Distributed Online Learning in Social Recommender Systems

Cem Tekin, Member, IEEE, Simpson Zhang, Mihaela van der Schaar, Fellow, IEEE

Abstract—In this paper, we consider decentralized sequential decision making in distributed online recommender systems, where items are recommended to users based on their search query as well as their specific background including history of bought items, gender and age, all of which comprise the context information of the user. In contrast to centralized recommender systems, in which there is a single centralized seller who has access to the complete inventory of items as well as the complete record of sales and user information, in decentralized recommender systems each seller/learner only has access to the inventory of items and user information for its own products and not the products and user information of other sellers, but can get commission if it sells an item of another seller. Therefore the sellers must distributedly find out for an incoming user which items to recommend (from the set of own items or items of another seller), in order to maximize the revenue from own sales and commissions. We formulate this problem as a cooperative contextual bandit problem, analytically bound the performance of the sellers compared to the best recommendation strategy given the complete realization of user arrivals and the inventory of items, as well as the context-dependent purchase probabilities of each item, and verify our results via numerical examples on a distributed data set adapted based on Amazon data. We evaluate the dependence of the performance of a seller on the inventory of items the seller has, the number of connections it has with the other sellers, and the commissions which the seller gets by selling items of other sellers to its users.

Index Terms—Multi-agent online learning, collaborative learning, distributed recommender systems, contextual bandits, regret.

I. INTRODUCTION

One of the most powerful benefits of a social network is the ability for cooperation and coordination on a large scale over a wide range of different agents [1]. By forming a network, agents are able to share information and opportunities in a mutually beneficial fashion. For example, companies can collaborate to sell products, charities can work together to raise money, and a group of workers can help each other search for jobs. Through such cooperation, agents are able to share the information that they learn over time about user’s background. By building upon the theory of contextual bandits, where the context is based on the user’s history, we produce a class of algorithms that allows agents to take near-optimal actions in the face of incomplete information, while still incentivizing them to fully cooperate within the network.

The framework we consider is very broad and applicable to a wide range of social networking situations. We analyze a group of agents that are connected together via a fixed network, each of whom experiences inflows of users to its page. Each time a user arrives, an agent chooses from among a set of items to offer to that user, and the user will either reject or accept each item. These items can represent a variety of things, from a good that the agent is trying to sell, to a cause that the agent is trying to promote, to a photo that the user is trying to circulate. In each application, the action of accepting or rejecting by the user will likewise have a distinct meaning. When choosing among the items to offer, the agent is uncertain about the user’s acceptance probability of each item, but the agent is able to observe specific background information about the user, such as the user’s gender, location, age, etc. Users with different backgrounds will have different probabilities of accepting each item, and so the agent must learn this probability over time by making different offers.

We allow for cooperation in this network by letting each agent recommend items of neighboring agents to incoming users, in addition to its own items. Thus if the specific background of the incoming user makes it unlikely for him to accept any of the agent’s items, the agent can instead recommend him some items from a neighboring agent with more attractive offerings. By trading referrals in this fashion, all of the agents that are connected together can benefit. To provide proper incentives, a commission will be paid to the recommending agent every time an item is accepted by a user from the recommended agent. When defined appropriately, this commission ensures that both sides will benefit each time a recommendation occurs and thus is able to sustain cooperation.

However, since agents are decentralized, they do not directly share the information that they learn over time about user preferences for their own items. So when the decision to recommend a neighboring agent occurs, it is done based solely on the previous successes the agent had when recommending that neighbor. Thus agents must learn about their neighbor’s acceptance probabilities through their own trial and error, unlike in other social learning papers such as [2]–[5], where agents share information directly with their neighbors.

Another key feature of our algorithms is that they are non-Bayesian unlike [2], [3]. Instead we model the learning through contextual bandits, where the context is based on the user’s background. By building upon the theory of contextual bandits, we produce a class of algorithms that allows agents to

This work is partially supported by the grants NSF CNS 1016081 and AFOSR DDDAS.
take near-optimal actions even with decentralized learning. We prove specific bounds for the regret, which is the difference between the total expected reward of an agent using a learning algorithm and the total expected reward of the optimal policy for the agent, which is computed given perfect knowledge about acceptance probabilities for each context. We show that the regret is sublinear in time in all cases. We further show that our algorithms can operate regardless of the specific network topology, including the degree of connectivity, degree distribution, clustering coefficient, etc., although the performance is better if the network is more connected since each agent will have access to more items of other agents.

The rest of the paper is organized as follows. Related work is given in Section II. The problem formulation is given in Section III. In Section IV we consider the online learning problem involving multiple decentralized agents and analyze its regret. In Section V-A we develop an algorithm to achieve sublinear regret when the purchase probability of the items depend on the other recommended items, and in Section V-C we develop a faster learning algorithm when item purchase probabilities are independent of each other. The effect of connectivity between the agents is given in Section VI. In Section VII numerical results demonstrating the effects of commissions, size of the set of items of agents and connectivity of agents are given, using an artificial data set which is based on a real data set. Finally, we conclude the paper in Section VII.


table

| Multi-agent | Cooperative | Contextual | Context arrival process | Regret | Multi-play for each agent | Action set size | Action sets of the agents |
|-------------|-------------|------------|-------------------------|--------|--------------------------|----------------|-------------------------|
| no          | yes         | yes        | arbitrary               | sublinear | no                       | no             | N/A                     |
| yes         | no          | yes        | arbitrary               | log    | no                       | no             | same                    |
| Contextual  | no          | yes        | arbitrary               | sublinear | yes                      | yes            | different               |
| yes         | yes         | yes        | arbitrary               | N/A    | yes                      | yes            | different               |

TABLE I: Comparison with related work in multi-armed bandits.

decentralized contextual combinatorial bandit model studied in this paper. This decentralization is important because it allows us to analyze a social network framework and the fundamental challenges associated with it including commissions, third-party sellers, etc. We are also able to consider the specific effects of the network structure on the regret in our model. In contrast, our approach in [10] does not address the network structure concerns. Several other examples of related work in contextual bandits are [12], in which a contextual bandit model is used for recommending personalized news articles based on a variant of the UCB1 algorithm in [13] designed for linear rewards, and [14] in which the authors solve a classification problem using contextual bandits, where they propose a perceptron based algorithm that achieves sublinear regret.

Apart from contextual bandits, there is a large set of literature concerned in multi-user learning using a multi-armed bandit framework [5]–[9]. We provide a detailed comparison between our work and related work in multi-armed bandit learning in Table I. Our cooperative contextual learning framework can be seen as an important extension of the centralized contextual bandits framework [6]. The main differences are that: (i) a three phase learning algorithm with training, exploration and exploitation phases is needed instead of the standard two phase algorithms with exploration and exploitation phases that are commonly used in centralized contextual bandit problems; (ii) the adaptive partitions of the context space should be formed in a way that each learner can efficiently utilize what is learned by other learners about the same context; (iii) since each agent has multiple selections at each time slot, the set of actions for each agent is very large, making the learning rate slow. Therefore, the correlation between the arms of the agents should be exploited to quicken the learning process. In our distributed multi-agent multiplay contextual bandit formulation, the training phase, which balances the learning rates of the agents, is necessary since the context arrivals to agents are different which makes the learning rates of the agents for various context different.

There is also an extensive literature on recommender systems that incorporates a variety of different methods and
frameworks. Table I provides a summary of how our work is related to other work. Of note, there are several papers that also use a similar multi-armed bandit framework for recommendations. For example, [20] considers a bandit framework where a recommender system learns about the preferences of users over time as each user submits ratings. It uses a linear bandit model for the ratings, which are assumed to be functions of the specific user as well as the features of the product. [21] is another work that utilizes multi-armed bandits in a recommendation system. It considers a model that must constantly update recommendations as both preferences and the item set changes over time.

There are also numerous examples of works that do not use a bandit framework for recommendations. One of the most commonly used methods for recommendations are collaborative filtering algorithms such as [22]–[28], which make recommendations by predicting the user’s preferences based on a similarity measure with other users. Items with the highest similarity score are then recommended to each user; for instance items may be ranked based on the number of purchases by similar users. There are numerous ways to perform the similarity groupings, such as the cluster model in [24], [27] that groups users together with a set of like-minded users and then makes recommendations based on what the users in this set choose. Another possibility is presented in [25], which pre-filters ratings based on context before the recommendation algorithm is started.

An important difference to keep in mind is that the recommendation systems in other works are a single centralized system, such as Amazon or Netflix. Thus the system has complete information at every moment in time, and does not need to worry about incentive issues or pay commissions. However, in this paper each agent is in effect its own separate recommendation system, since agents do not directly share information with each other. Therefore the mechanism we propose must be applied separately by every agent in the system based on that agent’s history of user acceptances. So in effect our model is a giant collection of recommendation systems that are running in parallel. The only cross-over between these different systems is when one agent suggests which of its items should be recommended by another agent. This allows for an indirect transfer of information, and lets that other agent make better choices than it could without this suggested list of items.

Also, it is important to note that decentralization in the context of our paper does not mean the same thing as in other papers such as [29]. Those papers assume that the users are decentralized, and develop mechanisms based on trust and the similarity of different users in order to allow users to recommend items to each other. We are assuming that the agents are decentralized, and so each user still only gets a recommendation from one source, it is just that this source may be different depending on which agent this user arrives at. Thus this paper is fundamentally different from the works in that literature.

### III. Problem Formulation

There are $M$ decentralized agents/learners which are indexed by the set $\mathcal{M} := \{1, 2, \ldots, M\}$. Let $\mathcal{M}_{\neq i} := \mathcal{M} - \{i\}$.

| Item-based (IB), user-based (UB) | Memory-based, model-based | Uses context info. | Performance measure | Similarity distance | Centralized(C), Decentralized(D) |
|--------------------------------|---------------------------|-------------------|-------------------|------------------|-----------------------------|
| UB                             | Memory-based              | No Ranking precision | -                 | C                |                             |
| UB                             | Bayesian-based model      | No MRE, RMS, 0/1 loss | Pearson correlation | C                |                             |
| UB                             | Bayesian-based Markov model | No Precision & Recall | Pearson correlation | C                |                             |
| IB                             | Cluster model             | No - Cosine       | -                 | C                |                             |
| UB                             | Memory-based              | Yes Precision & Recall | Pearson correlation | C                |                             |
| UB                             | Bayesian classifier model | No Precision & Recall | Pearson correlation | C                |                             |
| UB                             | Cluster model             | No MAE & Coverage  | Pearson correlation | C                |                             |
| UB                             | MDP model                 | No Recall          | Self-defined similarity | C                |                             |
| UB                             | MAB model                 | No Reward          | Lipschitz continuous | C                |                             |
| UB                             | MAB model                 | Yes Regret         | Lipschitz continuous | C                |                             |
| Our work                       | MAB model                 | Yes Regret         | Holder continuous | D                |                             |

TABLE II: Comparison with prior work in recommender systems.

Each agent $i$ has an inventory of items denoted by $\mathcal{F}_i$, which it can offer to its users and the users of other agents by paying some commission. Users arrive to agents in a discrete time setting ($t = 1, 2, \ldots$). Agent $i$ recommends $N$ items to its user at each time slot. For example, $N$ can be the number of recommendation slots the agent has on its website, or it can be the number of ads it can place in a magazine. We assume that $N$ is fixed throughout the paper. Each item $j \in \mathcal{F}_i$ has a fixed price $p_{ij} > 0$. We assume that the inventories of agents are mutually disjoint. For now, we will assume that all the agents in the network are directly connected to each other, so that any agent can sell items to the users of any other agent directly without invoking intermediate agents. We will discuss the agents in more general network topologies in Section V. Let $\mathcal{F} := \bigcup_{i \in \mathcal{M}} \mathcal{F}_i$ be the set of items of all agents. We assume that there is an unlimited supply of each type of item. This assumption holds for digital goods such as e-books, movies, videos, songs, photos, etc. An agent does not know the inventory of items of the other agents but knows an upper bound on $|\mathcal{F}_j|, j \in \mathcal{M}_{\neq i}$ which is equal to $F_{\text{max}}$.

We note that our model is suitable for a broad range of applications. The agent can represent a company, an entrepreneur, a content provider, a blogger, etc., and the items can be goods, services, jobs, videos, songs, etc. The notion of an item can be generalized even further to include such things as celebrities that are recommended to be followed in Twitter, Google+, etc. And the prices that we introduce later can also be generalized to mean any type of benefit the agent can receive from a user. For expositional convenience, we will adopt the formulation

1Even when an item is in the inventory of more than one agent, its price can be different among these agents, thus different IDs can be assigned to these items. We do not assume competition between the agents, hence our methods in this paper will work even when the inventories of agents are not mutually disjoint.

