Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Online Collaborative Filtering on Graphs

Siddhartha Banerjee, Sujay Sanghavi, Sanjay Shakkottai

To cite this article:
Siddhartha Banerjee, Sujay Sanghavi, Sanjay Shakkottai (2016) Online Collaborative Filtering on Graphs. Operations Research 64(3):756-769. https://doi.org/10.1287/opre.2016.1508

OPERATIONS RESEARCH
Vol. 64, No. 3, May–June 2016, pp. 756–769
ISSN 0030-364X (print) | ISSN 1526-5463 (online)

http://dx.doi.org/10.1287/opre.2016.1508

Online Collaborative Filtering on Graphs

Siddhartha Banerjee
School of ORIE, Cornell University, Ithaca, New York 14853, sbanerjee@cornell.edu

Sujay Sanghavi, Sanjay Shakkottai
Department of ECE, The University of Texas at Austin, Austin, Texas 78705
{sanghavi@mail.utexas.edu, shakkott@mail.utexas.edu}

Existing approaches to designing recommendation systems with user feedback focus on settings where the number of items is small and/or admit some underlying structure. It is unclear, however, if these approaches extend to applications like social network news feeds and content-curation platforms, which have large and unstructured content pools and constraints on user-item recommendations. To this end, we consider the design of recommendation systems in content-rich setting—where the number of items and the number of item-views by users are of a similar order and an access graph constrains which user is allowed to see which item. In this setting, we propose recommendation algorithms that effectively exploit the access graph, and characterize how their performance depends on the graph topology. Our results demonstrate the importance of *serendipity* in exploration and how recommendation improves when the access graph has higher expansion; they also suggest reasons behind the success of simple algorithms like Twitter's latest-first policy. From a technical perspective, our model presents a framework for studying explore-exploit trade-offs in large-scale settings, with potentially infinite number of items. We present algorithms with competitive-ratio guarantees under both finite-horizon and infinite-horizon settings; conversely, we demonstrate that naive policies can be highly suboptimal and also that in many settings, our results are orderwise optimal.

Keywords: online recommendation; social networks; competitive analysis.
Subject classifications: networks/graphs: stochastic; information systems: analysis and design.
Area of review: Special Issue on Information and Decisions in social and Economic Networks.
History: Received July 2013; revisions received February 2015, April 2016; accepted April 2016. Published online in *Articles in Advance* May 13, 2016.

1. Introduction

The modern Internet experience hinges on the ability of content providers to effectively recommend content to users. Moreover, in many online platforms, user feedback often provides the best (and most scalable) guide to the value of a piece of content. This feedback is incorporated into the recommendations in different ways: Curation websites like Digg and Reddit promote "popular stories"—content that other users found interesting on viewing; social networks like Twitter and Facebook show each user a small selected subset of all the content generated by their friends/contacts, based on, among other things, feedback ("likes") from other users; in online advertising, ad quality, i.e., the uptake from previous impressions, is an important input in deciding the allocation of ads to impressions.

Any system that both recommends items to users and then leverages their feedback to improve the recommendations, faces an *exploration–exploitation* trade-off: should a user be shown a new item of unknown value, in the hope of benefiting future users? Or should she be shown an item that is already known to give her good value? This trade-off has been extensively studied in settings wherein the number of items is small or admits some underlying structure (see Section 6 for a discussion of this prior work). The resulting algorithms, however, are often quite complex; moreover, it is unclear if the assumptions in these settings extend to the applications we describe above. On the other hand, simple heuristics often seem to do very well in practical settings—for example, Twitter's *latest-first* policy, which shows items in reverse chronological order along with a small number of recommended items. Our work presents an alternative model and performance metric for studying recommender systems, which provide some surprising insights into the design of such algorithms, and posits reasons for the success of certain simple heuristics.

In particular, we consider the design of recommender systems in large-scale online settings that display three distinguishing features:

1. *Content richness*: settings where new content is created far faster than the rate users can consume content.

2. *Unstructured content*: settings where superficially related items (for example, articles from the same source) may have very different values, and thus the value of one item of content need not be predictive of the value of other items.

3. *Binding constraints on recommendations*: the presence of constraints on which set of items can be presented to which users—these can be encoded via a bipartite *access graph* between users and items, wherein a user can only

be shown items from her neighborhood. A complete graph thus encodes the absence of constraints.

