Reinforcement mechanism design for fraudulent behaviour in e-commerce*

Qingpeng Cai
IIIS, Tsinghua University, China
cqp14@mails.tsinghua.edu.cn

Aris Filos-Ratsikas
University of Oxford, UK
Aris.Filos-Ratsikas@cs.ox.ac.uk

Pingzhong Tang
IIIS, Tsinghua University, China
kenshinping@gmail.com

Yiwei Zhang
UC Berkeley, USA
zhangyiwei1234567@126.com

Abstract

In large e-commerce websites, sellers have been observed to engage in fraudulent behaviour, faking historical transactions in order to receive favourable treatment from the platforms, specifically through the allocation of additional buyer impressions which results in higher revenue for them, but not for the system as a whole. This emergent phenomenon has attracted considerable attention, with previous approaches focusing on trying to detect illicit practices and to punish the miscreants.

In this paper, we employ the principles of reinforcement mechanism design, a framework that combines the fundamental goals of classical mechanism design, i.e. the consideration of agents' incentives and their alignment with the objectives of the designer, with deep reinforcement learning for optimizing the performance based on these incentives. In particular, first we set up a deep-learning framework for predicting the sellers' rationality, based on real data from any allocation algorithm. We use data from one of largest e-commerce platforms worldwide and train a neural network model to predict the extent to which the sellers will engage in fraudulent behaviour. Using this rationality model, we employ an algorithm based on deep reinforcement learning to optimize the objectives and compare its performance against several natural heuristics, including the platform's implementation and incentive-based mechanisms from the related literature.

Introduction

One of the most fundamental optimization challenges in e-commerce websites is the allocation of impressions to the sellers who offer their products on the platform. When a user enters a keyword in the search field, a list of sellers with related products is displayed to the user. But how should one choose which sellers to be displayed or equivalently, how should the platform divide millions of user impressions to sellers with products that match the keyword?

Most major websites, such as Taobao, Amazon or eBay, incorporate reputation systems (Jurca and Faltings 2007) for allocating buyer impressions. Sellers with higher reputations, typically expressed through higher conversion rates (the number of transactions that a seller can carry out given one unit of buyer impressions) and higher overall transaction numbers, will normally be rewarded with higher chances of showing up in buyers' recommendation lists. As a result, sellers with higher reputations are usually rewarded with more buyer impressions, a process which is implemented implicitly via the employment of an impression allocation algorithm. The intention of such reputation schemes is to bring the most capable sellers forward, ultimately generating more revenue in the long run.

However, knowing that all these allocation algorithms are heavily based on historical transaction records, sellers have been known to manipulate the algorithms by faking the number of transactions using various illicit ways. In fact, there has even been an emerging underground industry that provides sophisticated services for the sellers who want to quickly increase the number of historical transactions. Xu et al. (2015) refer to such enterprises as *seller-reputation-escalation (SRE) markets*. Attesting to the severity of the problem are the several lawsuits by Amazon against sellers that were allegedly using fake reviews to boost their profits, the most recent of which was against more than 1000 sellers in 2016 (e.g. see (Techcrunch 2016)).

Detecting such behaviour is usually achieved by a combination of machine learning techniques (Mukherjee *et al.* 2013; Jindal and Liu 2008; Yoo and Gretzel 2009; Hooi *et al.* 2016; Schoenebeck *et al.* 2016), as well as manual labour, to minimize the number of "false positives", i.e. the chances of penalizing honest sellers (Ott *et al.* 2011). This is achievable in a smaller scale, but becomes increasingly more difficult as the number of sellers grows very large.

An alternative approach that has been proposed recently (Cai *et al.* 2016) is to use the principles of *mechanism design* to prevent such fraudulent behaviour. The idea is to design the allocation algorithms in a way that does not incentivize the sellers to resort to illicit means of boosting their reputations. Specifically, Cai et al. (2016) model the problem as an instance of truthful resource allocation (Shoham and Leyton-Brown 2009, Chapter 11), in which sellers aim to maximize their *reputation scores* given an associated cost for eliciting fake reviews and transactions and design mechanisms for which it is a *dominant strategy* to act honestly.

*Qingpeng Cai and Pingzhong Tang were supported in part by the National Natural Science Foundation of China Grant 61303077, 61561146398, a Tsinghua University Initiative Scientific Research Grant, a China Youth 1000-talent program and Alibaba Innovative Research program. Aris Filos-Ratsikas was supported by the ERC Advanced Grant 321171 (ALGAME).