2For a set $A$, $|A|$ denotes its cardinality.
of firms selling goods to customers for most of the paper, but we emphasize that many other interpretations are equally valid and applicable.

A. Notation

Natural logarithm is denoted by \( \log(\cdot) \). For sets \( A \) and \( B \), \( A - B \) denotes the elements of \( A \) that are not in \( B \). \( P(\cdot) \) is the probability operator, \( E[\cdot] \) is the expectation operator. For an algorithm \( \alpha \), \( E_{\alpha}[\cdot] \) denotes the expectation with respect to the distribution induced by \( \alpha \). Let \( [t] := \{1, \ldots, t - 1\} \). Let \( \beta_2 := \sum_{t=1}^{\infty} 1/t^2 \).

B. Definition of users with contexts

At each time slot \( t = 1, 2, \ldots \), a user with a specific search query indicating the type of item the user wants, or other information including a list of previous purchases, price-range, age, gender etc., arrives to agent \( i \). We define all the properties of the arriving user known to agent \( i \) at time \( t \) as the context of that user, and denote it by \( x_i(t) \). We assume that the contexts of all users belong to a known space \( \mathcal{X} \), which without loss of generality is taken to be \( [0,1]^d \) in this paper, where \( d \) is the dimension of the context space. Although the model we propose in this paper has synchronous arrivals, it can be easily extended to the asynchronous case where agents have different user arrival rates, and even when no user arrives in some time slots. The only difference of this from our framework is that instead of keeping the same time index \( t \) for every agent, we will have different time indices for each agent depending on the number of user arrivals to that agent.

C. Definition of commissions

In order to incentivize the agents to recommend each other’s items, they will provide commissions to each other. In this paper we focus on sales commission, which is paid by the recommended agent to the recommending agent every time a user from the recommending agent purchases an item of the recommended agent. We assume that these commissions are fixed at the beginning and do not change over time. The system model is shown in Fig. 1. Basically, if agent \( i \) recommends an item \( f \) of agent \( j \) to its user, and if that user buys the item of agent \( j \), then agent \( i \) obtains a fixed commission which is equal to \( c_{i,j} > 0 \). All of our results in this paper will also hold for the case when the commission is the function of the price of the item \( f \) sold by agent \( j \), i.e., \( c_{i,j}(p^j_f) \). However we use the fixed commission assumption, since privacy concerns may cause the agent \( j \) or the user to not want to reveal the exact sales price to agent \( i \). We assume that \( c_{i,j} \leq p^j_f \) for all \( i,j \in \mathcal{M}, f \in \mathcal{F}_j \). Otherwise, full cooperation can result in agent \( j \) obtaining less revenue than it would have obtained without cooperation. Nevertheless, even when \( c_{i,j} > p^j_f \) for some \( f \in \mathcal{F}_j \subset \mathcal{F}_i \), by restricting the set of items that agent \( j \) can recommend to agent \( i \) to \( \mathcal{F}_j - \mathcal{F}_i \), all our analysis for agent \( i \) in this paper will hold.

Agents may also want to preserve the privacy of the items offered to the users, hence the privacy of their inventory in addition to the privacy of prices. This can be done with the addition of a regulator to the system. The regulator is a non-strategic entity whose goal is to promote privacy among the

---

### Fig. 1: Operation of the system for agent \( i \) for \( N = 3 \) recommendations. At each time a user arrives to agent \( i \) with context \( x_i(t) \), agent \( i \) recommends a set of its own items and items from other agents, which is denoted by \( \mathcal{N}_i(t) \).

---

D. Recommendations, actions and purchase probabilities

The \( N \) items that agent \( i \) recommends to its user are chosen in the following way: An item can be chosen from the inventory \( \mathcal{F}_j \) of agent \( i \), or agent \( i \) can call another agent \( j \) and send the context information of \( x_i(t) \) of its user, then agent \( j \) returns back an item \( f' \) with price \( p^j_{f'} \) to be recommended to agent \( i \) based on \( x_i(t) \). Let \( \mathcal{N}_i(t) \) be the set of items recommended by agent \( i \) to its user at time \( t \). Let \( \mathcal{A}_N \) be the set of subsets of \( \mathcal{F} \) with \( N \) items. Let \( \mathcal{N} \subset \mathcal{A}_N \) be a set of \( N \) recommended items.

Consider agent \( i \). Let \( U_i := \mathcal{M}_i - \mathcal{F}_i \). For \( u_i \in \mathcal{F}_i \), let \( (u_i, 1) ((u_i, 0)) \) denote the event that \( u_i \) is recommended (not recommended) to agent \( i \)’s user. For \( u_i \in \mathcal{M}_i - \mathcal{F}_i \), let \( (u_i, n_{u_i}) \) denote the event that agent \( i \) requests \( n_{u_i} \) (distinct) items to

---

3 Another method to preserve the privacy of prices is the following: Agent \( j \)’s item will be recommended by agent \( i \) without a price tag, and when the user clicks to agent \( j \)’s item it will be directed to agent \( j \)’s website where the price will be revealed to the user.
be recommended to its user from agent \( i \). Let \((u_i, n_i) := \{(u_i, n_{ui})\}_{u_i \in U_i}\) be an action for agent \( i \). Based on this, let \( \mathcal{L}_i := \{(u_i, n_i)\text{ such that } n_{ui} \in \{0, 1\}\text{ for } u_i \in \mathcal{F}_i\text{ and } n_{ui} \in \{0, \ldots, N\}\text{ for } u_i \in \mathcal{M}_{-i}\text{ and } \sum_{u_i \in U_i} n_{ui} = N\}\),

be the set of actions available to agent \( i \). We assume that \(|\mathcal{F}_i| \geq N\) for all \( j \in \mathcal{M} \). Let \( k_i \) be the components of vector \((u_i, n_i) \in \mathcal{L}_i\) with nonzero \( n_{ui} \). Since \( n_{ui} = 0 \) means the corresponding item will not be recommended for \( u_i \in \mathcal{F}_i \), and the corresponding agent will not recommend any items to agent \( i \) for \( u_i \in \mathcal{M}_{-i}, k_i \) is enough to completely characterize the action of agent \( i \). Thus, for notational simplicity we denote an action of agent \( i \) by \( k_i \). With an abuse of notation \( k_i \in \mathcal{L}_i \) means that \((u_i, n_i)\) corresponding to \( k_i \) is in \( \mathcal{L}_i \). For \( u_i \in \mathcal{U}_i\), let \( n_{ui}(k_i) \) be the value of \( n_{ui} \) in the vector \((u_i, n_i)\) corresponding to \( k_i \). Let \( \mathcal{M}_i(k_i) \) be the set of agents in \( \mathcal{M}_{-i} \) that recommend at least one item to agent \( i \) when it chooses action \( k_i \). We define \( m_j(k_i) \) as the number of items agent \( j \) recommends to agent \( i \) when agent \( i \) chooses action \( k_i \in \mathcal{L}_i \). Clearly we have \( m_j(k_i) = n_j(k_i) \) for \( j \in \mathcal{M}_{-i} \), and \( m_k(k_i) = \sum_{u_i \in \mathcal{F}_i} n_{ui}(k_i) \). Let \( m(k_i) := (m_1(k_i), \ldots, m_M(k_i)) \) be the recommendation vector given action \( k_i \),

\[ m_{-i}(k_i) := (m_1(k_i), \ldots, m_{j-1}(k_i), m_{j+1}(k_i), \ldots, m_M(k_i)), \]

and \( \mathcal{F}_i(k_i) \subset \mathcal{F}_i \) be the set of own items recommended by agent \( i \) to its user when it takes action \( k_i \in \mathcal{L}_i \). For a set of recommended items \( \mathcal{N} \in \mathcal{A}_N \), let \( \mathcal{G}_i(\mathcal{N}) := \mathcal{N} \cap \mathcal{F}_i \) and \( g_i(\mathcal{N}) := |\mathcal{G}_i(\mathcal{N})| \). Let \( g_{-i}(\mathcal{N}) := \{g_j(\mathcal{N})\}_{j \in \mathcal{M}_{-i}} \). Below, we define the purchase probability of an item \( f \) that is recommended to a user with context \( x \) along with the set of items \( \mathcal{N} \in \mathcal{A}_N \).

**Assumption 1: Group-dependent purchase probability:** We assume that the inventory of each agent forms a separate group (or type) of items. For example, if agents are firms, then firm \( i \) may sell televisions while firm \( j \) sells headphones. For each item \( f \) recommended together with the set of items \( \mathcal{N} \in \mathcal{A}_N \), if \( f \) is an item of agent \( j \), i.e., it belongs to group \( \mathcal{F}_j \), then its acceptance probability will depend on other recommended items in the same group, i.e., \( \mathcal{G}_j(\mathcal{N}) \). The acceptance probability of an item \( f \) in \( \mathcal{F}_j \) may also depend on groups of other items offered together with \( f \) and also on their number, i.e., \( g_{-j}(\mathcal{N}) \), but not on their identities. For \( f \in \mathcal{F}_j \), a user with context \( x \in \mathcal{X} \) will buy the item with an unknown probability \( q_f(x, \mathcal{N}) := q_f(x, \mathcal{G}_j(\mathcal{N}), g_{-j}(\mathcal{N})) \). For all \( f \in \mathcal{F}_j \), users with contexts similar to each other have similar purchase probabilities. This similarity is characterized by a Hölder condition that is known by the agents, i.e., there exists \( L > 0, \alpha > 0 \) such that for all \( x, x' \in \mathcal{X} = \{0, 1\}^d \), we have

\[ |q_f(x, \mathcal{N}) - q_f(x', \mathcal{N})| \leq L||x - x'||^\alpha, \]

where \( || \cdot || \) denotes the Euclidean norm in \( \mathbb{R}^d \).

In Assumption 1, we do not require the purchase probabilities to be similar for different sets of recommendations from the same group, i.e., for \( f \in \mathcal{N} \) and \( f \in \mathcal{N}' \) such that \( \mathcal{G}_j(\mathcal{N}) \neq \mathcal{G}_j(\mathcal{N}') \), the purchase probability of \( f \) can be different for context \( x \in \mathcal{X} \). Even though the Hölder condition can hold with different constants \( L_f \) and \( \alpha_f \) for each item \( f \in \mathcal{F}_j \), taking \( L \) to be the largest among \( L_f \) and \( \alpha \) to be the smallest among \( \alpha_f \) we get the condition in Assumption 1.

For example, the context can be the (normalized) age of the user, and users with ages similar to each other can like similar items. We will also consider a simplified version, in which the purchase probability of an item is independent of the set of other items recommended along with that item.

**Assumption 2: Independent purchase probability:** For each item \( f \) offered along with the items in the set \( \mathcal{N} \in \mathcal{A}_N \), a user with context \( x \) will buy the item with an unknown probability \( q_f(x, \mathcal{N}) := q_f(x) \), independent of the other items in \( \mathcal{N} \), for which there exists \( L > 0, \alpha > 0 \) such that for all \( x, x' \in \mathcal{X} \), we have \( |q_f(x) - q_f(x')| \leq L||x - x'||^\alpha \).

When Assumption 2 holds, the agents can estimate the purchase probability of an item by using the empirical mean of the number of times the item is purchased by users with similar context information. The purchase probabilities of items can be learned much faster in this case compared with Assumption 1 since the empirical mean will be updated every time the item is recommended. However, when the purchase probability of an item is group-dependent, the agents need to learn its purchase probability separately for each group the item is in.

### E. Objective, rewards, and the regret

The goal of agent \( i \) is to maximize its total expected reward (revenue). However, since it does not know the purchase probability of any item or the inventories of other agents a-priori, it must learn how to make optimal recommendations to its users over time. In this subsection we define the optimal actions for the agents and the “oracle” benchmark (optimal) solution the agents’ learning algorithms compete with, which is derived from the unknown purchase probabilities and inventories.

