optimal policy with myopic agents. The idea of shaping demand through monetary incentives in the sharing economy has also been tested in practice. For example, (Singla et al. 2015) use monetary incentives to encourage users of bike sharing systems to return bikes at beneficial locations, making automatic offers through the bike sharing app.

In this context, an important question is what amounts a platform should offer to maximize its utility. (Singla et al. 2015) introduce a simple protocol for learning optimal incentives in the bike sharing system to make users switch from the preferred station to a more beneficial one, ignoring information about specific switches and additional context. Extending on these ideas, we explore a general online learning protocol for efficiently learning optimal incentives.

Our Contributions
We provide the following main contributions in this paper:

• **Structural information:** We consider structural information in user preferences to speed up learning of incentives, and provide a general framework to model structure across tasks via convex constraints. Our algorithm, Coordinated Online Learning (CoOL) is also of interest for related multi-task learning problems.

• **Computational efficiency:** We introduce two novel ideas of sporadic and approximate projections to increase the computational efficiency of our algorithm. We derive formal guarantees on the performance of the CoOL algorithm and achieve no-regret bounds in this setting.

• **User study on Airbnb:** We collect a unique data set through a user study with apartments on Airbnb and test the viability and benefit of the CoOL algorithm on this dataset.

---

1Work performed while at ETH Zurich.
Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

2Data from insideairbnb.com.
Preliminaries

In the following, we introduce the general problem setting of this paper.

Platform. We investigate a general platform in the sharing economy, such as the apartment rental marketplace Airbnb. On this platform, users can choose from a set of goods and services, denoted as items. A user that arrives at time $t$ chooses an item $i^t \in [n]$. If the user chooses to buy item $i^t$, the platform gains utility $u^t_i$.

Incentivizing exploration. The initial choice, item $i^t$, might not maximize the platform’s utility, and the platform might be interested in offering a different item $j^t$ with utility $u^t_j > u^t_i$ instead. For example, $j^t$ could represent an infrequently reviewed item that the platform wants to explore. To motivate the user to select item $j^t$ instead, the platform can offer an incentive $p^t$, for example in the form of a monetary discount on that item. The user can either accept or reject the offer $p^t$ depending on the private cost $c^t$, where the user accepts the offer if $p^t \geq c^t$ and rejects the offer otherwise. If the user accepts the offer, the utility gain of the platform is $u^t_j - u^t_i - p^t$.

Objective. In this setting, two tasks need to be optimized to achieve a high utility gain: finding good switches $i^t \rightarrow j^t$, and finding good incentives $p^t$. Good switches $i^t \rightarrow j^t$ are those in which the achievable utility gain is positive, i.e. $u^t_j - u^t_i - c^t > 0$. To realize a positive utility gain, the offer $p^t$ needs to be greater or equal to $c^t$, since otherwise the offer would be rejected.

In this paper, we focus on learning optimal incentives $p^t$ over time, while the platform chooses relevant switches $i^t \rightarrow j^t$ independently.

Methodology

In this section, we present our methodology for learning optimal incentives $p^t_{i,j}$ and start with a single pair of items $(i, j)$. We allow for natural constraints on $p^t_{i,j}$, such that $p^t_{i,j} \in S_{i,j}$, where $S_{i,j}$ is convex and non-empty. For example, $S_{i,j}$ might be lower-bounded by 0 and upper-bounded by the maximum discount that the platform is willing to offer.

Single Pair of Items

We consider the popular algorithmic framework of online convex programming (OCP) (Zinkevich 2003) to learn optimal incentives $p^t_{i,j}$ for a single pair of items. The OCP algorithm is a gradient-descent style algorithm that updates with an adaptive learning rate and performs a projection after every gradient step to maintain feasibility within the constraints $S_{i,j}$. We use $\tau_{i,j}^t$ to denote the number of times a pair of items $i, j$ has been observed and $\eta$ to denote the learning rate. To measure the performance of the algorithm, we use the loss $l^t(p^t_{i,j})$, which is the difference between the optimal prediction and the prediction provided by the algorithm, such that $l^t(p^t_{i,j}) = \mathbb{1}_{p^t \geq c^t} \cdot (p^t - c^t) + \mathbb{1}_{p^t < c^t} \cdot (u - c^t)$ for $u \geq c^t$, and $l^t(p^t) = 0$ for $u < c^t$.