However, the applicability of such an approach relies on quite strong informational and cognitive assumptions. On one hand, the costs of sellers for faking transactions are assumed to be public information and are used to design the mechanisms. This is not always realistic as one would conceivably have to engage in such illicit behaviour themselves to actually get an estimate of the average costs and even in that case, the actual costs might defer depending on the capabilities of each seller. On the other hand, the assumption (Cai *et al.* 2016) is that of classical mechanism design (Maskin 2008), i.e. that the participants are perfectly rational and that they fully understand how the mechanisms operate. As it has often been pointed out however (Rubinstein 1998; Simon 1957), in reality people are often bounded by either computational constraints or even cognitive limitations, especially in large and dynamically evolving systems like those of e-commerce applications. Additionally, it is often the case that sellers do not know the allocation mechanism that the platform uses; they rather observe their gains or costs by faking different numbers of transactions and they adjust their actions accordingly. From the discussion above, it is clear that the potential remedies to the problem are situated at the two ends of the spectrum; either detecting and punishing illicit behaviour *ex-post* or trying to prevent such practices altogether, under often unrealistic assumptions.

To achieve some middle ground and obtain more realistic solutions, we employ the recently proposed agenda of *reinforcement mechanism design* (Tang 2017; Cai *et al.* 2017), which constitutes a general framework for solving large scale problems in dynamic environments like e-commerce websites by combining the principles of mechanism design with deep reinforcement learning. Specifically, assuming some rationality model for the sellers (e.g. some variant of no-regret learning (Cai *et al.* 2017; Nekipelov *et al.* 2015)), the framework advocates the design of deep reinforcement learning algorithms that take this rationality model into account and optimize some objective function. In simple words, we expect sellers to be strategic, but we take this behaviour into account when designing our algorithms.

Our Contributions

In this paper, we build upon the principles of reinforcement mechanism design (Tang 2017) and employ reinforcement learning for allocating buyer impressions to sellers in large e-commerce websites, taking the strategic nature of the sellers into account. However, instead of assuming a generic unsupervised learning-based rationality model, we take a more structured approach and employ the idea of *inferring and predicting the rationality of sellers from data*. Intuitively, given a dataset from an e-commerce platform that runs an allocation algorithm, we can train a neural network to predict the decisions of the sellers in terms of how many orders they elect to fake.

More concretely, we propose the following framework to obtain a *behavioural model of sellers* that can be used to predict the number of transactions that each seller will fake in the next time period. Given a set of data that records the transactions (real and detected fake) of sellers for a specific (perhaps unknown) allocation implementation in an e-commerce platform, we train a neural network to predict the actions of the sellers. Note that while the detection process can be resource-consuming, it can usually be done for a reasonably large amount of sellers, which constitutes the training set of the network. Using this framework we can:

- Evaluate the performance of the allocation algorithm in terms of the volume of real and fake transactions and design new algorithms based on this behavioural model.
- Evaluate the loss in performance of a new allocation algorithm due to the adjustment period for the sellers, in which they are trying to "learn" the new algorithm and find their optimal strategies.

The first part can be seen as a "repeated evolution" process in which the rationality of sellers is predicted, a new allocation algorithm is developed based on this prediction with performance considerations in mind and then, after sufficient data is collected, a new behavioural model is developed that gives rise to new predictions, with comparisons between the implemented algorithms being made in every step. The second part becomes especially relevant together with rationality assumptions, as it allows us to quantify the performance loss during the migration period between the currently implemented allocation algorithm of the platform and the mechanism to be used in the future.

Performance of heuristics. For the performance evaluation, first we consider three natural heuristics, one of a greedy nature and two that have been presented in the related literature (Cai *et al.* 2016) and would be truthful under perfect rationality and informational assumptions. Interestingly, the comparison between them reveals that according to our predicted rationality model, sellers tend to fake transactions in all algorithms, but the extent to which they fake drastically decreases for the latter allocation algorithms. In addition, the greedy algorithm is superior to the others, including the platform's current implementation.

Performance of deep reinforcement learning. Then, we prove that the employment of deep reinforcement learning allows us to achieve improved results in terms of performance. In particular an implementation of the Deep Deterministic Policy Gradient (DDPG) algorithm of (Lillicrap *et al.* 2015) outperforms all the different heuristics, as well as the algorithm in use by the platform in terms of the total volume of generated revenue. Note that in order to solve the impression allocation model problem using deep reinforcement learning, we first model it as a Markov Decision Process (similarly to (Cai *et al.* 2017)) which naturally has very large (or even continuous) state and action spaces and therefore the DDPG algorithm is a fitting choice, as it has been designed for problems of this nature.