The expected reward of agent \( i \) at time \( t \) from recommending a set of items \( \mathcal{N}_i \) to its user with context \( x_i(t) = x \) is given by

\[
\sigma_i(\mathcal{N}_i, x) := \sum_{f \in (\mathcal{N}_i - \mathcal{F}_i)} e_{i,j(f)} q_f(x, \mathcal{N}_i) + \sum_{f \in \mathcal{N}_i \cap \mathcal{F}_i} p_{j,f} q_f(x, \mathcal{N}_i),
\]

where \( j(f) \) is the agent who owns item \( f \). Based on this, given a user with context \( x \), the set of items which maximizes the one-step expected reward of agent \( i \) is

\[
\mathcal{N}_i^*(x) := \text{arg max}_{\mathcal{N} \in \mathcal{A}_N} (\sigma_i(\mathcal{N}, x)). \quad (1)
\]

Since the inventory of other agents and \( q_f(x, \mathcal{N}) \), \( x \in \mathcal{X} \), \( \mathcal{N} \in \mathcal{A}_N \) are unknown a priori to agent \( i \), \( \mathcal{N}_i^*(x) \) is unknown to agent \( i \) for all contexts \( x \in \mathcal{X} \). For a recommendation vector \( m = (m_1, \ldots, m_M) \), agent \( j \) has \( |\mathcal{F}_j| \) actions \( k_j \in \mathcal{F}_j \) for which \( m_{j'}(k_j) = m_{j'} \) for all \( j' \neq j \). Denote this set of actions of agent \( j \) by \( B_j(m) \). For an action \( k_j \in B_j(m) \), let

\[
\pi_{j,k_j}(x) := \sum_{f \in \mathcal{F}_j(k_j)} q_f(x, \mathcal{F}_j(k_j), m_{-j}(k_j)),
\]


be the purchase rate of action $k_i$ for a user with context $x$. The purchase rate is equal to the expected number of items in $F_j(k_j)$ sold to a user with context $x$, when the number of items recommended together with $F_j(k_j)$ from the other groups is given by the vector $m_{-j}(k_j)$. For actions of agent $i$ such that $m_j(k_i) > 0$, the best set of items agent $j$ can recommend to agent $i$ (which maximizes agent $i$’s expected commission from agent $j$) for a user with context $x$ is $F_j(k_j^*(k_i,x))$, where

$$k_j^*(k_i,x) := \arg \max_{k_j \in \mathcal{B}_j(m_{-j}(k_j))} \pi_{j,k_j}(x).$$

According to the above definitions, the expected reward of action $k_i \in \mathcal{L}_i$ to agent $i$ for a user with context $x$ is defined as

$$\mu_{i,k_i}(x) := \sum_{f \in F_i(k_i)} p^f_i q_f(x,F_i(k_i), m_{-i}(k_i)) + \sum_{j \in M_i(k_i)} c_{i,j} \pi_{j,k^*_j(k_i,x)}(x).$$

(2)

Then, given a user with context $x$, the best action of agent $i$ is

$$k_i^*(x) := \arg \max_{k_i \in \mathcal{L}_i} \mu_{i,k_i}(x).$$

(3)

Note that for both Assumptions 1 and 2, $\mu_{i,k_i^*(x)}(x) = \sigma_i(N_i^*(x), x)$. Let $\alpha_i(t)$ be the recommendation strategy adopted by agent $i$ for its own users, i.e., based on its past observations and decisions, agent $i$ chooses a vector $\alpha_i(t) \in \mathcal{L}_i$ at each time slot. Let $\beta_i$ be the recommendation strategy adopted by agent $i$ when it is called by another agent to recommend its own items. Let $\alpha = (\alpha_1, \ldots, \alpha_M)$ and $\beta = (\beta_1, \ldots, \beta_M)$. Let $Q_{\alpha,\beta}(T)$ be the expected total reward of agent $i$ by time $T$ from item sales to its own users and users of other agents and commissions it gets from item sales of other agents to its own users. For $f \in F_i$, let $Y_{f,i}^\beta(t)$ be a random variable which is equal to 1 when agent $i$ recommends its own item $f$ to its user at time $t$ (0 otherwise), let $Y_{f,j}^{\beta_i,\alpha_j}(t)$ be a random variable which is equal to 1 when agent $i$ recommends its own item $f$ to agent $j$ when it is called by agent $j$ at time $t$ (0 otherwise). For $f \in F_i$, let $Y_{f,i}^\alpha(t)$ be a random variable that is equal to 1 when agent $i$ asks for recommendations from agent $j$ at time $t$ for its own user (0 otherwise). Let the (random) reward agent $i$ gets from the set of recommendations made by $(\alpha, \beta)$ to its user at time $t$ be

$$O_{\alpha,\beta}(t,x_i(t)) := \sum_{f \in F_i} Y_{f,i}^\alpha(t)p^f_i q_f(x_i(t), N_i(t)) + \sum_{j \in M_{-i}} Y_{f,j}^{\alpha_j}(t) \left( \sum_{f \in F_j} Y_{f,j}^{\beta_j,\alpha_j}(t)c_{i,j} p^f_j q_f(x_i(t), N_i(t)) \right),$$

(4)

and let

$$S_{\alpha,\beta}(T) := E_{\alpha,\beta} \left[ \sum_{t=1}^T O_{\alpha,\beta}(t,x_i(t)) \right],$$

be the total expected reward agent $i$ can get based only on recommendations to its own users by time $T$. Let

$$U_{\alpha,\beta}(t,x_j(t)) := Y_{i,j}^\alpha(t) \sum_{f \in F_j} Y_{f,j}^{\alpha_j}(t),$$

(5)

be the number of items of agent $i$ purchased by agent $j$’s user at time $t$. Clearly we have

$$Q_{\alpha,\beta}(T) = S_{\alpha,\beta}(T) + E_{\alpha,\beta} \left[ \sum_{t=1}^T \sum_{j \in M_{-i}} c_{i,j} U_{\alpha,\beta}(t,x_j(t)) \right].$$

We can see that $Q_{\alpha,\beta}(T) \geq S_{\alpha,\beta}(T)$. Agent $i$’s goal is to maximize its total reward $S_{\alpha,\beta}(T)$ from its own users for any $T$. Since agents are cooperative, agent $i$ also helps other agents $j \in M_{-i}$ to maximize $S_{\alpha,\beta}(T)$ by recommending its items to them. Hence the total reward the agent gets, $Q_{\alpha,\beta}(T)$, is at least $S_{\alpha,\beta}(T)$.

We assume that user arrivals to the agents are independent of each other. Therefore, agent $j$ will also benefit from agent $i$ if its item can be sold by agent $i$. In this paper, we develop distributed online learning algorithms $(\alpha, \beta)$ for the agents in $\mathcal{M}$ such that the expected total reward $S_{\alpha,\beta}(T)$ for any agent $i \in \mathcal{M}$ is maximized. This corresponds to minimizing the regret, which is given for agent $i$ at time $T$ as

$$R_{\alpha,\beta}(T) := \sum_{t=1}^T \mu_{i,k_i^*(x_i(t))}(x_i(t)) - S_{\alpha,\beta}(T).$$

(6)

Note that the regret is calculated with respect to the highest expected reward agent $i$ can obtain from its own users, but not the users of other agents. Therefore, agent $i$ does not act strategically to attract the users of other agents, such as by cutting its own prices or paying commissions even when an item is not sold to increase its chance of being recommended by another agent. We assume that agents are fully cooperative and follow the rules of the proposed algorithms. We provide a comparison between cooperative and strategic agents in the following remark.

Remark 1: In our cooperative algorithms, an agent $j$ when called by agent $i$ recommends a set of items that has the highest probability of being purchased by agent $i$’s user. This recommendation does not decrease the reward of agent $j$, compared to the case when agent $j$ does not cooperate with any other agent, since we assume that $p^f_j \geq c_{i,j}$ for all $f \in F_j$. However, when the commission is fixed, recommending the set of items with the highest purchase rate does not always maximize agent $j$’s reward. For example, agent $j$ may have another item which has a slightly smaller probability of being purchased by agent $i$’s user, but has a much higher price than the item which maximizes the purchase rate. Then, it is of interest to agent $j$ to recommend that item to agent $i$ rather than the item that maximizes the purchase rate. This problem can be avoided by charging a commission which is equal to a percentage of the sales price on the learning rates and set of items recommended by the agents in Section VI.
We will show that the regret of the algorithms proposed in this paper will be sublinear in time, i.e., \( R^i_{\alpha, \beta}(T) = O(T^\phi) \), \( \phi < 1 \), which means that the distributed learning scheme converges to the average reward of the best recommender strategy \( N_i^* (x) \) given in (1) (or equivalently \( k_i^* (x) \) given in (3)) for each \( i \in M, x \in X \), i.e., \( R^i_{\alpha, \beta}(T)/T \to 0 \). Moreover, the regret also provides us with a bound on how fast our algorithm converges to the best recommender strategy.

IV. CONTEXTUAL PARTITIONING ALGORITHMS FOR MULTIPLE RECOMMENDATIONS

In this section we propose a series of distributed online learning algorithms called Context Based Multiple Recommendations (CBMR). We denote the part of CBMR which gives an action to agent \( i \) at every time slot for its own user with \( \alpha_i^{CBMR} \), and the part of CBMR which gives an action to agent \( i \) at every time slot for the recommendation request of another agent with \( \beta_i^{CBMR} \). When clear from the context, we will drop the superscript. Basically, an agent using CBMR forms a partition of the context space \([0, 1]^d\), depending on the final time \( T \), consisting of \((m_T)^d\) sets where each set is a \( d \)-dimensional hypercube with dimensions \( 1/m_T \times 1/m_T \times \ldots \times 1/m_T \). The sets in this partition are indexed by \( I_T = \{1, 2, \ldots, (m_T)^d\} \). We denote the set with index \( i \) with \( I_i \). Agent \( i \) learns about the rewards and purchase rates of its actions in each set in the partition independently from the other sets in the partition based on the context information of the users that arrived to agent \( i \) and the users for which agent \( i \) is recommended by another agent. Since users with similar contexts have similar purchase probabilities (Assumptions [1] and [2]), it is expected that the optimal recommendations are similar for users located in the same set \( I_i \in I_T \). Since the best recommendations are learned independently for each set in \( I_T \), there is a tradeoff between the number of sets in \( I_T \) and the estimation of the best recommendations for contexts in each set in \( I_T \). We will show that in order to bound regret sublinearly over time, the parameter \( m_T \) should be non-decreasing in \( T \).

There are two versions of CBMR: one for group-dependent purchase probabilities given by Assumption [1] which is called CBMR-d, and the other for independent purchase probabilities given by Assumption [2] which is called CBMR-ind. The difference between these two is that CBMR-d calculates the expected reward from each action in \( \hat{L}_i \) for agent \( i \) separately, while CBMR-ind forms the expected reward of each action in \( \hat{L}_i \) based on the expected rewards of the items recommended in the chosen action. We have

\[
|\hat{L}_i| = \left( _{|F_i|}^N \right) + \sum_{n=0}^{N-1} \left( _{|F_i|}^n \right) \left( \frac{N - n + M - 2}{M - 1} \right),
\]

which grows combinatorially in \( N \) and \( M \) and polynomially in \(|F_i|\). We will show that CBMR-ind provides much faster learning than CBMR-d when Assumption [2] holds. We explain these algorithms in the following subsections.

A. Definition of CBMR-d

The pseudocode of CBMR-d is given in Fig. 3. At any time \( t \) in which agent \( i \) chooses an action \( k_i \in \hat{L}_i \), for any agent \( j \in M_i(k_i) \), it should send a request to that agent for \( m_j(k_i) \) recommendations along with the context of its user \( x_i(t) \) and the recommendation vector \( m(k_i) \). If the agents do not want to share the set of recommendations and prices they made for agent \( i \)'s user with agent \( i \), they can use the regulator scheme proposed in Section III-C. CBMR-d can be in any of the following three phases at any time slot \( t \): the training phase in which agent \( i \) chooses an action \( k_i \) such that it trains another agent \( j \) by asking it for recommendations and providing \( i \)'s user's context information \( x \), so that agent \( j \) will learn to recommend its set of \( m_j(k_i) \) items with the highest purchase rate for a user with context \( x \), the exploration phase in which agent \( i \) forms accurate estimates of the expected reward of actions \( k_i \in \hat{L}_i \) by selecting \( k_i \) and observing the purchases, and the exploitation phase in which agent \( i \) selects the action in \( \hat{L}_i \) with the highest estimated reward to maximize its total reward. The pseudocodes of these phases are given in Fig. 3.

At each time \( t \), agent \( i \) first checks which set in the partition \( I_T \) context \( x_i(t) \) belongs to. We separate the set of actions in \( \hat{L}_i \) into two. Let

\[ \hat{L}_i = \{(u_i, n_i) \in \hat{L}_i \text{ such that } n_i = 0, \ \forall u_i \in M_{-i} \} \]

be the set of actions in which all recommendations are from \( F_i \), and \( \hat{L}_i = \hat{L}_i - \hat{L}_i \) be the set of actions in which at least one recommendation is from an agent in \( M_{-i} \).