\footnote{Note that this loss function is non-convex. A convex version is presented in the case study.}

Algorithm 1: OL – Online Learning

1. Input:
   - Learning rate constant: $\eta > 0$
2. Initialize: $p^0_{i,j} \in S, \tau^0_{i,j} = 0$
3. for $t = 1, 2, \ldots, T$
   4. Suffer loss $l^t(p^t_{i,j})$
   5. Calculate gradient $g^t_{i,j}$
   6. Set $\tau^t_{i,j} = \tau^t_{i,j} + 1$
   7. Update $p^t_{i,j} = p^t_{i,j} - \eta \frac{\sqrt{\tau^t_{i,j}}}{\gamma_{i,j}} g^t_{i,j}$

Multiple Pairs of Items

We now relax the assumption of a fixed pair of items and return to our original problem of learning optimal incentives for multiple pairs of items, i.e. the algorithm receives specific items $i^t$ and $j^t$ as input for each user. If we consider all items $n$ on the platform, the total number of pairs is $n^2 - n$.

For learning the optimal incentive for each pair of items, the algorithm maintains a specific learning rate proportional to $1/\sqrt{\tau_{i,j}}$ for each pair of items and performs one gradient update step using Algorithm 1. We refer to this straightforward adaptation of the OL algorithm as Independent Online Learning (IOL) and use this algorithm as a baseline for our analysis. Using regret bounds of (Zinkevich 2003) and denoting the number of pairs of items as $K$, we can upper bound the regret of IOL as

$$\text{Regret}_{IOL}(T) \leq \frac{3}{2} \sqrt{TK} \|S_{\max}\| \|g_{\max}\|.$$  (2)

Structural Information

In a real-world setting, incentives for different pairs of items typically are not independent, and in some cases, certain structural information may help to speed up learning of optimal incentives. In the following, we discuss several relevant types of structural information.

Independent learning. In this baseline setting each pair of items is learned individually. Thus, the number of incentives that need to be learned grows quadratically with the number of items on a platform. While applicable for a small number of items, this approach is not favored on typical platforms in the sharing economy.

Shared learning. Another commonly studied setting is shared learning. In this setting, all pairs of items are considered equivalent, and only one global incentive is learned. While allowing the platform to learn about many pairs of
items at the same time, this approach fails to consider natural asymmetries in the problem. For example, the required incentive for switching from \(i\) to \(j\) is often different than the required incentive for switching from \(j\) to \(i\), as can be also observed in the case study of this paper.

**Metric/hemimetric structure.** Assuming that the required incentives are related to the dissimilarity of items \(i\) and \(j\), metrics are a natural choice to model structural dependencies, as they capture the property of triangle inequalities in dissimilarity functions. However, incentives for pairs of items are not necessarily required to be symmetric. For example, the required incentives for switching from a highly reviewed apartment on Airbnb to one without reviews is likely higher than vice versa. Therefore, we use hemimetrics, which are a relaxed form of a metric that satisfy only non-negativity constraints and triangular inequalities, capturing asymmetries in preferences (cf. (Singla, Tschatschek, and Krause 2016)). The usefulness of the hemimetric structure for learning optimal incentives is demonstrated in the experiments.

In the following section, we introduce a general-purpose algorithm for learning with structural information, where the structure is defined by convex constraints on the solution space. The key idea of our algorithm is to coordinate between individual pairs of items by projecting onto the resulting convex set. We generalize our approach for contextual learning, where additional features, such as information about users, may be available. Since projecting onto convex sets may be computationally expensive, we further extend our analysis to allow projections to be sporadic (i.e. only after certain gradient steps) and approximate (i.e. with some error compared to the optimal projection).

**Learning with Structural Information**

We begin this section by introducing a general framework for specifying structural information via convex constraints. We denote each pair of items as a distinct problem \(z \in [K]\), where \(K\) is the total number of pairs of items. Each problem \(z\) may be associated with additional features, for example with information about the current user. As is common in online learning, we consider a \(d\) dimensional weight vector \(w_z\) for each problem \(z \in [K]\) for learning optimal incentives. The prediction \(p_z\) is equal to the inner product between \(w_z\) and the \(d\) dimensional feature vector. In the previous section, we described the special case with \(d = 1\) and a unit feature vector, such that \(w_z\) is equivalent to the prediction \(p_z\).