Finally, we remark that in parallel to this work, Cai et al. (2017) apply a similar reinforcement mechanism design framework to solve the impression allocation problem, but our contribution is fundamentally different from theirs in two ways. First, we are concerned with the problem of faking transactions and fraudulent seller behaviour

whereas their setting considers sellers who strategize with their choice of prices, which is inherently different. Secondly, the authors in (Cai *et al.* 2017) model seller rationality as a low-regret type strategy whereas crucially, we infer the rationality in a more structured manner using real data.

The setting

In the setting of an e-commerce website, there are m sellers, each of which controls a product.* In each day t, there are n^t *buyer impressions* to be allocated to these sellers; a buyer impression means that the seller will be shown on the buyers' screen when looking for a specific product. The ability of a seller to facilitate a transaction given an impression is captured by the *conversion rate*; $cr_i(t)$ denotes the conversion rate of seller i on day t and is a random variable that indicates the number of real transactions that seller i will make given one buyer impression. On each day t, each seller i chooses a price $p_i(t)$ for its product as well as the number of transactions to fake denoted by $a_i(t)$. These daily choices of the seller are based on its history $H_i(t) = (n_i^t, r_i^t, a_i^t, p_i^t)$. on day t which is a vector consisting of:

- The number of impressions that each seller i is allocated before day t, denoted by $n_i^t = (n_i(1), \ldots, n_i(t-1))$, where $n_i(t)$ is the number of impressions allocated to seller i on day t.
- The number of real transactions that seller i made from the beginning until day t, denoted by $r_i^t = (r_i(1), \ldots, r_i(t-1))$, where $r_i(t)$ is the number of real transactions of seller i on day t. Note that the number of real transactions $r_i(t)$ on any day t is a random variable defined by $\{\min(n_i(t), c_i) \cdot cr_i(t)\}$, where c_i is an upper bound on the number of impressions that a seller might get allocated, imposed by the allocation algorithm.
- The number of fake transactions that seller i made from the beginning until day t, denoted by $a_i^t = (a_i(1), \ldots, a_i(t-1))$, where $a_i(t)$ is the number of fake transactions of seller i on day t.
- The vector of prices that buyer i selected in each round from the beginning until time t, denoted by $p_i^t = (p_i(1), \ldots, p_i(t-1))$.

Having access to the histories of the sellers, we can define their predicted rationality.

Definition 1 *The* behavioural model of the sellers *is a function f that inputs the history $H_i(t)$ of seller i before round t and returns the number of fake transactions of the seller at round t, i.e. $a_i(t) = f(H_i(t))$.*

While the sellers know how many transactions they have faked in each round, this information is not public and the algorithm designer can only hope to infer an estimation. We define the *record history* $R_i(t)$ of seller i before day t similarly to the definition of the history $H_i(t)$ but we use the combination of the number of transactions (real and fake) that the seller facilitated in the past, denoted by $v_i^t = (v_i(1), ..., v_i(t-1))$, with $v_i(j) = r_i(j) + a_i(j)$ instead, i.e. $R_i(t) = (v_i^t, n_i^t, p_i^t)$.

An *allocation algorithm* M is a function that inputs the record history of all sellers $R(t) = (R_1(t), ..., R_m(t))$ before round t and outputs the number of impressions allocated to each seller, i.e. $M(R(t)) = (n_1(t), \ldots, n_m(t))$. An allocation algorithm is *feasible* if the total number of impressions allocated to the sellers in any day t does not exceed the supply of impressions available on that day, i.e, $\forall t$ it holds that $\sum_{i=1}^m n_i(t) \leq n^t$.

The performance of an algorithm will be quantified by the *revenue* or *welfare* that it achieves (we will use the two terms interchangeably - the former being more intuitive but the latter being more consistent with the field of mechanism design). The revenue of an allocation algorithm M within T days is the revenue generated by the real transactions of all sellers on these days, i.e, $R(M, T) = \sum_{i=1}^m \sum_{t=1}^T r_i(t) p_i(t)$.

Inferring seller rationality from data

In this section, we will develop a procedure for establishing our behavioural model for the sellers. The general idea is the following. We make use of a large dataset of several characteristics of sellers, including the volume of real and fake transactions over a period of several months, under the allocation mechanisms of the platform (which are evolving and unknown). After some preprocessing of the data, to make them appropriate for classification, we design a neural network that we train using samples from this dataset. Our neural network allows us to obtain a prediction for the action of each seller on the next day, i.e. the number of transactions that it will elect to fake.