The training phase is required for actions in \( \hat{L}_i \), while only exploration and exploitation phases are required for actions in \( \hat{L}_i \). When agent \( i \) chooses an action \( k_i \in \hat{L}_i \), the agents \( j \in M_i(k_i) \) recommend \( m_j(k_i) \) items from \( F_j \) to agent \( i \). Recall that \( N_i(t) \) denotes the set of \( N \) items that is recommended to agent \( i \)'s user at time \( t \), based on the actions chosen by agent \( i \) and the recommendations chosen by other agents for agent \( i \). For \( k_i \in \hat{L}_i \), let \( N_{k_i}(t) \) be the number of times action \( k_i \) is selected in response to a context arriving to the set \( I_i \in I_T \) by time \( t \). Since agent \( i \) does not know \( F_i \) and the purchase rates for \( j \in M_i(k_i) \), before forming estimates about the reward of action \( k_i \), it needs to make sure that \( j \) will almost always recommend its set of \( m_j(k_i) \) items with the highest purchase rate for agent \( i \)'s user. Otherwise, agent \( i \)'s estimate about the reward of action \( k_i \) might deviate a lot from the correct reward of action \( k_i \), with a high probability, resulting in agent \( i \) choosing suboptimal actions most of the time, hence resulting in a high regret. This is why the training phase is needed for actions in \( \hat{L}_i \).

In order to separate training, exploration and exploitation phases, agent \( i \) keeps two counters for actions \( k_i \in \hat{L}_i \) for each \( I_i \in I_T \). The first one, i.e., \( N_{k_i,t}(t) \), counts the number of user arrivals to agent \( i \) with context in set \( I_i \) in the training phases of agent \( i \) by time \( t \), in which it had chosen action \( k_i \). The second one, i.e., \( N_{k_i,t}(t) \), counts the number of user arrivals to agent \( i \) with context in set \( I_i \) in the exploration and the exploitation phases of agent \( i \) by time \( t \), in which it had chosen action \( k_i \). The observations made at the exploration and the exploitation phases are used to estimate the expected reward of agent \( i \) from taking action \( k_i \). In addition to the counters \( N_{k_i,t}(t) \), \( N_{k_i,t}(t) \) and \( N_{k_i,t}(t) \), agent \( i \) also keeps control functions, \( D_1(t) \), \( D_2(t) \), \( k_i \in \hat{L}_i \) and \( D_3(t) \), which
Context Based Multiple Recommendations for dependent purchase probabilities (CBMR-d for agent $i$):

1: Input: $D_1(t), D_2,k(t), k \in \hat{L}_i, D_3(t), T, m_T$
2: Initialize: Partition $[0, 1]^d$ into $(m_T)^d$ sets, indexed by the set $\mathcal{I}_T = \{1, 2, \ldots, (m_T)^d\}$.
3: $N^i_{k,l,t} = 0, \forall k \in \mathcal{L}_i, l \in \mathcal{I}_T, N^i_{1,k,l} = 0, N^i_{2,k,l} = 0, \forall k \in \hat{L}_i, l \in \mathcal{I}_T.$
4: $\tilde{\pi}_{k,l,t} = \{N^i_{k,l,t}\}_{k \in \mathcal{L}_i}, \tilde{\pi}_{1,l,t} = \{N^i_{2,k,l}\}_{k \in \hat{L}_i}.$
5: $\tilde{\pi}_{k,l,t} = 0, \tilde{\pi}_{1,l,t} = 0, \forall k \in \mathcal{L}_i, l \in \mathcal{I}_T.$
6: $\tilde{\pi}_{k,l,t} = \{\tilde{\pi}_{k,l,t}\}_{l \in \mathcal{I}_T}.$
7: while $t \geq 1$ do
8: Algorithm $\alpha_i$ (Send recommendations to own users)
9: for $l = 1, \ldots, (m_T)^d$ do
10: if $x_i(t) \in I_i$ then
11: if $\exists k \in \hat{L}_i$ such that $N^i_{k,l,t} \leq D_1(t)$ then
12: Run Explore($k, N^i_{k,l,t}, \tilde{\pi}_{k,l,t}, t$)
13: else if $\exists k \in \mathcal{L}_i$ such that $N^i_{1,k,l} \leq D_2,k(t)$
14: Run Train($k, N^i_{1,k,l}, t$)
15: else if $\exists \hat{k} \in \hat{L}_i$ such that $N^i_{2,k,l} \leq D_3(t)$ then
16: Run Explore($k, N^i_{2,k,l}, \tilde{\pi}_{2,k,l}, t$)
17: else
18: Run Exploit($N^i_{1,l,t}, N^i_{2,l,t}, \tilde{\pi}_{1,l,t}, t$)
19: end if
20: end if
21: end for
22: Algorithm $\beta_i$ (Send recommendations to other agents’ users)
23: for $j \in M_{-i}$ such that $i \in M_j(\alpha_j(t))$ do
24: Receive $x_j(t)$ and $m_j(\alpha_j(t))$
25: for $l = 1, \ldots, (m_T)^d$ do
26: if $x_j(t) \in I_i$ then
27: if $\exists k \in B_j(m_j(\alpha_j(t)))$ such that $N^i_{k,l,t} \leq D_1(t)$
28: Run Explore2($k, N^i_{k,l,t}, \tilde{\pi}_{k,l,t}, j, t$)
29: else
30: Run Exploit2($N^i_{1,l,t}, \tilde{\pi}_{1,l,t}, m_i(\alpha_j(t)), j, t$)
31: end if
32: end if
33: end for
34: end for
35: $t = t + 1$
36: end while

Fig. 2: Pseudocode of CBMR-d for agent $i$.

are used together with the counters determine when to train, explore or exploit. The control functions are deterministic and non-decreasing functions of $t$ that are related to the minimum number of observations of an action that is required so that the estimated reward of that action is sufficiently accurate to get a low regret in exploitations. We will specify the exact values of these functions later when we prove our regret results.

In addition to the counters and control functions mentioned above, for each $k_i \in \mathcal{L}_i$ and $l_i \in \mathcal{I}_T$, agent $i$ keeps a sample mean reward $\tilde{\pi}_{k_i,l_i,t}$ which is the sample mean of the rewards (sum of prices of sold items and commissions divided by number of times $k_i$ is selected except the training phase) agent $i$ obtained from its recommendations when it selected action $k_i$ in its exploration and exploitation phases for its own user with context in $I_i$ by time $t$. Agent $i$ also keeps a sample mean purchase rate $\tilde{\pi}_{k_i,l_i,t}$ which is the sample mean of the number of agent $i$’s own items in $F_i(k_i)$ sold to a user (either agent $i$’s own user or user of another agent) with context in set $I_i$ when agent $i$ selects action $k_i$ for that user.

Train($k, n, t$):
1: Select action $\alpha_i(t) = k$.
2: Receive reward $\tilde{r}_k(t) := O_{\alpha_i(t)}(t,x_i(t))$ given in (4).
3: $n + +$.

Explore($k, n, r, t$):
1: Select action $\alpha_i(t) = k$.
2: Receive reward $\tilde{r}_k(t) := O_{\alpha_i(t)}(t,x_i(t))$ given in (4).
3: $r = (nm + \tilde{r}_k(t))/(n + 1)$.
4: $n + +$.

Exploit($n, r, t$):
1: Select action $\alpha_i(t) \in \arg \max_{k \in \mathcal{L}_i} r_k$.
2: Receive reward $\tilde{r}_{\alpha_i(t)}(t) := O_{\alpha_i(t)}(t,x_i(t))$ given in (4).
3: $r_{\alpha_i(t)} = (n_{\alpha_i(t)} + \tilde{r}_{\alpha_i(t)}(t))/(n_{\alpha_i(t)} + 1)$.
4: $n_{\alpha_i(t)} + +$.

Experimental($n, \pi, \beta_i, t$):
1: Select action $\beta_i(t) \in \arg \max_{k \in B_j(m_j)} \pi_{k_i}$.
2: Recommend set of items $F_j(k_i)$ to agent $j$.
3: Receive purchase feedback $\tilde{r}_{\beta_i(t)}(t) := U_{\beta_i(t)}(t,x_j(t))$ given in (5) and the commission.
4: $n + +$.

Experimental($n, \pi, \beta_i, j$):
1: Select action $\beta_i(t) \in \arg \max_{k \in B_j(m_j)} \pi_{k_i}$.
2: Recommend set of items $F_j(k_i)$ to agent $j$.
3: Receive purchase feedback $\tilde{r}_{\beta_i(t)}(t) := U_{\beta_i(t)}(t,x_j(t))$ given in (5) and the commission.
4: $n + +$.

$S_{1,i}(t) := \{k_i \in \mathcal{L}_i$ such that $N^i_{1,k_i,l}(t) \leq D_1(t)$ or $k_i \in \hat{L}_i$ such that $N^i_{2,k_i,l}(t) \leq D_2,k_i(t)$ or $N^i_{3,k_i,l}(t) \leq D_3(t)\}$,

in order to make sure that it has accurate estimates of the expected rewards of each of its actions $k_i \in \mathcal{L}_i$, as well as to make sure that other agents have accurate estimates about the purchase rates of their own set of items for a user with context in set $I_i$.

For $k_i \in \hat{L}_i$, let $E_{k_i,l_i}(t)$ be the set of rewards collected from selections of action $k_i$ at times $t \in [t]$ when agent $i$’s user’s context is in set $I_i$, and all the other agents in $M_i(k_i)$ are trained sufficiently, i.e., $N^i_{1, k_i, l_i}(t') > D_2, k_i(t')$. For $k_i \in \hat{L}_i$, let $E_{k_i,l_i}(t)$ be the set of rewards collected from action

We will explain how agent $i$ selects an action $k_i$ when agent $j$ requests items from $i$ when we describe $\beta_i$.
k_i by time t. If \( S_{i,t} \neq \emptyset \), then agent i trains or explores by randomly choosing an action \( \alpha_i(t) \in S_{i,t}(t) \). If \( S_{i,t}(t) = \emptyset \), this implies that all actions in \( L_i \) have been trained and explored sufficiently, so that agent i exploits by choosing the action with the highest sample mean reward, i.e.,

\[
\alpha_i(t) \in \arg \max_{k_i \in L_i} \bar{r}_{k_i,i}(t).
\]

We have \( \bar{r}_{k_i,i}(t) = (\sum_{r \in E_{k_i,i}(t)} r)/|E_{k_i,i}(t)| \). When there is more than one action which has the highest sample mean reward, one of them is randomly selected.

The other part of CBMR-d, i.e., \( \beta_i \), gives agent i the set of items to recommend to agent j when agent j takes an action \( \alpha_j(t) = k_j \in L_j \) for which \( i \in M_j(k_j) \), and sends to agent i its user's context \( x_j(t) \in I_l \in \mathcal{I}_T \) and the recommendation vector \( m(k_j) \). In order to recommend the set of items with the maximum purchase rate for the user of agent j, agent i should learn the purchase rate of its own items for the recommendation vector \( m(k_j) \). Agent i responds to agent j's request in the following way.

In the following subsection we prove an upper bound on the regret of CBMR-d.

**B. Analysis of the regret of CBMR-d**

For each \( I_l \in \mathcal{I}_T \) and \( k_i \in L_i \), let \( \bar{\mu}_{k_i,k_i}(x) = \sup_{x \in I_l} \mu_{k_i,k_i}(x) \), \( \underline{\mu}_{k_i,k_i} := \inf_{x \in I_l} \mu_{k_i,k_i}(x) \), \( \bar{\pi}_{k_i,k_i} := \sup_{x \in I_l} \pi_{k_i,k_i}(x) \), and \( \underline{\pi}_{k_i,k_i} := \inf_{x \in I_l} \pi_{k_i,k_i}(x) \). Let \( k_i^*(x) \) be the context of the center of the set \( I_l \). We define the optimal reward action of agent i for set \( I_l \) as

\[
k_i^*(l) := \arg \max_{k_i \in L_i} \mu_{k_i,k_i}(x_i^*) ,
\]

and the optimal purchase rate action of agent j for agent i given agent i selects action \( k_i \) for set \( I_l \) as

\[
k_j^*(k_i,l) := \arg \max_{k_j \in B_j(m(k_i))} \pi_{j,k_j}(a_j^*). \]

Let

\[
K_{\theta,a_i,l}(t) := \{ k_i \in L_i : \bar{\mu}_{k_i,k_i}(l,t) - \underline{\mu}_{k_i,k_i} > a_1 t^\theta \},
\]

be the set of suboptimal reward actions for agent i at time t, and

\[
\mathcal{J}_{\theta,a_i,l}(t) := \{ k_j \in B_j(m(k_i)) : \bar{\pi}_{j,k_j}(l,t) - \underline{\pi}_{j,k_j} > a_1 t^\theta \},
\]

be the set of suboptimal purchase rate actions of agent j for agent i, given agent i chooses action \( k_i \) at time t, where \( \theta < 0 \), \( a_1 > 0 \). The agents are not required to know the values of the parameters \( \theta \) and \( a_1 \). They are only used in our analysis of the regret. First, we will give regret bounds that depend on values of \( \theta \) and \( a_1 \), and then we will optimize over these values to find the best bound. Let \( \mathcal{Y}_T := \sup_{x \in X} R_{t,k_i,k_i}(x) \), which is the maximum expected loss agent i can incur for a time slot in which it chooses a suboptimal action.