**Specifying Structure via Convex Constraints**

Similar to constraints on \(p_z\), we allow for convex constraints on \(w_z\), such that \(w_z \in S_z \subseteq \mathbb{R}^d\). We assume \(S_z\) is a convex, non-empty, and compact set, where \(\|S_z\|\) is the Euclidean norm of the solution space\(^3\). Further, we assume \(\|S_z\| \leq \|S_{\text{max}}\|\) for some constant \(\|S_{\text{max}}\|\). We denote the joint solution space of the \(K\) problems as \(S = S_1 \times \cdots \times S_z \times \cdots \times S_K \subseteq \mathbb{R}^{dK}\) and define \(w^t \in S\) as the concatenation of the problem specific weight vectors at time \(t\), i.e.

\[
\begin{equation}
\tilde{w}^t = [(w_1^t)' \cdots (w_k^t)']'.
\end{equation}
\]

The available structural information is modelled by a set of convex constraints, such that the joint competing weight vector \(w^*\), against which the loss at each round is measured, lies in a convex, non-empty, and closed set \(S^* \subseteq S\), representing a restricted joint solution space, i.e. \(w^* \in S^*\). In the following, we provide several practical examples of how \(S^*\) can be defined.

**Independent learning.** \(S^* \equiv S\) models the setting where the problems are unrelated/independent.

**Shared learning.** A shared parameter setting can be modeled as

\[
S^* = \{w^* \in S | w^*_1 = \cdots = w^*_z = \cdots = w^*_K\}.
\]

Instead of sharing all parameters, another common scenario is to share only a few parameters. For a given \(d' \leq d\), sharing \(d'\) parameters across the problems can be modeled as

\[
S^* = \{w^* \in S | w^*_1[1:d'] = \cdots = w^*_z[1:d'] = \cdots = w^*_K[1:d']\}
\]

where \(w^*_z[1:d']\) denotes the first \(d'\) entries in \(w^*_z\). This approach is useful for sharing certain parameters that do not depend on the specific problem. For example, in the case of apartments on Airbnb, a shared feature could be the distance between apartments.

**Hemimetric structure.** To model dissimilarities between items for learning optimal incentives, we use the hemimetric set. Specifically, we use \(r\)-bounded hemimetrics, which, next to non-negativity constraints and triangular inequalities, also include non-negativity and upper bound constraints. For \(d = 1\), the convex set representing \(r\)-bounded hemimetrics is given by \(S^* = \{w^* \in S | w^*_{i,j} \in [0,r], w^*_{i,j} \leq w^*_{i,k} + w^*_{k,j} \forall i, j, k \in [n]\}\)

**Our Algorithm**

In the following, we introduce our algorithm, Coordinated Online Learning (CoOL).  

**Exploiting Structure via Weighted Projections.** The CoOL algorithm exploits structural information in a principled way by performing weighted projections to \(S^*\), with weights for a problem \(z\) proportional to \(\sqrt{\tau_z}\). Intuitively, the weights allow us to learn about problems that have been observed infrequently while avoiding to “unlearn” problems that have been observed more frequently. A formal justification for using weighted projections is provided in the extended version of this paper (Hirnschall et al. 2018).

We define \(Q^t\) as a square diagonal matrix of size \(dK\) with each \(\sqrt{\tau_z}\) represented \(d\) times. In the one-dimensional case \((d = 1)\), we can write \(Q^t\) as

\[
Q^t = [\sqrt{\tau_1} \cdots 0 \cdots \sqrt{\tau_K}].
\]

Using \(\tilde{w}\) to jointly represent the current weight vectors of all the learners at time \(t\) (cf. Line 7 in Algorithm 2), we
Algorithm 2: CoOL – Coordinated Online Learning

1 Input:
   • Projection steps: \((\xi^t)_{t \in [T]}\) where \(\xi^t \in \{0, 1\}\)
   • Projection accuracy: \((\delta^t)_{t \in [T]}\) where \(\delta^t \geq 0\)
   • Learning rate constant: \(\eta > 0\)