Dataset and preprocessing Our dataset is provided by one of largest e-commerce sites in China. The relational dataset contains a history of 50000 different sellers with different items in the past three months. Each record in the dataset contains a daily record of the number of buyer impressions that a seller received, the number of transactions (real and fake) in which the seller participated, the total recorded revenue of a seller (the number of all transactions involving the seller's product multiplied by its price) and the total fake revenue of the seller (defined similarly but only with respect to the fake transactions).† We also extract the features of the associated seller of each product, including the consignment rate, the rate of the good, the merchandise score average, the refund rate and the service rate. Because of the sparsity of the dataset with respect to the number of transactions, since many items do not participate in even a single transaction, we rebuild records filling the gaps with 0.

After careful inspection of the dataset, we use sampling on the original data to exclude the non-trivial parts that could hurt the prediction accuracy. For example, we exclude records that do not include the sellers' features and

*Or multiple products, but each product is treated individually in our setting.

†The number of fake transactions is estimated by counting the number of transactions which are related to buyers who are always penalized by the platform's detection team, and is quite reliable since the original recommendation system does not use this data at all.

more importantly, those records that correspond to products that were not sold even once during the three-month period; those data points compose nearly 1/3 of the whole dataset. The records associated with these products are not useful for training, since the data contains no information. We also observe that a portion of the dataset contains a rather large amount of detected fake transactions. After tracing those entries, they turn out to be mostly e-ticket sellers that charge only 1 RMB (Chinese yuan, roughly equivalent to 0.15 USD) per transaction, or compensation items worth 0.99 RMB, on which the platform's detection system does not work properly. For this reason, we filter all products with price lower than 3 RMB to help eliminate such occurrences.[‡]

After this elimination process, we randomly sample 64 continuous days from the remaining data five times, and add up a 4 days' total amount of fake transactions after 64 days as a prediction target.[§] To avoid the effects of imbalances on the classification, we sample 10.000 positive and 10.000 negative data points from the whole item database, where a point is positive if it has non-zero fake transactions for its prediction. Such balancing of datasets is common in the literature, to avoid classification inaccuracies (Chawla *et al.* 2002; Kotsiantis *et al.* 2006). For those two types of points, we use 16,000 points for training and 4,000 points for validation. Given the input of 64 days of training samples and predicted target t output by the network, the number of fake transactions a seller carries out on the next day is $t/4$.

Neural Network Structure We have tested two types of neural networks, using conventional *Convolutional layers* and *ResNet blocks*. As general techniques for boosting, the activation function is set as Rectified Linear Unit (ReLU) and a dropout rate (Srivastava *et al.* 2014) of 0.5 is enabled for fully-connected layers. The cross entropy loss function is used for the classification tasks and the squared loss is used for regression. Adam boost (Kingma and Ba 2014) is used for training with learning rate auto-adjusted according to the validation accuracy.

For the conventional Convolution network, the input product record tensor propagates through 6 convolutional layers, and 3 max pools of window size $(1, 2)$ are added for every two convolutional layers. For each such layer, the kernel size varies from $(1, 7)$ to $(1, 5)$ through $(1, 3)$, with each output channel size set as 32. We concatenate the output with 5 features and send them all to 2 fully-connected layers which are added, with output dimension 128, following a fully-connected layer with label size output, and finally a softmax layer for probability prediction. Similar deep convolutional neural networks on time series data like we have here were also used in (Yang *et al.* 2015).

The ResNet structure utilizes ResBlocks with the same implementation as in (He *et al.* 2015), each with 256 channels. We stack 10 ResBlocks on top of each other, and perform a max pooling on the channel for each 5 blocks. Finally, the output consists of a fully-connected layer along with a softmax output.

Table 1: Typical Performance for each structure

Network	classification	validation accuracy
3FC	two-class	0.5
Conv	two-class	0.82
ResNet	two-class	0.82
Conv-blind	two-class	0.8
ResNet-blind	two-class	0.8
ResNet-blind	three-class	0.79
ResNet-blind	four-class	0.76

For the sake of comparison, we have also implemented a naive 3 fully-connected layer to verify the effectiveness of our convolutional network and perform regression with these networks. We transform the prediction target y from a specific label to the value $\ln(1 + y)$ for regression using squared loss. The intuition for the regressor is the following: we hope it may work on small cases as accurately as possible, but for much bigger cases, knowing an amount roughly is enough for further treatment. Interestingly, in a "blind" training setting where the input channel size is changed to 3, with the fake history data part intentionally excluded, Table 1 shows that the accuracy in fact does not decrease too much, which implies that the fraudulent sellers have discernible patterns compared to "clean" users.