The regret given in (6) can be written as a sum of three components:

\[
R^i(T) = E[R^i_e(T)] + E[R^i_w(T)] + E[R^i_n(T)],
\]

where \( R^i_e(T) \) is the regret due to training and explorations by time \( T \), \( R^i_w(T) \) is the regret due to suboptimal action selections in exploitations by time \( T \), and \( R^i_n(T) \) is the regret due to near optimal action \( k_i \in \mathcal{L}_i - K_{\theta,a_i,l}(t) \) selections in exploitations by time \( T \) of agent i, which are all random variables. In the following lemmas we will bound each of these terms separately. The following lemma bounds \( E[R^i_e(T)] \). Due to space limitations we give the lemmas in this and the following subsections without proofs. The proofs can be found in our online appendix [31].

**Lemma 1:** When CBMR-d is run with parameters \( D_1(t) = D_3(t) = t^2 \log t \), \( D_{2,k_i}(t) = \max_{j \in M_i(k_i)} F_{\max}(m_j(k_i)) t^2 \log t \), and \( m_T = [T^\gamma] \), where \( 0 < z < 1 \) and \( 0 < \gamma < 1/d \), we have

\[
E[R^i_e(T)] \leq Y R^{2d} \left( \left| \mathcal{L}_i \right| + \left| \mathcal{L}_i \right| \left( \frac{F_{\max}}{F_{\max}/2} \right) \right)^{T+\gamma d} \log T + Y R^{2d} \left| \mathcal{L}_i \right|^T d.
\]

**Proof:** We sum over all exploration and training steps by time \( T \). The contribution to the regret is at most \( Y R \) in each of these steps.

From Lemma 1 we see that the regret due to explorations is linear in the number of sets in partition \( \mathcal{I}_T \), i.e., \( (m_T)^d \), hence exponential in parameters \( \gamma \) and \( z \). We conclude that \( z \) and \( \gamma \) should be small to achieve sublinear regret in training and exploration phases.

For any \( k_i \in \mathcal{L}_i \) and \( I_l \in \mathcal{I}_T \), the sample mean \( \bar{r}_{k_i,l}(t) \) represents a random variable which is the average of the independent samples in set \( E_{k_i,l}(t) \). Different from classical finite-armed bandit theory [13], these samples are not identically distributed. In order to facilitate our analysis of the regret, we generate two different artificial i.i.d. processes to bound the probabilities related to \( \bar{r}_{k_i,l}(t) \), \( k_i \in \mathcal{L}_i \). The first one is the best process in which rewards are generated according to a bounded i.i.d. process with expected reward \( \bar{\mu}_{k_i,k_i} \), the one other is the worst process in which rewards are generated according to a bounded i.i.d. process with expected reward \( \underline{\mu}_{k_i,k_i} \). Let \( \tau_{k_i,l} \) denote the sample mean of the \( \tau \) samples from the best process and \( \tau_{k_i,l} \) denote the sample mean of the \( \tau \) samples from the worst process. We will bound the terms \( E[R^i_e(T)] \) and \( E[R^i_n(T)] \) by using these artificial processes along with the Hölder condition given in Assumption 1. Details of this are given in [31]. The following lemma bounds \( E[R^i_e(T)] \).
Lemma 2: When CBMR-d is run with parameters $D_1(t) = D_3(t) = t^2 \log t$, $D_{2,k}(t) = \max_{j \in M_i(k_i)} \left( F_{\text{max}}(m_{j,k_i}) \right) t^2 \log t$, and $m_T = [T^3]$, where $0 < z < 1$ and $0 < \gamma < 1/d$, given that $2LY_R(\sqrt{d})^{1-\gamma} + 2(Y_R + 2)t^{-z/2} \leq a_1^\theta$, we have

$$E[R_i^z(T)] \leq 2^{d+1}Y_R\hat{\ell}_i|_2 T^{1+\gamma} + 2^{d+2}Y_R\hat{\ell}_i|_2 T^{1+\gamma}\sqrt{\gamma}\sqrt{d}/z.$$ 

Proof: This proof is similar to the proof of Lemma 2 in [32]. The difference is that instead of bounding the probabilities that a suboptimal arm in $F_i$ will be chosen or another agent $j \in M_i$, $\hat{\ell}_i$ chooses a suboptimal arm in $F_j$ when called by agent $i$, we bound the probabilities that a suboptimal action in $\hat{\ell}_i$ is chosen and a suboptimal action in $\hat{\ell}_j$ is chosen. Another difference is that every time a suboptimal action is chosen, the one-step regret can be at most $Y_R$.

From Lemma 2, we see that the regret increases exponentially with parameters $\gamma$ and $z$, similar to the result of Lemma 1. These two lemmas suggest that $\gamma$ and $z$ should be as small as possible, given that the condition $2LY_R(\sqrt{d})^{1-\gamma} + 2(Y_R + 2)t^{-z/2} \leq a_1^\theta$ is satisfied.

When agent $i$ chooses an action $k_i \in \hat{\ell}_i$ such that $j \in M_i(k_i)$, there is a positive probability that agent $j$ will choose a suboptimal set of $m_{j,k_i}$ items to recommend to agent $i$'s user, i.e., it will choose a suboptimal purchase rate action for agent $i$'s action. Because this is not optimal, a near optimal action $k_i \in \hat{\ell}_i - \hat{\ell}_i|_2 (t)$ will still get a low reward. We need to take this into account in order to bound $E[R_i^z(T)]$. The following lemma gives the bound on $E[R_i^z(T)]$.

Lemma 3: When CBMR-d is run with parameters $D_1(t) = D_3(t) = t^2 \log t$, $D_{2,k}(t) = \max_{j \in M_i(k_i)} \left( F_{\text{max}}(m_{j,k_i}) \right) t^2 \log t$, and $m_T = [T^3]$, where $0 < z < 1$ and $0 < \gamma < 1/d$, given that $2LY_R(\sqrt{d})^{1-\gamma} + 2(Y_R + 2)t^{-z/2} \leq a_1^\theta$, we have

$$E[R_i^z(T)] \leq (2a_1 T^{1+\theta})/(1 + \theta) + 2Y_R \left( \frac{F_{\text{max}}}{F_{\text{max}}/2} \right) \beta_2.$$ 

Proof: If a near optimal action in $\hat{\ell}_i$ is chosen at time $t$, the contribution to the regret is at most $a_1^\theta$. If a near optimal action $k_i \in \hat{\ell}_i$ is chosen at time $t$, and if some of the agents in $M_i(k_i)$ choose their near optimal items to recommend to agent $i$, then the contribution to the regret is at most $2a_1^\theta$. Therefore, the total regret due to near optimal action selections in $\hat{\ell}_i$ by time $T$ is upper bounded by $2a_1 \sum_{t=1}^{T} t^\theta \leq (2a_1 T^{1+\theta})/(1 + \theta)$, by using the result in Appendix A in [32]. Each time a near optimal action in $k_i \in \hat{\ell}_i$ is chosen in an exploitation step, there is a small probability that an item chosen by some agent $j \in M_i(k_i)$ is a suboptimal one. Using a result similar to Lemma 3 in [32], the expected number of times a suboptimal arm is chosen is bounded by $2( F_{\text{max}}/2) \beta_2$. Each time a suboptimal arm is chosen, the regret can be at most $Y_R$.

From Lemma 3, we see that the regret due to near optimal actions depends exponentially on $\theta$ which is related to the negative of $\gamma$ and $z$. Therefore, $\gamma$ and $z$ should be chosen as large as possible to minimize the regret due to near optimal actions. Combining the above lemmas, we obtain the finite time regret bound for agents using CBMR-d, which is given in the following theorem.

Theorem 1: When CBMR-d is run with parameters $D_1(t) = D_3(t) = t^{2\alpha/(3\alpha + d)} \log t$, $D_{2,k}(t) = \max_{j \in M_i(k_i)} \left( F_{\text{max}}(m_{j,k_i}) \right) t^{2\alpha/(3\alpha + d)} \log t$, and $m_T = [T^{1/(3\alpha + d)}]$, we have

$$R_i^z(T) \leq T^{2\alpha+d} \left( \frac{4(Y_R(Ld^{\alpha/2} + 1) + 1)}{(2\alpha + d)/(3\alpha + d)} + Y_R 2^d Z_i \log T \right) + T^{2\alpha+d} \frac{2\alpha/(3\alpha + d)}{2\alpha/(3\alpha + d)}$$ 

$$+ T^{2\alpha+d} 2^d Y_R(2)\hat{\ell}_i|_2 + |\ell_i| + 2Y_R \left( \frac{F_{\text{max}}}{F_{\text{max}}/2} \right) \beta_2,$$

i.e., $R_i(T) = O \left( Z_i T^{2\alpha+d} \right)$, where $Z_i = |\ell_i| + |\hat{\ell}_i| \left( \frac{F_{\text{max}}}{F_{\text{max}}/2} \right)$. The values that minimize the regret for which this constraint holds are $\theta = -z/2$, $\gamma = z/(2\alpha)$, $a_1 = 2Y_R(Ld^{\alpha/2} + 1) + 4$ and $z = 2\alpha/(3\alpha + d)$.

The result follows from summing the bounds in Lemmas 1, 2 and 3.

Remark 2: A uniform partition of the context space such as $\mathcal{T}$ may not be very efficient when user arrivals have contexts that are concentrated into some (unknown) regions of the context space. In such a case, it is better to explore and train these regions more frequently than regions with few context arrivals. Algorithms that start with the entire context space as a single set and then adaptively partition the context space into smaller regions as more and more users arrive may achieve faster convergence rates, i.e., smaller regret, for the types of arrivals mentioned above. Such an algorithm that will work for $N = 1$ is given in [32].

Theorem 1 indicates that agent $i$ can achieve sublinear regret with respect to the best “oracle” recommendation strategy which knows the purchase probabilities of all the items in $\mathcal{F}$. However, the learning rate of CBMR-d can be slow when $M$, $N$ and $\mathcal{F}$ are large since the set of actions is combinatorial in these parameters. As a final remark, since regret is sublinear, the average reward of CBMR-d converges to the average reward of the best “oracle” recommendation strategy, i.e., $\lim_{T \to \infty} R_i^z(T)/T = 0$.

C. Definition of CBMR-ind

In this subsection we describe the learning algorithm CBMR-ind. We assume that Assumption 2 holds. Let $\mathcal{J}_{i,j} := \{j_1, j_2, \ldots, N_j\}$ denote the set of the number of recommendations agent $i$ can request from agent $j$, where we use the subscript $j$ to denote that the recommendations are requested from agent $j$. Let $\mathcal{J}_i := \bigcup_{j \in M_i} \mathcal{J}_{i,j}$, and $\mathcal{J}_i := F_i \cup \mathcal{J}_i$ be the set of arms of agent $i$. We have $|\mathcal{J}_i| = |\mathcal{F}_i| + (M - 1)N_j$. We denote an arm of agent $i$ by index $u$. For arm $u \in \mathcal{J}_i$, let $j(u)$ be the agent that is called for item recommendations to agent $i$’s user, and let $n(u)$ be the number of requested items
from agent $j(u)$ for $u \in \mathcal{F}_i$. $j(u) = i$ and $n(u) = 1$. In CBMR-ind, at each time $t$, agent $i$ chooses a set of arms such that the total number of item recommendations it makes to its user is $N_i$. It is evident that choosing such a set of arms is equivalent to choosing an action $k_i \in \mathcal{L}_i$. Every action $k_i \in \mathcal{L}_i$ maps to a unique set of arms. Thus, for an action $k_i \in \mathcal{L}_i$, let $\mathcal{A}(k_i)$ be the set of arms corresponding to $k_i$.

Let $\mathcal{G}_j(u)(x) \subset \mathcal{F}_j$ be the set of $n$ items in $\mathcal{F}_j$ with the highest purchase probabilities for a user with context $x$. For an arm $u \in \mathcal{F}_i$, let its purchase rate for a user with context $x$ be $\nu_i(u)(x) := q_u(x)$, and for an arm $u \in \mathcal{J}_i$, let it be $\nu_i(u)(x) := \sum_{f \in \mathcal{G}_j(u), (n_i, n)} q_f(x)$. Since Assumption 2 holds, we have

$$
\mu_{i,k}(x) = \sum_{u \in \mathcal{F}_i \cap \mathcal{A}(k_i)} p_i^{ji} u_i(u(x)) + \sum_{u \in \mathcal{A}(k_i) - \mathcal{F}_i} c_{i,j}(u) \nu_i(u(x)).
$$

Different from CBMR-d, which estimates the reward of each action in $\mathcal{L}_i$ separately, CBMR-ind estimates the purchase rates of the arms in $\mathcal{J}_i$, and uses them to construct the estimated rewards of actions in $\mathcal{L}_i$. The advantage of CBMR-ind is that the purchase rate estimate of an arm $u \in \mathcal{J}_i$ can be updated based on the purchase feedback whenever any action $\mathcal{L}_i$ that contains arm $u$ is selected by agent $i$.