2 Initialize: \(w^1_z \in S_z, \tau^0_z = 0\)

3 for \(t = 1, 2, \ldots, T\) do
   4 Suffer loss \(l^t(\bar{w}_z)\)
   5 Calculate (sub-)gradient \(g^t_z\)
   6 Set \(\tau^t_z = \tau^{t-1}_z + 1\)
   7 Update \(\bar{w}^{t+1}_z = w^t_z, \tilde{w}^{t+1}_z = w^t_z - \frac{\eta}{\sqrt{\tau^t_z}}g^t_z\)
   8 if \(\xi^t = 1\) then
      9 Define \(Q^t\) as per Equation (3)
      10 Compute \(w^{t+1} = AProj(\tilde{w}^{t+1}_z, \delta^t, Q^t)\)
   else
      11 \(w^{t+1} = \arg\min_{w \in S_z} \|w - \tilde{w}^{t+1}_z\|_2\)
   end

end

compute the new joint weight vector \(w^{t+1}\) (cf. Line 10 in Algorithm 2) by projecting onto \(S^*\), using
\[
\tilde{w}^{t+1}_z = \arg\min_{w \in S^*} \|w - \bar{w}^{t+1}_z\|_2^2 \quad (4)
\]

We refer to this as the weighted projection onto \(S^*\). Since \(S^*\) is convex and the projection is a special case of the Bregman projection, the projection onto \(S^*\) is unique (cf. (Cesa-Bianchi and Lugosi 2006; Rakhlin and Tewari 2009)).

**Sporadic and Approximate Projections.** For large scale applications (i.e. large \(K\) or large \(d\)), projecting at every step could be computationally very expensive: a projection onto a generic convex set \(S^*\) would require solving a quadratic program of dimension \(dK\). To allow for computationally efficient updates, we introduce two novel algorithmic ideas: **sporadic** and approximate projections, defined by the above-mentioned sequences \((\xi^t)_{t \in [T]}\) and \((\delta^t)_{t \in [T]}\). Here, \(\delta^t\) denotes the desired accuracy at time \(t\) and is given as input to Function 3, AProj, for computing approximate projections. This way, the accuracy can be efficiently controlled using the duality gap of the projections. As we shall see in our experimental results, these two algorithmic ideas of sporadic and approximate projections allow us to speed up the algorithm by an order of magnitude while retaining the improvements obtained through the projections.

Algorithm 2, when invoked with \(\xi^t = 1, \delta^t = 0\ \forall t \in [T]\), corresponds to a variant of our algorithm with exact projections at every time step. When invoked with \(\xi^t = 0 \forall t \in [T]\), our algorithm corresponds to the IOL baseline.

**Relation to existing approaches.** A related algorithm is the AdaGrad algorithm (Duchi, Hazan, and Singer 2011), which uses the sum of the magnitudes of past gradients to determine the learning rate at each time \(t\), where larger past gradients correspond to smaller learning rates. A key difference to the CoOL algorithm is that the AdaGrad algorithm

Function 3: AProj – Approximate Projection

1 Input: \(\bar{w}, \delta^t, Q^t\)
2 Define \(f^t(w) = (w - \bar{w})'Q^t(w - \bar{w})\) for \(w \in S\)
3 Choose \(w^{t+1} \in \{w \in S^* : f^t(w) - \min_{w' \in S^*} f^t(w') \leq \delta^t\}\)
4 Return: \(w^{t+1}\)

enforces exact projections after every iteration. This is particularly problematic for large, complex structures since projections on these structures often rely on numeric approximations, that may not guarantee to converge to the exact solution in finite time.

**Performance Guarantees and Analysis**

In this section, we analyze worst-case regret bounds of the CoOL algorithm against a competing weight vector \(w^* \in S^*\). The proofs are provided in the extended version of this paper (Hirnschall et al. 2018).

**General Bounds**

We begin with a general result, without assumptions on the projection accuracy and rate.

**Theorem 1.** The regret of the CoOL algorithm is bounded by \(\text{Regret}_{CoOL}(T) \leq\)

\[
\frac{1}{2\eta} \left( \frac{\|S_{\max}\|}{\sqrt{T K}} + 2\eta \|g_{\max}\| \right)^2 \sqrt{T K} \quad (R1)
\]

\[
+ \sum_{t=1}^{T} \mathbb{I}_{\{-\xi^t-1\} \land (\xi^t)} \|S_{\max}\| \|g_{\max}\| \quad (R2)
\]

\[
+ \frac{1}{\eta} \sum_{t=1}^{T} \mathbb{I}_{\{\xi^t\}} \left( \delta^t + \frac{\sqrt{2\delta^t(T K)^{1/4}}}{\|S_{\max}\|} \right) \quad (R3)
\]

\[
+ \frac{1}{2\eta} \|S_{\max}\|^2 - 2\eta \|g_{\max}\|^2 K \quad (R4)
\]

The regret in Theorem 1 has four components. R1 comes from the standard regret analysis in the OCP framework, R2 comes from sporadic projections, R3 comes from the allowed error in the projections, and R4 is a constant.