We demonstrate the validation accuracy of the different networks in Table 1. With a two-class classification (whether the seller will perform fake transactions or not) the validation accuracy exceeds 0.8, compared against the 0.5 accuracy of a random guess. For more levels of classification (whether the seller fakes more or less than a certain amount of transactions), the accuracy is still relatively high. Note that the ResNet and the conventional Convolution networks achieve similar accuracy, which can be interpreted by the small dimension of the dataset, which does not allow for the deep neural network to be fully utilized.

Experimental Setup

The general process that we will adopt for the experiments is the following. Given the history of all the items within 64 days, including the number of transactions (real and fake), the number of impressions and an allocation algorithm, we generate data for each item and each seller.

In each day $64 + t$, with $1 \leq t \leq 20$, the generation[¶] process contains the following steps.

1. We set the total number of buyer impressions to the total number of buyer impressions of day $64 + t$ of the dataset.
2. For each seller i, we predict the number of fake transactions $a_i(t)$ by Convolution networks presented in last section, we estimate the price $p_i(t)$ of the associated product, as well as the seller's real conversion rate $cr_i(t)$ and the feigned conversion rate $fcr_i(t)$, i.e. the conversion rate calculated based on the number of total transactions (real and fake).

[‡]We have checked the total transaction volume of items of price less than 3RMB and their effect on the total revenue is negligible.

[§]64 is chosen because we have 3 months of data, and 64 is the closest power of 2 we can reach for convenience of the CNN layers.

[¶]20 is chosen relatively to 64 as we can then compare our results (64+20=84) to the real data (3 months) which helps with validation.

3. We run an allocation algorithm $\mathcal{A}$ to allocate the impressions to the sellers, and then sample the real transaction revenue of each seller using the real conversion rate, to calculate the real revenue of the algorithm.

Heuristic allocation algorithms

For the choice of $\mathcal{A}$, we will consider three different allocation algorithms, which are either intuitive heuristics or have been presented in the related literature. The first algorithm is of a greedy nature; at each round t, it simply sorts the sellers by order of feigned conversion rates and then greedily allocates the impressions based on this order.

Algorithm GREEDY *Sort sellers in decreasing order of* $fcr_i(t-1)$. *For* $i = 1, \ldots, m$ *according to that ordering, let* $n_i(t) = \min(c_i, n^t - \sum_{j=1}^{i-1} n_j(t))$.

The other two algorithms that we will consider (which we will refer to as CVR and MIXED respectively) were presented in (Cai *et al.* 2016), and were termed as "Mechanism 3" and "Mechanism 2" respectively. These algorithms are truthful under perfect rationality assumptions and assuming that the cost functions of the sellers are known to the designer. We will evaluate their performance in our seller rationality framework. In the interest of space, we will omit the descriptions of the algorithms here, the interested reader is referred to (Cai *et al.* 2016) for the definitions.

Note that the algorithms require knowledge of the sellers' prices. We explain how to estimate the prices from our dataset.

Estimator of prices: Although prices might be adjusted everyday, we observe that in our dataset, the prices do not fluctuate much. Therefore, after excluding such extreme cases, we can obtain a good estimation of each seller's price. We treat the price as a random variable drawn from a Gaussian distribution; we extract prices from the records containing the number and the turnover of transactions and we calculate the mean and the standard deviation. Then, we filter those items for which the standard deviation of the price is 0.5 times larger than the mean, which implies a huge price fluctuation. The quality of our estimation is verified in Figure 1. The revenue of the curve for each day is the sum of products of the number of real transactions and the estimated prices. As the figure shows, the revenue is really close to the one given by the real prices in each day. To be concise, we let "history" stand for the platform's allocation algorithm in all figures.

Experimental comparison of the algorithms

In this section, we evaluate and compare the performance of these different heuristic algorithms. We remark that in Algorithms CVR and MIXED, the number of buyer impressions is treated as a real number. For this reason, we transform those allocation vectors to a discretized allocation by randomly sampling impressions with the given probability scaled by the weight for each impression to obtain integer allocations. Additionally, we implement a "max-impression estimator" to avoid assigning more impressions that the seller limit

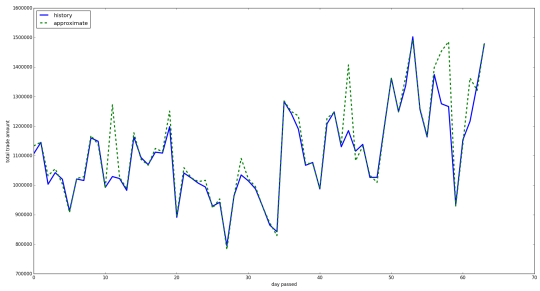

Figure 1: Comparison between real and estimated prices.

supply c_i allows. More specifically, the maximum number of impressions that each seller j can get in a round is $(\bar{n}_j^{65}) + \alpha\sigma(n_j^{65})$, where (i) n_j^{65} denotes the number of buyer impressions seller j gets in rounds $1, ..., 64$, (ii) $\bar{n}_j^{65}$ denotes the mean of the vector n_j^{65}, (iii) σ is the standard deviation of n_j^{65} and (iv) α is a positive unknown factor. The value of α controls the maximum number of impressions that each seller can be allocated.