We henceforth refer to settings exhibiting the above features as *content-rich settings*. A prototypical content-rich setting is a social network: Here, every user both creates and consumes content at comparable rates; thus, the content available for any user far exceeds what she can view and in fact is of the same order as the total content views across all users. Next, content posted on social networks exhibits high variability because of periodic trends, one-time events, etc.; in particular, knowing the quality of one piece of content need not imply that other content uploaded by the same user is of similar quality. Finally, constraints on recommendations are typically imposed by the social graph, whereby users can only be shown content uploaded by their friends. Another content-rich setting is a content-curation website, where items are classified into topics, with each topic comprising a rich and unstructured pool of content; moreover, users typically subscribe to a small set of topics, indicating that they are only interested in content related to these topics.

We capture the notion of content-richness via the following stylized model: a bipartite access graph between users and items constrains which items can be shown to which users. To model user-item dynamics, we first consider a finite-horizon setting, with a large static set of items and a fixed pool of users who arrive in a uniform random order; subsequently, we generalize this to a dynamic setting, where users arrive to the system following a Poisson process to consume content and, on the other hand, new content also arrives via a Poisson process and then departs after spending a fixed amount of time on the system. For each visiting user, the algorithm must recommend a subset of neighboring items. Each item has an associated value, which can be *arbitrary*, thus capturing the unstructured nature of the content. Furthermore, item values are *a priori* unknown to the algorithm—to learn them, the algorithm depends on feedback from users.

We say an item is *pre-explored* if the algorithm has presented it to at least one user (more generally, to some fixed number of users) before the current time. Now we model the notion of learning from feedback via the following *identifiability* condition: *for any user, we assume that the algorithm can identify the corresponding highest valued items from the set of pre-explored items*. A special case, which we focus on in our exposition, is where every user has the same value for each item, and this value is learned by the platform the first time the item is shown to a user. However, in Section 5 we discuss how our algorithms extend to settings where each item requires a finite number of views to estimate its value to within a multiplicative factor. Empirical observations (see Szabo and Huberman 2010 and Yang and Leskovec 2011) suggest that in large social networks/content-curation sites, *the popularity of an item can be reliably inferred by showing it to a small number of users*. Furthermore, work on *static* recommendation (Keshavan et al. 2010, Jagabathula and Shah 2011) also provides guarantees for learning item value from a few ratings under alternate structural assumptions; these observations tie in well with our model.

Although exploration–exploitation trade-offs for recommendation algorithms have been extensively studied in online recommendation literature, most work has considered the problem under the *multi-armed bandit* model and its many variants (Bubeck and Cesa-Bianchi 2008). Using these models, and in particular, using regret (i.e., additive loss against an adversary) as a performance measure, is problematic in the content-rich settings we are interested in. In particular, most bandit algorithms depend on exploring each item a minimum number of times before being able to reject suboptimal items. However, in content-rich settings where each user sees only a small fraction of all available content, presenting one item to several users may result in other items never being presented to anyone. We discuss the relation between our setting and the multi-armed bandit settings in more detail in Section 6.

An alternative performance measure for an algorithm is its *competitive ratio*—the ratio of the reward that the algorithm earns for an *arbitrary user* to the best available reward for that user. Our work uses competitive-ratio maximization as an objective, resulting in interesting insights into the design of recommendation algorithms for content-rich settings. Moreover, we also show that naive algorithms can have much worse competitive ratios compared to better designed (yet simple) algorithms.

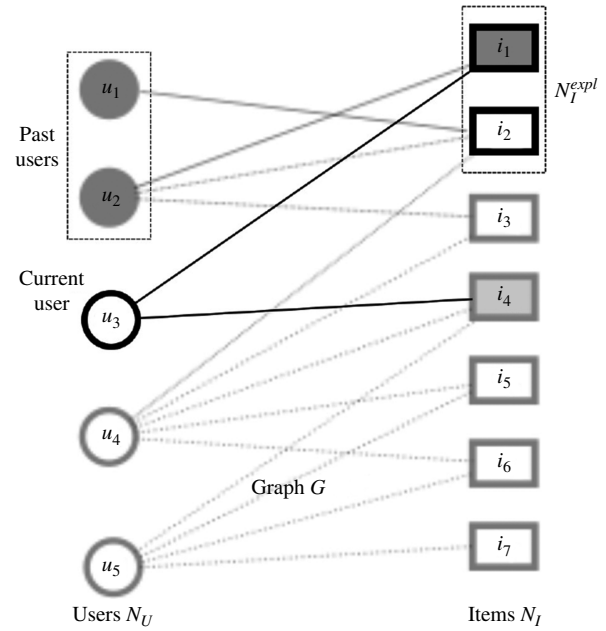
1.1. Summary of Our Contributions

We consider two settings—a finite-horizon setting and an infinite-horizon setting. The former can be used as a model for ad placement or content-curation settings, wherein items typically arrive in *batches*; the latter is more natural for applications like social network updates, which have continuous arrivals and departures.