The pseudocode of CBMR-ind is given in Fig. 4. CBMR-ind partitions the context space in the same way as CBMR-d. Unlike CBMR-d, CBMR-ind does not have exploration, exploitation and training phases for actions $k_i \in \mathcal{L}_i$. Rather than that, it has exploration and exploitation phases for each arm $u \in \mathcal{F}_i$, and exploration, exploitation and training phases for each arm $u \in \mathcal{J}_i$. Since a combination of arms is selected at each time slot, selected arms can be in different phases. For each $u \in \mathcal{J}_i$ and $i \in \mathcal{I}_T$, CBMR-ind keeps the sample mean purchase rate $\tilde{\nu}^i_{u,t}(t)$, which is the sample mean of the number of purchased items corresponding to arm $u$, purchased by users with contexts in $\mathcal{J}_i$ in exploration and exploitation phases of agent $i$ by time $t$.

Similar to the counters and control functions of CBMR-d, CBMR-ind also keeps counters and control functions for each $u \in \mathcal{J}_i$ and $i \in \mathcal{I}_T$. Basically, for $u \in \mathcal{J}_i$, $N^{u}_{i,t}(t)$ counts the number of times arm $u$ is selected by agent $i$ to make a recommendation to a user with context in $\mathcal{I}_i$ by time $t$. Counters $N^1_{i,u,t}(t)$ and $N^2_{i,u,t}(t)$ are only kept for arms $u \in \mathcal{J}_i$. The former one counts the number of times arm $u$ is trained by agent $i$ for times its users had contexts in $\mathcal{J}_i$ by time $t$, while the latter one counts the number of times arm $u$ is explored and exploited by agent $i$ for times its users had contexts in $\mathcal{J}_i$ by time $t$.

Let

$$
\mathcal{S}^{ind}_{i,t}(t) := \left\{ u \in \mathcal{F}_i \text{ such that } N^1_{i,u,t}(t) \leq D^1_{i,t}(t) \text{ or } u \in \mathcal{J}_i \text{ such that } N^2_{i,u,t}(t) \leq D^2_{i,t}(t) \right\},
$$

where $D^1_{i,t}(t)$, $D^2_{i,t}(t)$, $u \in \mathcal{J}_i$ and $D^2_{i,t}(t)$ are counters similar to the counters of CBMR-d. Assume that $x_i(t) \in \mathcal{I}_t$. If $\mathcal{S}^{ind}_{i,t}(t) \neq \emptyset$, then agent $i$ randomly chooses an action $\alpha_i(t) \in \mathcal{L}_i$ from the set of actions for which $\mathcal{A}(k_i) \cap \mathcal{S}^{ind}_{i,t}(t) \neq \emptyset$.

$$
j_i(u), u \in \mathcal{J}_i \text{ is different from } f_i, f \in \mathcal{F}, \text{ which denotes the agent that owns item } f, \text{ and } n(u) \text{ is different from } n_i(k_i), \text{ which denotes the number of items agent } j \text{ should recommend when agent } i \text{ chooses action } k_i \in \mathcal{L}_i.
$$

if $\mathcal{S}^{ind}_{i,t}(t) = \emptyset$, then agent $i$ will choose the action

$$
\alpha_i(t) = \arg \max_{k_i \in \mathcal{L}_i} \sum_{u \in \mathcal{F}_i \cap \mathcal{A}(k_i)} p_i^{ji} u_i(u(t)) + \sum_{u \in \mathcal{A}(k_i) - \mathcal{F}_i} c_{i,j}(u) \tilde{\nu}^i_{u,t}(t). \tag{9}
$$

After action $\alpha_i(t)$ is chosen, the counters and sample mean purchase rates of arms in $\mathcal{A}(\alpha_i(t))$ are updated based on in which phase they are in.

Agent $i$ responds to item requests of other agents with users with contexts in $I_i \in \mathcal{I}_T$ in the following way. Any under-explored item $u \in \mathcal{F}_i$, i.e., $N^1_{i,u,t}(t) \leq D^1_{i,t}(t)$ is given priority to be recommended. If the number of under-explored items is less than the number of requested items, then the remaining items are selected from the set of items in $u \in \mathcal{F}_i$ with the highest sample mean purchase rates such that $N^1_{i,u,t}(t) > D^1_{i,t}(t)$.

The pseudocode of this is not given due to limited space.

**Context Based Multiple Recommendations for independent purchase probabilities (CBMR-ind for agent $i$)**

1. **Input:** $D_{1,t}(t)$, $D_{2,u,t}(t)$, $u \in \mathcal{J}_i$, $D_{3,t}(t)$, $T$, $m_T$.
2. **Initialize:** Partition $[0, 1]$ into $(m_T)^d$ sets, indexed by the set $\mathcal{I}_T = \{1, 2, \ldots, (m_T)^d\}$.
3. $N^1_{i,u,t} = 0, \forall u \in \mathcal{J}_i, I_i \in \mathcal{I}_T$.
4. $\tilde{\nu}^i_{u,t} = 0, \forall u \in \mathcal{J}_i, I_i \in \mathcal{I}_T$.
5. **while** $t \geq 1$ **do**
6. **for** $l = 1, \ldots, (m_T)^d$ **do**
7. **if** $x_i(t) \in I_l$ **then**
8. **if** $\mathcal{S}^{ind}_{i,t}(t) \neq \emptyset$ **then**
9. Choose $\alpha_i(t)$ randomly from the subset of $\mathcal{L}_i$ such that $\mathcal{A}(\alpha_i(t)) \cap \mathcal{S}^{ind}_{i,t}(t) \neq \emptyset$.
10. **else**
11. Choose $\alpha_i(t)$ as in (9).
12. **end if**
13. **end if**
14. **end for**
15. **Receive reward** $O^t \tilde{\nu}^i_{u,t}(t, x_i(t))$ **given in (4).
16. **if** $u \in \mathcal{A}(\alpha_i(t))$ **do**
17. **if** $u \in \mathcal{J}_i$ and $N^1_{i,u,t}(t) \leq D_{2,u,t}(t)$ **then**
18. $N^1_{i,u,t} +$.
19. **else if** $u \in \mathcal{J}_i$ and $N^1_{i,u,t}(t) > D_{2,u,t}(t)$ **then**
20. $\tilde{\nu}^i_{u,t} = \frac{N_{2,u,t}^i \tilde{\nu}^i_{u,t} + N^2_{i,u,t} \nu^i_{u,t}(t, x_i(t))}{N_{2,u,t}^i + 1}$. (given in (5)).
21. $N_{2,u,t}^i +$.
22. **else**
23. $\tilde{\nu}^i_{u,t} = (N^1_{i,u,t} \tilde{\nu}^i_{u,t} + Y^i_{u,t}(t))/(N^2_{i,u,t} + 1)$.
24. $N_{i,u,t}^i +$.
25. **end if**
26. **end if**
27. $t = t + 1$.
28. **end while**

Fig. 4: Pseudocode of CBMR-ind for agent $i$.

**D. Analysis of the regret of CBMR-ind**

In this subsection we bound the regret of CBMR-ind. Under Assumption 2, the expected reward of an item $f$ to agent $i$ for a user with context $x$ is $\kappa_{i,f}(x) := p_i q_f(x)$ for $f \in \mathcal{F}_i$ and $\kappa_{i,j}(x) := c_{i,j} q_f(x)$ for $f \in \mathcal{J}_i$. For a set of items $\mathcal{N}$, let $f_n^i(\mathcal{N})$ denote the item in $\mathcal{N}$ with the $n$th highest expected reward for agent $i$. For an item $f \in \mathcal{F}$ and $I_i \in \mathcal{I}_T$, let
We will optimize over $\alpha_i$ and $\theta$ as we did in Section IV-B. The set of near-optimal arms at time $t$ for $I_i$ is $J_i - U_{\theta,\alpha_i}(t)$. Similar to the approach we took in Section IV-B, we divide the regret into three parts: $R_i(t), R_i(T)$, and $R_i(T)$, and bound them individually. Different from the analysis of CBMR-d, here $R_i(t)$ denotes the regret in time slots in which all selected arms are exploited and at least one them is suboptimal, while $R_i(T)$ denotes the regret in time slots in which all selected arms are exploited and all of them are near-optimal. In the following lemma, we bound the regret of CBMR-ind due to explorations and trainings.

**Lemma 4:** When CBMR-ind is run by the agents with $D_1(t) = t^2 \log t$, $D_{2,u}(t) = (F_{\text{max}}(u)) t^2 \log t$, $D_3(t) = t^2 \log t$ and $m_T = [T^\gamma]$, where $0 < z < 1$ and $0 < \gamma < 1/d$, we have

$$E[R_i(t)] \leq Y_{R}T^{2d}([J_i] + (M - 1)N)T^\gamma d + Y_{R}T^{2d}([J_i] + (M - 1)\sum_{z=1}^{N} \frac{F_{\text{max}}(z)}{z}) T^z + \gamma d \log T.$$  

**Proof:** For a set $I_i \in T_i$, the regret due to explorations is bounded by $[J_i]T^\gamma \log T$. Agent $i$ spends at most $\sum_{z=1}^{N} \frac{F_{\text{max}}(z)}{z} T^\gamma \log T$ time steps to train agent $j$. Note that this is the worst-case number of trainings for which agent $j$ does not learn about the purchase probabilities of its items in set $I_i$ from its own users, and from the users of other agents other than agent $i$. The result follows from summing over all sets in $T_i$.

In the next lemma, we bound $E[R_i(t)]$.

**Lemma 5:** When CBMR-ind is run by the agents with $D_1(t) = t^2 \log t$, $D_{2,u}(t) = (F_{\text{max}}(u)) t^2 \log t$, $D_3(t) = t^2 \log t$ and $m_T = [T^\gamma]$, where $0 < z < 1$ and $0 < \gamma < 1/d$, given that $2LY_{R}(\sqrt{d})\alpha_t^{z-\gamma} + 2(Y_{R} + 2)T^{z-\gamma}/2 \leq \alpha_t^T$, we have

$$E[R_i(t)] \leq 2d^{t+1}Y_{R}T^{\beta_2}T^{\gamma d} + 2d^{t+2}NY_{R}[J_i]\beta_2T^{\gamma d + z}/2.$$  

**Proof:** Let $\Omega$ denote the space of all possible outcomes, and let $w$ be a sample path. Let $W_i(t) := \{w : \hat{S}_i(w) = \emptyset\}$ denote the event that CBMR-ind is in the exploitation phase at time $t$. The idea is to bound the probability that agent $i$ selects at least one suboptimal arm in an exploitation step, and then using this to bound the expected number of times a suboptimal arm is selected by agent $i$. Let $V_{\hat{S}_i}(t)$ be the event that a suboptimal action $u \in J_i$ is chosen at time $t$ by agent $i$. We have

$$E[R_i(t)] \leq Y_{R} \sum_{t \in T_i} \sum_{i=1}^{T} \sum_{u \in U_{\theta,\alpha_i}(t)} P(V_{\hat{S}_i}(t), W_i(t)).$$

Similar to the proof of Lemma 2 in [32], by using a Chernoff bound it can be shown that for any $u \in U_{\theta,\alpha_i}(t)$, we have

$$P(V_{\hat{S}_i}(t), W_i(t)) \leq 2/t^2.$$  

We refer the reader to [32] for the proof.

**Theorem 2:** When CBMR-ind is run by the agents with $D_1(t) = t^{2\gamma} \log t$, $D_{2,u}(t) = (F_{\text{max}}(u)) t^{2\gamma} \log t$, $D_3(t) = t^{2\gamma} \log t$ and $m_T = [T^\gamma]$, where $0 < z < 1$ and $0 < \gamma < 1/d$, given that $2LY_{R}(\sqrt{d})\alpha_t^{z-\gamma} + 2(Y_{R} + 2)T^{z-\gamma}/2 \leq \alpha_t^T$, we have

$$E[R_i(t)] \leq 2d^{t+1}Y_{R}[J_i]\beta_2T^{\gamma d + z}/2.$$  

**Proof:** Since agent $i$ can choose at most $N$ arms at each time step, if all arms chosen at time $t$ by agent $i$ and by the other agents called by agent $i$ are near-optimal, then the contribution to the regret is bounded by $2N\alpha_t^T$. This is because for a near optimal arm $u \in J_i$, the contribution to the regret is at most $2\alpha_t^T$. Therefore, the total regret due to near optimal arm selections by agent $i$ and the agents $\{j\}$ calls by time $T$ is upper bounded by $2N\alpha_t^T \leq 2d^{t+1}Y_{R}(1/2)/(1/2)$, and over time, and then by using the result in Appendix A in [32].