We simulate GREEDY, CVR, MIXED and the *uniform algorithm*, in which each seller gets an even split of the total number of buyer impressions. We concentrate on two objectives that we care about in the process: the *real* total revenue (total number of transactions multiplied by prices), which is the main desideratum of e-commerce platforms and the revenue of fake transactions, which quantifies the degree to which sellers fake orders. We consider two cases:

1. We fix the value of α (at some reasonable value such as 2) and compare GREEDY, CVR, MIXED, the uniform algorithm and the algorithm that the platform employs.
2. We explore the effect of different values of α and compare the different algorithms.

For the first case, in order to perform an accurate comparison, we use the same total impression value and the same α in the simulation for all algorithms. We run the simulation for a period of 20 days and we evaluate the real revenue in Figure 2, which shows that CVR is just slightly better than the uniform algorithm, and both algorithms per-

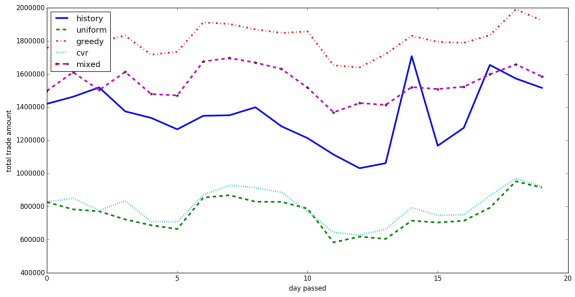

Figure 2: The real revenue per day.

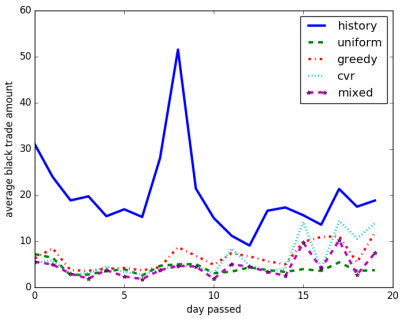

Figure 3: The fake revenue per day.

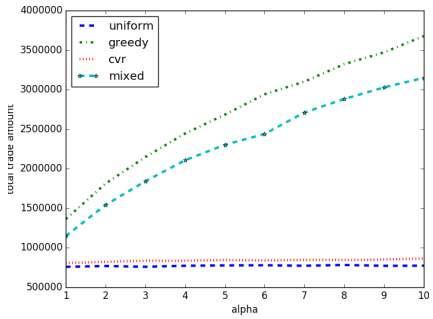

Figure 4: Average real revenue with varying α.

form worse than the platform's implementation. On the other hand, GREEDY and MIXED both outperform the platform's algorithm and GREEDY performs best among all algorithms.

As it can be seen in Figure 3, the revenue due to fake trades of all algorithms is lower than the platform's algorithm, while the uniform algorithm performs the best as expected, as the uniform algorithm maintains a low rate of the "black trade" transactions, since reputation is ignored in the allocation. From day 1 to day 10, CVR and MIXED perform better than GREEDY, which can be explained by the fact that they are designed to incorporate the rationality of the sellers and although the rationality here is imperfect, it is still better than not taking it at all into account. GREEDY also outperforms the platform's implementation for the fake revenue objective, since the parameter α restricts the largest number of buyer impressions each seller can get. Thus by choosing a small α like 2, GREEDY performs relatively well. In all, MIXED performs well across both objectives, while the platform's algorithm is outperformed by GREEDY and CVR.

When considering the effect of α, which acts as an implicit upper bound of the number of impressions that each seller receives, we can see how those algorithms actually perform as α changes. The results for the real revenue can be seen in Figure 4. GREEDY is mostly affected by the parameter; as α becomes larger, its real revenue increases quickly. For CVR and the uniform algorithm, α has almost no effect because the allocation rules of these algorithms are too proportional to achieve the upper limits on the number of buyer impressions. For MIXED, the revenue increases when α increases, because it follows a best-one-first strategy, and it also gives the sellers with lower conversion rates a chance to be displayed.