Model: In the finite-horizon model (Section 2.1) we assume there is a bipartite access graph G between a (fixed) set of n_U users and n_I items—a user can view an item if and only if she is connected to it. Users arrive in a uniform random order and are presented with r item recommendations. Each item i has a value $V(i) \in \mathbb{R}_+$—we assume all item values are nonnegative and are the same for each user. The total reward earned by a user is the sum of rewards of presented items. The item values are *a priori* unknown to the algorithm but become known after an item is recommended for the first time. Thus, for any user, the algorithm can always sort the pre-explored items and identify the top r. The model is illustrated in Figure 1.

In the infinite-horizon model (Section 3.1), the underlying access graph G is between a finite set of users N_u and a finite set of *item classes* N_C. The system evolves in time, with user/item arrivals and departures. Each user makes multiple visits to the system according to an independent Poisson process; similarly, for each item class, individual

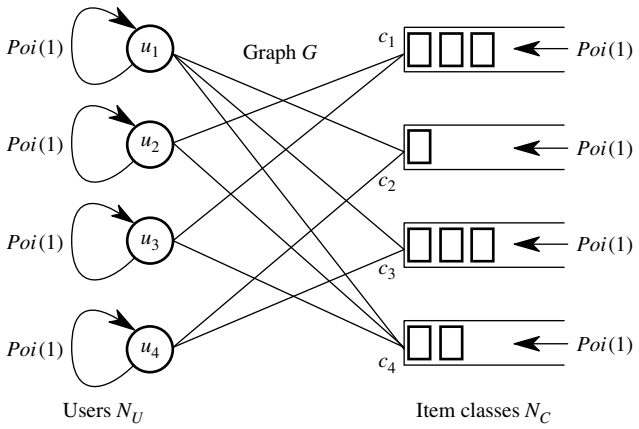
Figure 1. Illustration of the finite-horizon setting.



Notes. Items i_1 and i_4 (shaded) have $V(i) = 1$ and the rest all have value 0. Each user is presented one item ($r = 1$). Past users u_1 and u_2 have explored items i_2 and i_1, respectively. The recommendation algorithm needs to decide which item to recommend to user u_3: i_1 (exploit) or i_4 (explore).

items arrive according to an independent Poisson process. Items have arbitrary nonnegative values, which again are *a priori* unknown but become known when the item is recommended to a user for the first time. Furthermore, each item is available only for a fixed lifetime. To the best of our knowledge, ours is the first work that provides guarantees for online recommendation under Markovian dynamics but arbitrary item values. The infinite-horizon setting is illustrated in Figure 2.

Figure 2. Illustration of the infinite-horizon setting.



Notes. There are four users and four item classes. Users visit according to independent *Poisson*(1) (for short *Poi*(1)) processes, and similarly items are generated in each class according to an independent *Poisson*(1) process. Items disappear after a fixed lifetime. Items in the same class can have completely unrelated values.

Our results in this work can be summarized as follows:

1. *Exploration via balanced partitions*: In the finite-horizon setting, we present an algorithm based on picking unexplored items via *balanced semi-matchings* (or balanced item-partitions). We show this achieves a competitive-ratio guarantee of $\Omega(r/d^*(G))$ (Theorem 1), where r is the number of recommendations per user and $d^*(G)$ is the *minimum makespan* of the graph G.

2. *Exploration via inverse-degree sampling*: We present an alternative algorithm that eschews preprocessing and chooses items for exploration by randomly picking items with a probability *inversely proportional to their degree*. Inverse-degree sampling biases recommendations toward less popular content, thereby promoting a form of *serendipity*. We prove that this policy has a competitive-ratio guarantee of $\Omega(r/Z_{\max}(G))$ (Theorem 2), where $Z_{\max}(G)$ is a measure of graph nonregularity.

3. In the case of regular graphs, both the above algorithms have competitive-ratio guarantees of $\Omega(rn_I/n_U)$. Conversely, in the finite setting, we show that for all graphs, no algorithm can achieve a competitive ratio better than $O(rn_I/n_U)$ (Theorem 4).

4. *Exploration via uniform latest-item sampling*: In the infinite-horizon setting, we propose an algorithm in which users explore items drawn *uniformly from the set of latest items*—those that have not had the chance to be presented to any other user. All unexplored latest items are then discarded. In the setting where all arrival processes (of users/items) have equal rates, we prove that this policy achieves a competitive ratio of $\Omega(r/Z_{\max}(G))$ (Theorem 3).