Note that when \(\xi^t = 0\) for all \(t\) (i.e. no projections are performed) and \(\eta\) is proportional to \(1/\sqrt{T K}\), we get the same regret bounds proportional to \(\sqrt{T}\) as for the IOL algorithm. This also reveals the worst-case nature of the regret bounds of Theorem 1, i.e. the proven bounds for CoOL are agnostic to the specific structure \(S^*\) and the order of task instances.

**Sporadic/Approximate Projection Bounds**

To provide specific bounds for the practically useful setting of sporadic and approximate projections, we introduce \(\alpha\) and \(\beta\) and the user chosen parameters \(c_\alpha\) and \(c_\beta\) to control the frequency and accuracy of the projections.
Figure 1: Simulation results for learning hemimetrics. (a,b,c) compare the performance of CoOL against IOL for different orders of problem instances. (d,e,f) show the speed/performance tradeoff using sporadic and approximate projections.

Corollary 1. Set \( \eta = \frac{1}{2} \frac{\|S_{\text{max}}\|}{\|g_{\text{max}}\|} \), \( \forall t \in [T] \), define

\[
\xi_t \sim \text{Bernoulli}(\alpha) \quad \text{with} \quad \alpha = \frac{c_\alpha}{\sqrt{T}},
\]

\[
\delta_t = c_\beta (1 - \beta)^2 \sqrt{K} \frac{\|S_{\text{max}}\|^2}{T}
\]

where constants \( c_\alpha \in [0, \sqrt{T}] \), \( c_\beta \geq 0 \), and \( \beta \in [0, 1] \). The expected regret (w.r.t. \( (\xi_t)_{t \in [T]} \)) of the CoOL algorithm is bounded by

\[
E[\text{Regret}_{\text{CoOL}}(T)] \leq 2\sqrt{TK} \|S_{\text{max}}\| \|g_{\text{max}}\| \left( 1 + \frac{c_\alpha}{2\sqrt{K}} \left( 1 - \frac{c_\alpha}{\sqrt{T}} \right) \right) + c_\alpha (c_\beta + \sqrt{2c_\beta}(1 - \beta)).
\]

As shown in Corollary 1, projections are required to be more accurate for higher values of \( t \). Intuitively, this is required so that already learned weights are not unlearned through inaccurate projections. Using the definitions under Corollary 1, we can prove worst-case regret bounds proportional to \( \sqrt{T} \) for this setting.

Performance Analysis for Hemimetric Structure

We now test the performance of the CoOL algorithm on synthetic data with an underlying hemimetric structure.

**Hemimetric projection.** To be able to perform weighted projections onto the hemimetric polytope, we use the metric nearness algorithm (Sra, Tropp, and Dhillon 2004) as a starting point. For our purposes, three modifications of the algorithm are required: First, we lift the requirement of symmetry to generalize from metrics to hemimetrics. Second, the metric nearness algorithm does not guarantee a solution in the metric set in finite time. However, to calculate the duality gap, the solution is required to be feasible. Thus, we apply the Floyd-Warshall algorithm (Floyd 1962) after every iteration to receive a solution in the hemimetric set. Third, we add weights to the triangle inequalities to allow for weighted projections and further add upper and lower bound constraints.

**Data structure.** To empirically test the performance of the CoOL algorithm on the hemimetric set, we synthetically generate data with \( d = 1 \) and model the underlying structure \( S^* \) as a set of \( r \)-bounded hemimetrics with \( n = 10 \), resulting in \( K = 90 \) problems. We use a simple underlying ground-truth hemimetric \( w^* \), where the \( n \) items belong to two equal-sized clusters, with \( p_{i,j}^* = 1 \) if \( i \) and \( j \) are from the same cluster and \( p_{i,j}^* = 9 \) otherwise. The results of our experiment in Figure 1 illustrate the potential runtime improvement using sporadic/approximate projections.
Random order of problems. Problem instances $z^t$ are chosen uniformly at random at every time step. The CoOL algorithm achieves a significantly lower regret than the IOL algorithm, benefiting from the weighted projections onto $S^*$. At $T = 500$, the regret of CoOL is less than half of that of the IOL, cf. Figure 1(a).