Optimizing by deep reinforcement learning

In this section, we explain how to employ deep reinforcement learning to obtain better performance. Given the behavioural model of the sellers, one can design an allocation algorithm to "fit" this behaviour by first modelling the problem appropriately as a Markov Decision Process (MDP) and then solving the MDP using deep reinforcement learning.

MDP formulation: For each day t we have a state $S_t = (v_1(t), n_1(t), p_1(t), ..., v_m(t), n_m(t), p_m(t))$, where for each seller $i = 1, \ldots, m$, $v_i(t)$ is the number of total transactions(real and fake) that seller i carries out, $n_i(t)$ is the number of buyer impressions that seller i gets allocated and $p_i(t)$ is the price that the seller selects. Additionally, for every possible allocation of the n^t impressions to the m sellers, we have a corresponding action. The payoff of an action from a state is the revenue of the corresponding impression allocation for that day, assuming a specific class for the seller.

Note that given that there are millions of buyer impressions to be allocated, the action space is very large and increases exponentially with the number of sellers; in fact if we model the impression allocation problem as division of a continuous resource[‖] similarly to (Cai *et al.* 2016; 2017), then the action space is actually continuous. For that reason, we will use an implementation of the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap *et al.* 2015), a deterministic actor-critic policy gradient algorithm, designed to handle continuous and high dimensional state and action spaces. We highlight the major implementation details below. For more information on actor-critic algorithms and deep reinforcement learning, the reader is referred to (Sutton and Barto 1998; Li 2017).

Training Setup: In the implementation of DDPG, the actor network uses four fully-connected layers with ReLU as the activation function and a softmax function at the output layer. The critic network inputs the (action, state) pair and outputs the estimation of the Q-value also with four fully-connected layers[**]. For different values of α ranging from 1 to 10, we train the DDPG algorithm separately. We use 1000 episodes with 1000 days in each episode for training and we randomly sample 500 sellers from the dataset. This is due to runtime limitations, as the computational overhead for more sellers increases drastically. The number is high enough to correspond to many settings of interest; impression allocation among 500 sellers is very common in smaller platforms but also in larger platforms for more specialized items of-

[‖]That is, to say, that a huge number of impressions can be sufficiently well approximated by a divisible impression unit.

[**]The original implementation of DDPG uses two-layer fully-connected networks in the actor and critic networks, but we find that more layers result in higher performance for our problem.

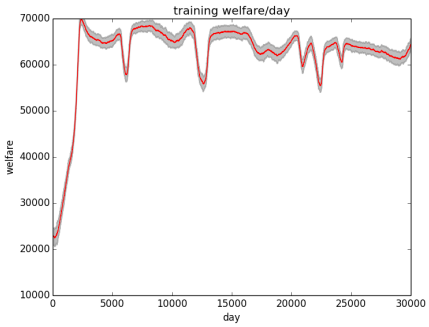

Figure 5: Revenue of training

fered by a smaller number of sellers.

The size of the replay buffer is 10^7, the discount factor is 0.99, and the rate of update of the target network is 10^{-3}. The actor network and the critic network are trained via the Adam algorithm (Kingma and Ba 2014) and the learning rates of these two networks are 10^{-3}. Following the same idea as in (Lillicrap *et al.* 2015), we add Gaussian noise to the action output by the actor network, with the mean of the noise decaying with the number of episodes for the exploration. We record the parameters and the revenue of training for each day and we set the parameters of the actor and the critic network as the parameters that achieve the maximum revenue at the training phase. Figure 5 shows the graph of the real revenue (welfare) of the algorithm with the number of days in the process of training with $\alpha = 2$; the gray band shows the variance in revenue.

Inferring the platform's implementation: As we have only 90 days of transaction data, if we aim to compare with the platform's algorithm for any number of days, we can not directly compare our algorithms with the results of the dataset, like we did in the previous section. For that reason, we will predict the algorithm employed by the platform using the data. We use the same dataset (except that the prediction target is set to the sum of number of impressions each seller gets allocated over 4 days) and the same structure of Convolutional networks that we did for predicting the number of fake transactions. We also apply the square loss; the square loss of the trained networks over testing data is 0.45. Using this network, one can predict the number of buyer impressions each seller will get in the platform's algorithm and then run the algorithm for any number of days.

Testing results: We test the DDPG algorithm, GREEDY, MIXED and the platform's inferred algorithm and evaluate and compare their performance in terms of the total revenue.[††] With the value of α ranging from 1 to 10, we test the algorithms for 500 sellers and 10000 days. Figure 6 illustrates the comparison between the revenue (welfare) of the algorithms as the number of days increases in testing with $\alpha = 2$. As we can see, the performance of DDPG is by far superior to any of the heuristics or the platform's algorithm.