However, each time a near optimal arm in $u \in J_i$ is chosen in an exploitation step, there is a small probability that an item chosen recommended by agent $j$ is a suboptimal one. Using a result similar to Lemma 3 in [32], the expected number of times a suboptimal arm is chosen by the called agent is bounded by $n(u)F_{\text{max}}\beta_2$. Each time a suboptimal item is chosen by the called agent $j(u), the regret can be at most $Y_{R}$.

Combining the above lemmas, we obtain the regret bound for agents using CBMR-ind.

**Theorem 2:** When CBMR-ind is run by the agents with $D_1(t) = t^{2\gamma} \log t$, $D_{2,u}(t) = (F_{\text{max}}(u)) t^{2\gamma} \log t$, $D_3(t) = t^{2\gamma} \log t$ and $m_T = [T^{1/2}]$, we have

$$R_i(t) \leq 2d^{t+1}Y_{R}[J_i]\beta_2T^{\gamma d} + 2d^{t+2}NY_{R}[J_i]\beta_2T^{\gamma d + z}/2.$$  

**Proof:** Let $\Omega$ denote the space of all possible outcomes, and let $w$ be a sample path. Let $W_i(t) := \{w : \hat{S}_i(w) = \emptyset\}$ denote the event that CBMR-ind is in the exploitation phase at time $t$. The idea is to bound the probability that agent $i$ selects at least one suboptimal arm in an exploitation step, and then using this to bound the expected number of times a suboptimal arm is selected by agent $i$. Let $V_{\hat{S}_i}(t)$ be the event that a suboptimal action $u \in J_i$ is chosen at time $t$ by agent $i$. We have

$$E[R_i(t)] \leq Y_{R} \sum_{t \in T_i} \sum_{i=1}^{T} \sum_{u \in U_{\theta,\alpha_i}(t)} P(V_{\hat{S}_i}(t), W_i(t)).$$

Similar to the proof of Lemma 2 in [32], by using a Chernoff bound it can be shown that for any $u \in U_{\theta,\alpha_i}(t)$, we have

$$P(V_{\hat{S}_i}(t), W_i(t)) \leq 2/t^2.$$  

We refer the reader to [32] for the proof. The result follows from summing the bounds in Lemmas 4 and 5.
depends linearly on $|J_i|$, which is much better than the linear dependence of the regret of CBMR-d on $|L_i|$. We would also like to note that the number of trainings and explorations, and hence the regret, can be significantly reduced when agent $i$ knows $|F_i|$ for all $j \in M_{-i}$. In this case, agent $i$ can use the control function $D_{2,u}(t) := \left(\frac{|F_i|}{n(u)}\right)^{2\alpha/(3\alpha+d)} \log t$ to decide if it needs to train arm $u$.

E. Comparison of CBMR-d and CBMR-ind

In this subsection we compare CBMR-d and CBMR-ind in terms of their regret bounds, training and exploration rates, and memory requirements. Note that the regret bound of CBMR-d depends on the size of the action space $L_i$, which grows combinatorially with $M$ and $N$, and is an $N$ degree polynomial of $|F_i|$. In contrast the size of the arm space $J_i$ is just $|F_i| + (M - 1)N$. This is due to the fact that CBMR-d explores and exploits each action without exploiting the correlations between different actions. When the purchase probabilities depend on the set of items offered together, in the worst case there may be no correlation between rewards of different actions, and therefore the best one can do is to form independent sample mean estimates for each action in $L_i$. However, when the purchase probabilities are independent of the items offered together, since the expected reward of agent $i$ from an action $k_i \in L_i$ is the sum of the expected rewards of the individual arms chosen by agent $i$ in that action, substantial improvement over the regret bound is possible due to smaller number of explorations and trainings.

Another advantage of CBMR-ind is that it requires a significantly smaller amount of memory than CBMR-d. CBMR-d needs to keep sample mean rewards and sample mean purchase rates for all actions and for all partitions of the context space, while CBMR-ind only needs to keep sample mean purchase rates for all arms and for all partitions of the context space.

V. PERFORMANCE AS A FUNCTION OF THE NETWORK

In our analysis in the previous section we assumed that all agents are directly connected to each other. In reality some agents may not be connected to each other. For example, agents $i$ and $j$ can both be connected to agent $j'$, but there may not exist a direct link between agent $i$ and agent $j$. This can happen when, for example, a pair of companies has a trade agreement between each other, but there is no trade agreement between companies $i$ and $j$. We assume that whenever a trade link exists between agents $i$ and $j$ it is bidirectional. In this case, even though agent $i$ cannot ask for items to recommend, it can ask for $j'$ and $j'$ can ask for $i$ for an item. Then, if $i$ sells $j'$’s item, agent $i$ will get commission $c_{i,j'}$ from $j'$, while agent $j'$ will get commission $c_{j',i} + c_{i,j'}$ from agent $j$ so that it recovers the payment it made to agent $i$ from agent $j$. We call agent $j'$ the relay agent. Let $M_{i}^\text{dir}$ denote the set of agents that are directly connected to agent $i$.

If agent $i$ only cooperates with the agents in $M_{i}^\text{dir}$ and all cooperating agents use CBMR-d or CBMR-ind, then agent $i$ can achieve the sublinear regret bounds given in Theorems 1 and 2 with respect to the optimal distributed recommendation policy involving only the agents in $\{i\} \cup M_{i}^\text{dir}$. However, since agent $i$ cannot exploit the advantage of the items of other agents which are in $M_{-i} - M_{i}^\text{dir}$, its regret can be linear with respect to the optimal distributed recommendation policy involving all the agents.

CBMR-d and CBMR-ind can be modified in the following way to account for agents distributed over the network. Let $M_{i,j}(p)$ be the set of relay agents between agents $i$ and $j$ when they are connected through path $p$. Let $D_{p} := |M_{i,j}(p)|$. Let $h_1(p)$ be the agent that is connected directly to agent $i$ in path $p$, let $h_2(p)$ be the agent that is connected directly to agent $h_1(p)$, and so on. Then, the commission framework is modified such that when agent $i$ sells agent $j$’s item it gets a commission $c_{i,h_1(p)}$ from agent $h_1(p)$, agent $h_1(p)$ gets a commission $c_{h_1(p),h_2(p)} + c_{i,h_1(p)}$ from agent $h_2(p)$, and so on, such that $c_{h_{D_{p}(p)},j} \leq p_{f}^j$, where $f$ is $F_i$ is the item recommended by agent $j$ to agent $i$’s user. Using this scheme, all agents benefit from the series of transactions described above, thus it is better for them to cooperate based on the rules of the commission scheme than to act on their own. We assume that agent $j$ will not recommend an item to agent $h_{D_{p}(p)}$ if $c_{h_{D_{p}(p)},j} > p_{f}^j$ for all $f \in F_i$. Assume a connected network of agents $G(M,E)$ in which the maximum degree is $D_k$ and the longest path has $D_h$ hops, where $E$ denotes the set of direct links between the agents. Assume that the commissions $c_{i,j} > 0$ are given between any agent $i$ and $j$ with direct links. We define agent $i$’s regret $R_{G(M,E)}^i(T)$ to be the difference between the expected total reward of the optimal policy for which agent $i$ has access to the set of all the items of all agents in the network $G(M,E)$ (but if the item is in $F_i$, it gets the commission from agent $j'$, which is the agent that is directly connected to agent $i$ in the lowest commission path from agent $i$ to agent $j$) and the expected total reward of the learning algorithm used by agent $i$. The exploration and training control functions are run using $(D_k)^{D_h}F_{\max}$ instead of $F_{\max}$. This way agent $i$ will help the relay agents learn the other agent’s recommendations accurately such that sublinear regret bounds can be achieved. The following theorem gives a bound on the regret of agents when they use the modified version of CBMR-ind discussed above, which we call CBMR-ind-N. A similar bound can also be proven for the modified version of CBMR-d.

Theorem 3: When Assumption 2 holds, if CBMR-ind-N is run by all agents in $G(M,E)$, with agent $i$ running CBMR-ind-N with the set of arms $J_i$ as described in Section IV-C, defined using $M_{i}^\text{dir}$ and $F_i$ instead of $M$ and $F_i$, control functions $D_1(t) = D_3(t) = t^{2\alpha/(3\alpha+d)} \log t$, $D_{2,u}(t) = \left(\frac{(D_k)^{D_h}F_{\max}}{n(u)}\right)^{2\alpha/(3\alpha+d)} \log t$, and $m_T = \lceil T/(3\alpha+d) \rceil$; and if commissions are such that all agents $j \in M_{-i}$ get a positive reward when their items are sold by agent $j$’s we have,$R_{G(M,E)}^i(T) \leq T^{\frac{2\alpha+d}{3\alpha+d}} 2^{2\alpha+2}NY_R|J_i|\beta_2^{\frac{2\alpha+d}{3\alpha+d}} + 2^2\alphaNY_D(D_k)^{D_h}F_{\max}\beta_2^2$.

4If the commission an agent needs to pay to sell an item to the user of agent $i$ is greater than the price of all the items of that agent, then that agent can be removed from the set of agents which agent $i$ can cooperate with. Then, our results will hold for the remaining set of agents.
Amazon data set, we take these products and other $F_1$ products that are frequently co-purchased with the set of $N_1$ products as our set of items.

The set of products that are taken in the first step of the above procedure is denoted by $C_h$. The set of all products, i.e., $\mathcal{F}$, contains these $N_1$ products and the set of products frequently co-purchased with them, which we denote by $C_f$. We assume that each item has a unit price of 1, but have different purchase probabilities for different types of users. Since user information is not present in the data set, we generate it artificially by assuming that every incoming user searches for a specific item in $C_h$. This search query (item) will then be the context information of the user, hence the context space is $C_h$. Thus, we set $\mathcal{I}_T = C_h$. Based on this, the agent that the user arrives to recommends $N$ items to the user. The agent’s goal is to maximize the total number of items sold to its users.

We generate the purchase probabilities in the following way: For group-dependent purchase probabilities, when $n$ of the products recommended for context $x \in C_h$ are in the set of frequently co-purchased products with item $x$, then the purchase probability of each of these products will be $g_{nc}(n) = 1 - an$, where $0 < a < 1/N$. For the other $N - n$ products which are not frequently co-purchased with item $x$, their purchase probability is $g_{nc} = b$, where $0 < b < 1 - a$. For independent purchase probabilities, when a product recommended for context $x$ is in the set of frequently co-purchased products with item $x$, the purchase probability of that product will be $g_c$. When it is not, the purchase probability of that product will be $g_{nc}$, for which we have $g_c > g_{nc}$.

We assume that there are 3 agents and evaluate the performance of agent 1 based on the number of users arriving to agent 1 with a specific context $x^*$, which we take as the first item in set $C_h$. We assume that $T = 100,000$, which means that 100,000 users with context $x^*$ arrive to agent 1. Since the arrival rate of context $x^*$ can be different for the agents, we assume arrivals with context $x^*$ to other agents are drawn from a random process. We take $N_1 = 20$, $F_1 = 2$ and $N = 2$. As a result, we get 30 distinct items in $\mathcal{F}$ which are distributed among the agents such that $|\mathcal{F}_i| = 10$ for every agent $i$. Since the context space is discrete we have $d = 1$, and there is no Hölder condition on the purchase probabilities as given in Assumptions 1 and 2, hence we take $\alpha = 1/13$ such that $2\alpha/(3\alpha + d) = 1/8$. Unless otherwise stated, we assume that $g_c = 0.1$, $g_{nc} = 0.01$, $a = 0.5$ and $c_{t,j} = 0.5$.