Batches of problems. In the batch setting, a problem instance is chosen uniformly at random, then it is repeated five times before choosing a new problem instance. Compared to the above-mentioned random order, the IOL algorithm suffers a lower regret because of a higher probability that problems are repeatedly shown. Furthermore, the benefit of the projections onto $S^*$ for the CoOL algorithm is reduced, cf. Figure 1(b), showing that the benefit of the projections depends on the specific order of the problem instances for a given structure.

Single-problem setting. A single problem $z$ is repeated in every round. As illustrated, in this case the IOL algorithm and the CoOL algorithm have the same regret, cf. Figure 1(c). In order to get a better understanding of using weights $Q^t$ for the weighted projection, we also show a variant uw-CoOL using $Q^t$ as identity matrix. Unweighted projection or using the wrong weights can hinder the convergence of the learners, as shown in Figure 1(c) for this extreme case of a single-problem setting.

Varying the rate of projection ($\alpha$). The regret of the CoOL algorithm monotonically increases as $\alpha$ decreases, and is equivalent to the regret of the IOL algorithm at $\alpha = 0$, cf. Figure 1(d). In the range of $\alpha$ values between 1 and 0.1, the regret of the CoOL algorithm is relatively constant and increases strongly only as $\alpha$ approaches 0. With $\alpha$ as low as 0.1, the regret of the CoOL algorithm in this setting is still almost half of that of the IOL algorithm.

Varying the accuracy of projection ($\beta$). The regret of the CoOL algorithm monotonically increases as $\beta$ decreases, and exceeds that of the IOL algorithm for values smaller than 0.65 because of high errors in the projections, cf. Figure 1(e). In the range of $\beta$ values between 1 and 0.85, the regret of the CoOL algorithm is relatively constant and less than half of that of the IOL algorithm.

Runtime vs. approximate projections. As expected, the runtime of the projection monotonically decreases as $\beta$ decreases, cf. Figure 1(f). For values of $\beta$ smaller than 0.95, the runtime of the projection is less than 10% of that of the exact projection. Thus, with $\beta$ values in the range of 0.85 to 0.95, the CoOL algorithm achieves the best of both worlds: the regret is significantly smaller than that of IOL, with an order of magnitude speed up in the runtime compared to exact projections.

Airbnb Case Study

To test the viability and benefit of the CoOL algorithm in a realistic setting, we conducted a user study with data from the marketplace Airbnb.

Experimental Setup

We use the following setup in our user study:

Airbnb dataset. Using data of Airbnb apartments from insideairbnb.com, we created a dataset of 20 apartments as follows: we chose apartments from 4 types in New York City by location (Manhattan or Brooklyn) and number of reviews (high, ≥ 20 or low, ≤ 2). From each type we chose 5 apartments, resulting in a total sample of $n = 20$ apartments.

Survey study on MTurk platform. In order to obtain real-world distributions of the users’ private costs, we collected data from Amazon’s Mechanical Turk marketplace. After several introductory questions about their preferences and familiarity with travel accommodations, participants were shown two randomly chosen apartments from the Airbnb dataset. To choose between the apartment, participants were given the price, location, picture, number of reviews and rating of each apartment, as shown in Figure 2. Participants were first asked to select their preferred choice between the two apartments. Next, they were asked to specify their private cost for choosing the other, less preferred apartment instead. The collected data from the responses consists of tuples $(i, j, c)$, where $i$ is the preferred choice, $j$ is the suggested alternative, and $c$ is the private cost of the user.

Sample. In total, we received 943 responses, as summarized in Table 1. The sample for the performance analysis of the CoOL algorithm consists of 323 responses, in which $i$ was a frequently reviewed apartment, $j$ an infrequently reviewed apartment, and participants were willing to explore

---

4Real pictures from Airbnb replaced with illustrative examples.
the infrequently reviewed apartment for a discount (i.e. they did not select NA).

**Utility gain.** The utility gain $u$ for getting a review for infrequently reviewed apartments is set to $u = 40$ in our experiments, based on referral discounts given by Airbnb in the past.