[††]The performance of CVR and the uniform algorithm is inferior and is not shown.

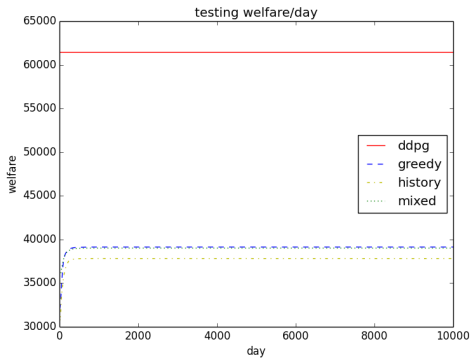

Figure 6: Testing revenue for $\alpha = 2$.

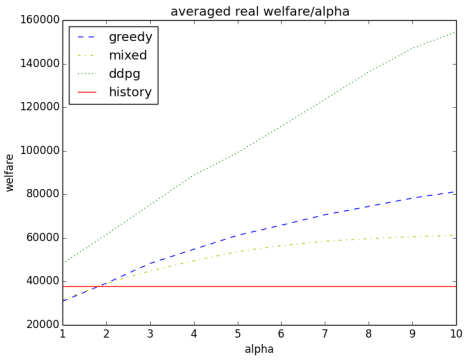

Figure 7: Testing average revenue for different values of α.

In Figure 7, we show the average revenue of the algorithms over different values of α. We can see that the performance of both heuristic algorithms and the DDPG algorithm increases as α increases and DDPG is mostly affected by the parameter in a positive way, since its performance increases drastically with the increase of α.

Conclusion

In this paper, we employed the principles of reinforcement mechanism design to tackle the problem of sellers' fraudulent behaviour in the impression allocation problem, one of the major challenges in e-commerce. Our contribution is two-fold; first, we design a rationality model for the sellers which infers their behaviour within a complex system from data and then we use this behavioural model to solve the problem efficiently using deep reinforcement learning.

Our "learning rationality from data"-based framework can in principle be used in other related problems in mechanism design for which we have sufficient data, substituting the perfect rationality assumptions or complementing the online learning approaches that have been proposed in the mechanism design literature more recently (Nekipelov *et al.* 2015); this would be worth exploring in the future.

References

Qingpeng Cai, Aris Filos-Ratsikas, Chang Liu, and Pingzhong Tang. Mechanism design for personalized recommender systems. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 159–166. ACM, 2016.

Qingpeng Cai, Aris Filos-Ratsikas, Pingzhong Tang, and Yiwei Zhang. Reinforcement mechanism design for e-commerce. *arXiv preprint arXiv:1708.07607*, 2017.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority oversampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 895–904. ACM, 2016.

Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230. ACM, 2008.

Radu Jurca and Boi Faltings. Obtaining reliable feedback for sanctioning reputation mechanisms. *Journal of Artificial Intelligence Research (JAIR)*, 29:391–419, 2007.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Eric S Maskin. Mechanism design: How to implement social goals. *The American Economic Review*, pages 567–576, 2008.

Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 632–640. ACM, 2013.

Denis Nekipelov, Vasilis Syrgkanis, and Eva Tardos. Econometrics for learning agents. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 1–18. ACM, 2015.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics, 2011.

Ariel Rubinstein. *Modeling bounded rationality*. MIT press, 1998.

Grant Schoenebeck, Aaron Snook, and Fang-Yi Yu. Sybil detection using latent network structure. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 739–756. ACM, 2016.

Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game theoretic and Logical Fundations.* Cambridge Uni. Press, 2009.

Herbert A Simon. Models of man; social and rational. 1957.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Pingzhong Tang. Reinforcement mechanism design. In *Early Carrer Highlights at Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI*, pages 5146–5150, 2017.

Techcrunch. Amazon cracks down on fake reviews with another lawsuit. `https://techcrunch.com/2016/04/26/amazon-cracks-down-on-fake-reviews\\-with-another-lawsuit`, 2016. [Online; posted 26-April-2016].

Haitao Xu, Daiping Liu, Haining Wang, and Angelos Stavrou. E-commerce reputation manipulation: The emergence of reputation-escalation-as-a-service. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1296–1306. International World Wide Web Conferences Steering Committee, 2015.

Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), Buenos Aires, Argentina*, pages 25–31, 2015.

Kyung-Hyan Yoo and Ulrike Gretzel. Comparison of deceptive and truthful travel reviews. *Information and communication technologies in tourism 2009*, pages 37–47, 2009.