B. Comparison of the reward and regret of CBMR-d and CBMR-ind for group-dependent purchase probabilities

We run both CBMR-d and CBMR-ind for group-dependent purchase probabilities assuming that both items that are frequently co-purchased with context $x^*$ are in agent 1’s inventory. The “oracle” optimal policy recommends one of the frequently co-purchased items and another item in agent 1’s inventory to the user with context $x^*$, instead of recommending the two frequently co-purchased items together. Expected total reward of the “oracle” optimal policy, total rewards of CBMR-d and CBMR-ind, and the number of trainings of CBMR-d

\[
\begin{align*}
+ T^{\frac{2\alpha + d}{3\alpha + d}} \left( 4N(Y_R(L_d^\alpha/2 + 1) + \frac{Y_{\alpha/2} d}{(2\alpha + d)/(3\alpha + d)} + Y^{\alpha/2} l \log T \right) \\
+ T^{\frac{2\alpha + d}{3\alpha + d}} 2d Y_R(2 |J_t| \beta_2 + |J_t| (M - 1) N),
\end{align*}
\]

i.e.,
\[R_{\alpha}G_{\alpha}(M,E)(T) = O \left( Z_T^{2\alpha + d} \log T \right),\]

where $Z_T := |J_t| + (M - 1) \sum_{z=1}^{N} (D_i z^d F_{max}).$

Proof: The proof is similar to the proof of Theorem 2 but more explorations and trainings are required to ensure that all agents in all paths learn the rewards of their arms accurately for each set in $\mathcal{I}_T$, before exploiting them.

Theorem 3 indicates that the regret increases exponentially in $D_h$ and is an $D_h$-th degree polynomial in $D_k$. While this makes CBMR-ind-N impractical in non-small world networks, CBMR-ind-N can achieve small regret in small world networks in which most agents are not directly connected to each other but all agents can be reached with a small number of hops. Because of the additional commission the relay agent gets, an item which $j$ will recommend when it is directly connected to agent $i$ may not be recommended by $j$ when it is connected via agent $j'$. This results in sub-optimality compared to the case when all agents are connected. In Section VI we numerically compare the effect of the agents’ commissions on the performance.

More refined versions of Theorem 3 can be derived if we focus on specific networks. One interesting network structure is when there is a monopolist agent which is directly connected to all of the other agents, while other agents are only directly connected to the monopolist agent. Corollary 1 gives the regret bound for agent $i$ when it is the monopolist agent, and Corollary 2 gives the regret bound for agent $i$ when it is not the monopolist agent.

Corollary 1: When agent $i$ is the monopolist agent, if it runs CBMR-ind-N with $D_{2,n}(t) = (\frac{F_{max}}{n}) t^{2\alpha/(3\alpha + d)} \log t$ and everything else remaining the same as in Theorem 3, it will achieve $R_{\alpha}G_{\alpha}(M,E)(T) = O \left( (F_i + (M - 1)N) T^{\frac{2\alpha + d}{3\alpha + d}} \right)$.

Corollary 2: When agent $i$ is a non-monopolist agent, if it runs CBMR-ind-N with $D_{2,n}(t) = (\frac{F_{max}}{n}) t^{2\alpha/(3\alpha + d)} \log t$ and everything else remaining the same as in Theorem 3, it will achieve $R_{\alpha}G_{\alpha}(M,E)(T) = O \left( (F_i + N) M^{2N} T^{\frac{2\alpha + d}{3\alpha + d}} \right)$. Corollaries 1 and 2 imply that the learning is much faster, and hence the regret is much smaller when an agent has direct connections with more agents. This is because agent $i$ learns directly about the purchase probabilities of items in agent $j$’s inventory when it is directly connected to it, while the learning is indirect through a relay agent otherwise.

VI. NUMERICAL RESULTS
A. Description of the Data Set

The Amazon product co-purchasing network data set includes product IDs, sales ranks of the products, and for each product the IDs of products that are frequently purchased with that product. This data is collected by crawling the Amazon website and contains 410,236 products and 3,356,824 edges between products that are frequently co-purchased together. We simulate CBMR-ind and CBMR-d using the following distributed data sets adapted based on Amazon data. For a set of $N_1$ chosen products from the

\[+ T^{\frac{2\alpha + d}{3\alpha + d}} \left( 4N(Y_R(L_d^\alpha/2 + 1) + \frac{Y_{\alpha/2} d}{(2\alpha + d)/(3\alpha + d)} + Y^{\alpha/2} l \log T \right) \]

\[+ T^{\frac{2\alpha + d}{3\alpha + d}} 2d Y_R(2 |J_t| \beta_2 + |J_t| (M - 1) N),\]

i.e.,
\[R_{\alpha}G_{\alpha}(M,E)(T) = O \left( Z_T^{2\alpha + d} \log T \right),\]

where $Z_T := |J_t| + (M - 1) \sum_{z=1}^{N} (D_i z^d F_{max}).$
and CBMR-ind are shown in Table III for agent 1. We have $|C_1| = 58$ and $|J_1| = 14$.

We see that the total reward of CBMR-ind is higher than the total reward of CBMR-d for this case. This is due to the fact that CBMR-d spends more than double the time CBMR-ind spends in training and exploration phases since it trains and explores each action in $L_1$ separately. The time averaged regrets of CBMR-d and CBMR-ind are given in Fig. 5. It is observed that CBMR-ind performs better than CBMR-d in all time slots, and the time averaged regret goes to 0. From these results it seems that CBMR-ind is a good alternative to CBMR-d even for group-dependent purchase probabilities.

C. Effect of commission on the performance

In this subsection we numerically simulate the effect of commissions $c_{i,j}$ that agent 1 charges to other agents on the total reward of agent 1. We consider CBMR-ind for independent purchase probabilities. We assume that agent 1 has one of the frequently co-purchased items for context $x^*$, while agent 3 has the other frequently co-purchased item. The total reward of agent 1 as a function of the commissions $c_{1,2} = c_{1,3} = c$ is given in Table IV. We note that there is no significant difference in the total reward when the commission is from 0 to 0.1. This is because 0.1 commission is not enough to incentivize agent 1 to recommend other agent’s items to its users. However, for commissions greater than 0.1, the optimal policy recommends the two frequently co-purchased items together, hence agent 1 learns that it should get recommendations from agent 3. Therefore, after 0.1 the total reward of the agent is increasing in the commission. Another remark is that $c = 0$ corresponds to the case when agent 1 is not connected to the other agents. Therefore, this example also illustrates how the rewards change as network connectivity changes. Since prices of items are set to 1 in this section, the commission agent 1 charges can increase up to 1. But if the prices are different, then the commission cannot exceed the recommended item’s price. In theory, in order to maximize its total reward from its own users, agent $i$ can adaptively select its commissions $c_{i,j}$, $j \in M_{-i}$ based on what it learned about the purchase probabilities. CBMR-d and CBMR-ind can be modified to adaptively select the commissions. Due to the limited space, we leave this as a future research topic.

D. Effect of the set of items of each agent on the performance

In this subsection we compare three cases for independent purchase probabilities when agents use CBMR-ind. In C-1 agent 1 has both items that are frequently co-purchased in context $x^*$, in C-2 it has one of the items that is frequently co-purchased in context $x^*$, and in C-3 it has none of the items that are frequently co-purchased in context $x^*$. The total reward of agent 1 for these cases is 17744, 14249 and 9402 respectively, while the total expected reward of the optimal policy is 20000, 15000 and 10000 respectively. Note that the total reward for C-3 is almost half of the total reward for C-1 since the commission agent 1 gets for a frequently co-purchased item is 0.5. The time averaged regret of CBMR-ind for all these cases is given in Figure 6. We see that the convergence rate for C-1 is slower than C-2 and C-3. This is due to the fact that in all of the training slots in C-1 a suboptimal set of items is recommended, while for C-2 and C-3 in some of the training slots the optimal set of items is recommended.

| Commission $c$ | 0   | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
|---------------|-----|-----|-----|-----|-----|-----|
| Reward (CBMR-ind) | 10471 | 10422 | 11476 | 12393 | 13340 | 14249 |

TABLE IV: Total reward of agent 1 as a function of the commission it charges to other agents.

In this paper we have presented a novel set of algorithms for multi-agent learning within a decentralized social network, and characterized the effect of different network structures on performance. Our algorithms are able to achieve sublinear regret in all cases, with the regret being much smaller if the user’s acceptance probabilities for different items are independent. This paper can be used as a groundwork for agents in many different types of networks to cooperate in

![Fig. 5: Time averaged regrets of CBMR-d and CBMR-ind for group-dependent purchase probabilities.](image)

![Fig. 6: Time averaged regret of CBMR-ind for independent purchase probabilities when agent 1 has both frequently co-purchased items (C-1), only one of the frequently co-purchased items (C-2) and none of the frequently co-purchased items (C-3).](image)

VII. Conclusion
a mutually beneficial manner, from companies, charities, and celebrities who wish to generate revenue to artists, musicians, and photographers who simply want to have their work published. By cooperating in a decentralized manner and using our algorithms, agents have the benefit of retaining their autonomy and privacy while still achieving near optimal performance.

References

[1] K.-C. Chen, M. Chiang, and H. Poor, “From technological networks to social networks,” Selected Areas in Communications, IEEE Journal on, vol. 31, no. 9, pp. 548–572, 2013.

[2] V. Krishnamurthy, “Quickest detection POMDPs with social learning: Interaction of local and global decision makers,” Information Theory, IEEE Transactions on, vol. 58, no. 8, pp. 5563–5587, 2012.

[3] V. Krishnamurthy and H. V. Poor, “Social learning and Bayesian games in multia gent signal processing: How do local and global decision makers interact?” Signal Processing Magazine, IEEE, vol. 30, no. 3, pp. 43–57, 2013.

[4] A. Jadbabaie, P. Molavi, A. Sandroni, and A. Tahbaz-Salehi, “Non-Bayesian social learning,” Games and Economic Behavior, vol. 76, no. 1, pp. 210–225, 2012.

[5] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” Automatic Control, IEEE Transactions on, vol. 54, no. 1, pp. 48–61, 2009.

[6] J. Langford and J. Zinkevich, “Efficient first-order methods for online learning and estimation,” Journal of Machine Learning Research, vol. 10, pp. 1209–1244, 2009.

[7] A. Slivkins, “Contextual bandits with similarity information,” in Proc. of the 24th Annual Conference on Learning Theory (COLT), July 2011.

[8] Y. Gai, B. Krishnamurthy, and R. Jain, “Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations,” in Proc. of the 51st Annual Allerton Conference on Communication, Control, and Computation, October 2013.

[9] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” Automatic Control, IEEE Transactions on, vol. 54, no. 1, pp. 48–61, 2009.

[10] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” Automatic Control, IEEE Transactions on, vol. 54, no. 1, pp. 48–61, 2009.

[11] Y. Gai, B. Krishnamurthy, and R. Jain, “Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations,” in Proc. of the 24th Annual Conference on Learning Theory (COLT), July 2011.

[12] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” Automatic Control, IEEE Transactions on, vol. 54, no. 1, pp. 48–61, 2009.

[13] J. Langford and J. Zinkevich, “Efficient first-order methods for online learning and estimation,” Journal of Machine Learning Research, vol. 10, pp. 1209–1244, 2009.

[14] Y. Gai, B. Krishnamurthy, and R. Jain, “Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations,” in Proc. of the 51st Annual Allerton Conference on Communication, Control, and Computation, October 2013.

[15] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” Automatic Control, IEEE Transactions on, vol. 54, no. 1, pp. 48–61, 2009.

[16] J. Langford and J. Zinkevich, “Efficient first-order methods for online learning and estimation,” Journal of Machine Learning Research, vol. 10, pp. 1209–1244, 2009.

[17] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” Automatic Control, IEEE Transactions on, vol. 54, no. 1, pp. 48–61, 2009.

[18] J. Langford and J. Zinkevich, “Efficient first-order methods for online learning and estimation,” Journal of Machine Learning Research, vol. 10, pp. 1209–1244, 2009.

[19] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” Automatic Control, IEEE Transactions on, vol. 54, no. 1, pp. 48–61, 2009.

[20] J. Langford and J. Zinkevich, “Efficient first-order methods for online learning and estimation,” Journal of Machine Learning Research, vol. 10, pp. 1209–1244, 2009.

[21] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” Automatic Control, IEEE Transactions on, vol. 54, no. 1, pp. 48–61, 2009.

[22] J. Langford and J. Zinkevich, “Efficient first-order methods for online learning and estimation,” Journal of Machine Learning Research, vol. 10, pp. 1209–1244, 2009.

[23] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” Automatic Control, IEEE Transactions on, vol. 54, no. 1, pp. 48–61, 2009.

[24] J. Langford and J. Zinkevich, “Efficient first-order methods for online learning and estimation,” Journal of Machine Learning Research, vol. 10, pp. 1209–1244, 2009.

[25] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” Automatic Control, IEEE Transactions on, vol. 54, no. 1, pp. 48–61, 2009.

[26] J. Langford and J. Zinkevich, “Efficient first-order methods for online learning and estimation,” Journal of Machine Learning Research, vol. 10, pp. 1209–1244, 2009.

[27] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” Automatic Control, IEEE Transactions on, vol. 54, no. 1, pp. 48–61, 2009.