**Loss function.** As introduced in the methodology section, we require a convex version of the true loss function for our online learning framework, ideally acting as a surrogate of the true loss. Additionally, the gradient of the loss function needs to be calculated from the binary feedback of acceptance/rejection of the offers. However, in the analyzed model with binary feedback, a loss function that satisfies both requirements cannot be constructed. Instead, we consider a simplified piece-wise linear convex loss function given by

$$l_t(p_t) = \begin{cases} 0 & \text{if } p_t \geq c_t \cr \frac{u}{2} & \text{if } p_t < c_t \end{cases} \cdot (c_t - p_t),$$

where $\frac{u}{2}$ denotes the magnitude of the gradient when a user rejects the offer. For the experiment, we use a delta value of 20. Due to this transformation, we use the utility gain rather than the loss as a useful measure of the performance of the CoOL algorithm.

**Structure.** Due to the small number of apartments, we consider a non-contextual setting with $d = 1$ and use an $r$-bounded hemimetric structure to model the relationship of the tasks, where $r$ is set to 40 to avoid recommending incentives $p_t > u$. Using a setting with $d > 1$ would allow for real-world applications with additional context.

**Main Results**

We now present and discuss the results of the user study.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$j$</th>
<th>Responses</th>
<th>Accepted</th>
<th>Avg. Discount</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Low</td>
<td>416</td>
<td>77.6%</td>
<td>29.5$</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
<td>228</td>
<td>83.3%</td>
<td>28.1$</td>
</tr>
<tr>
<td>High</td>
<td>High</td>
<td>219</td>
<td>82.2%</td>
<td>25.4$</td>
</tr>
<tr>
<td>Low</td>
<td>High</td>
<td>80</td>
<td>81.3%</td>
<td>25.9$</td>
</tr>
</tbody>
</table>

Table 1: Responses for different apartment types.

**Descriptive statistics.** Out of all responses, 758 (80.4%) respondents were willing to accept an offer for their less preferred apartment, given a certain discount per night. Out of these respondents, the average required discount for accepting the alternative apartment was 27.9 USD per night. The average required discount for switching from a frequently reviewed apartment to an infrequently reviewed apartment was 6% higher.

Out of the respondents who could choose between a frequently and an infrequently reviewed apartment, 83.9% respondents chose the frequently reviewed apartment, while only 16.1% respondents chose the infrequently reviewed apartment.

In the responses to an open question about the factors respondents considered to decide on the discount, we captured the frequency at which different factors were mentioned by defining several keywords for each factor. The number of times each factor was mentioned is shown in Table 2.

**Algorithm performance.** We use the cumulative utility gain to measure the performance of the IOL and the CoOL algorithm. The utility gains of both algorithms after 323 responses are shown in Figure 3(c). The utility gain in Figure 3(b) is almost 50% higher for the CoOL algorithm than for the IOL algorithm. Figure 3(c) reveals that this gain is mainly achieved due to a significant speed up in learning over the first 50 problems.

**Discussion.** The results of the user study confirm several findings of (Fradkin 2014), who studied the booking behavior on Airbnb. Similar to this study, we find that apartments with a high number of reviews are significantly more likely...
to be selected. We also find that the average required dis-
count per night is higher when the alternative choice is an
infrequently reviewed apartment. This also points toward a
difference in willingness to pay between frequently and in-
frequently reviewed apartments. Similar results have been
found in earlier studies on other marketplaces (Resnick et
al. 2006; Ye, Law, and Gu 2009; Luca 2011).

The user study also confirms that incentives influence
buying behavior and can help increase exploration on on-
line marketplaces (Avery, Resnick, and Zeckhauser 1999;
Robinson, Nightingale, and Mongrain 2012); when respond-
ents chose a frequently reviewed apartment and were
asked to instead choose an infrequently reviewed apartment,
77.6% of respondents were willing to accept a sufficiently
large offer. More than 10% of those respondents were will-
ing to accept a discount of 10 USD per night or less.

The performance of the IOL and CoOL algorithm in Fig-
ures 3(b) and 3(c) suggests that incentives can be learned via
online learning, and that structural information can be used
to significantly speed up the learning. Further, the speed up
in learning directly increases the marketplace’s utility gain
from suggesting alternative items. To reduce the problem
size on a real-world application such as Airbnb, items could
be grouped by features such as location or number of re-
views. Further, problem-specific features, such as the dis-
tance between apartments could be added to increase the ac-
curacy of the prediction.

Related Work
Multi-armed bandit / Bayesian games. A related path of
research are multi-armed bandit and Bayesian games, where
a principal attempts to coordinate agents to maximize its
utility. Research in this area mainly focuses on changing
the behavior of agents in the way information is disclosed,
rather than through provision of payments. (Kremer, Man-
sour, and Perry 2014) provide optimal information disclo-
sure policies for deterministic utilities and only two possible
actions. (Mansour, Slivkins, and Syrgkanis 2015) general-
ize the results for stochastic utilities and a constant number
of actions. Further, (Mansour et al. 2016) consider the in-
teraction of multiple agents, and (Chakraborty et al. 2017)
analyze a multi-armed bandits in the presence of communi-
cation costs. Our problem is different to previous research in
that utilities are not required to be stochastic, and additional
structural information is available to the principal.

Recommender systems. A different approach to encour-
gaging exploration in online marketplaces are recommender
systems, which are known to influence buyers’ purchasing
decisions and can be used to encourage exploration (Resnick
and Varian 1997; Senecal and Nantel 2004). For example, $\epsilon$-
greedy recommender systems recommend a product closest
to the buyer’s preferences with probability $(1 - \epsilon)$ and a ran-
dom product with probability $\epsilon$ (Ten Hagen, Van Someren,
and Hollink 2003). Such recommender systems can be ex-
tended using ideas studied in this paper.

Online/distributed multi-task learning. Multi-task
learning has been increasingly studied in online and dis-
tributed settings recently. Inspired by wearable computing,
a recent work by (Jin et al. 2015) studied online multi-task
learning in a distributed setting. They considered a setup,
where tasks arrive asynchronously, and the relatedness
among the tasks is maintained via a correlation matrix.
However, there is no theoretical analysis on the regret
bounds for the proposed algorithms. (Wang, Kolar, and
Storero 2016) recently studied the multi-task learning for
distributed LASSO with shared support. Their work is dif-
ferent from ours — we consider general convex constraints
to model task relationships and consider the adversarial
online regret minimization framework.

Conclusions and Future Work
We highlighted the need in the sharing economy to actively
shape demand by incentivizing users to differ from their pre-
ferred choices and explore different options instead. To learn
the incentives users require to choose different items, we
developed a novel algorithm, CoOL, which uses structural
information in user preferences to speed up learning. The
key idea of our algorithm is to exploit structural information
in a computationally efficient way by performing sporadic
and approximate projections. We formally derived no-regret
bounds for the CoOL algorithm and provided evidence for
the increase in performance over the IOL baseline through
several experiments. In a user study with apartments from
the rental marketplace Airbnb, we demonstrated the practi-
cal applicability of our approach in a real-world setting. To
conclude, we discuss several additional considerations for
offering incentives in a sharing economy platform.

Safety/individual consumer loss. Generally, exploration
in the sharing economy may be risky, and individuals can
face severe losses while exploring. For example, new hosts
might not be trustworthy, and new drivers in ridesharing sys-
tems might not be reliable. In our approach, the items to be
explored are controlled by the platform, and appropriate pre-
conditions would need to be implemented to minimize risks.

Reliability/Consistency. In order for platforms to imple-
ment an algorithmic provision of monetary incentives, it is
important that incentives are reliable and consistent over
time. Ideally, similar users should receive similar incentives,
and offers should be consistent with the user’s preferences.
Using the CoOL algorithm, consistency can be controlled
through appropriate convex constraints.

Strategy-proofness. Providing monetary incentives
based on user preferences creates possibilities for opportu-
tunistic behavior. For example, users could attempt to
repeatedly decline offers to receive higher offers in the
future or browse certain items hoping to receive offers for
similar items. To control for such behavior, markets need
to be large enough so that behavior of individuals does not
affect overall learning. Further, platforms can control the
number and frequency with which individual users receive
offers to minimize opportunistic possibilities.

Acknowledgments
This work was supported in part by the Swiss National Sci-
ence Foundation, and Nano-Tera.ch program as part of the
Opensense II project, ERC StG 307036, and a Microsoft Re-
search Faculty Fellowship. Adish Singla acknowledges sup-
port by a Facebook Graduate Fellowship.
References
Jin, X.; Luo, P.; Zhuang, F.; He, J.; and He, Q. 2015. Collaborating between local and global learning for distributed online multiple tasks. In CIKM.
Singla, A.; Santoni, M.; Bartók, G.; Mekerji, P.; Meenen, M.; and Krause, A. 2015. Incentivizing users for balancing bike sharing systems. In AAAI.
Singla, A.; Tschiatschek, S.; and Krause, A. 2016. Actively learning hemimetrics with applications to eliciting user preferences. In ICML.