5. Finally, we show that some natural policies—those that always exploit if sufficiently high-valued items are available or sample nodes uniformly or proportional to degree (or, in fact, proportional to any polynomial function other than 1/degree)—have 0 competitive ratio.

Finally in Section 5, we discuss how our results generalize to other settings—in particular, where item values can be approximately learned from multiple samples.

2. The Finite-Horizon Setting

We first consider a finite-horizon setting, where the number of users and items is fixed, and each user arrives once, following a uniform random order. This can be used to model content-curation settings like news aggregators (e.g., Google News), where a large number of articles appear together at the beginning of a day and expire at the end of the day—in the meantime, throughout the day, users appear uniformly at random.

2.1. System Model

Access graph: $G(N_U, N_I, E)$ represents the (given) bipartite access graph between users N_U and items N_I (with $|N_U| = n_U$ and $|N_I| = n_I$). For a user $u \in N_U$, we define its neighborhood as $\mathcal{N}(u) := \{i \in N_I \mid (u, i) \in E\}$ and degree $d_u = |\mathcal{N}(u)|$; similarly for item $i \in N_I$, we can define $\mathcal{N}(i)$

and d_i. Items are always present in the system, while users arrive to the system following a *uniform random permutation*.

Item exploration: Each item has an associated nonnegative real value $V(i) \in \mathbb{R}_+$, which is the same for all users. $V(i)$ is *a priori* unknown to the recommendation algorithm and is revealed the first time the item is presented to a user. Upon arrival, a user is presented a set of r items from $\mathcal{N}(u)$. We define N_I^{expl} to denote the set of *pre-explored items* at any instant, i.e., those that have been presented to at least one user in the past. We assume that $N_I^{\text{expl}} = \varnothing$ at the start; however, our competitive-ratio bounds continue to hold for any initial N_I^{expl}.

Objective: A recommendation algorithm $\mathcal{A}$ presents each arriving user u with r items $\{i_1^{\mathcal{A}}(u) \dots i_r^{\mathcal{A}}(u)\}$. For a given item-values vector V, the total reward earned by u under algorithm $\mathcal{A}$ is thus given by $R_r^{\mathcal{A}}(u) = \sum_{k=1}^{r} V(i_k^{\mathcal{A}}(u))$. Suppose the r highest-valued neighboring items for u are denoted as $\{i_1^*(u) \dots i_r^*(u)\}$—then we define the *optimal reward* $R_r^*(u) = \sum_{k=1}^{r} V(i_k^*(u))$. Now we define the *competitive ratio* $\gamma^{\mathcal{A}}(G, r)$ (for algorithm $\mathcal{A}$, graph G and r-recommendations) as

$$\gamma^{\mathcal{A}}(G, r) = \inf_{V \in \mathbb{R}_+^{n_I}} \inf_{u \in N_U} \frac{\mathbb{E}[R_r^{\mathcal{A}}(u)]}{R_r^*(u)}. \tag{1}$$

The expectation in Equation (1) is both over random user arrivals as well as randomness in algorithm $\mathcal{A}$; note, however, that $R_r^*(u)$ is uniquely determined $\forall u$ given G and item values V. The competitive ratio thus captures a worst case guarantee for individual users over all nonnegative item-value vectors. Note that taking an infimum over user rewards rather than considering the cumulative reward (i.e., the sum over all users) results in a more stringent objective. This, however, is often more appropriate in a recommendation setting because it corresponds to a natural notion of *fairness*—it is a guarantee on the quality of experience for any user on the platform.

2.2. Exploration via Balanced Item-Partitions

Before presenting our algorithms, we give a brief intuition behind how they work. All the algorithms we propose share the following structure: given r slots to present items to an arriving user, we split them between *explore* and *exploit* slots uniformly at random. In the exploit slots, we present the highest-valued pre-explored items, while for the explore slots, we present previously unexplored items. The crucial ingredient that distinguishes the algorithms is the *policy for choosing these items*.

To understand the role of the exploration policy, consider the case where $r = 1$ (i.e., we recommend only a single item). Since the algorithm knows the value of pre-explored items, a sufficient condition for guaranteeing a good per-user competitive ratio is that *the exploration rule ensures that for any user, her most relevant item is explored before she arrives to the system*. If this holds, then the user gets shown this item with probability $1/2$ (if the algorithm chooses to exploit). This property turns out to be too strong a requirement; however, we show a milder condition—that the above property holding with a constant probability—can be achieved in many settings.

In our first algorithm, the rule for selecting exploration items is based on a preprocessing step to construct a *balanced partition*—where the item set N_I is partitioned into n_U sets, one for each user, in a way such that the partitions are *balanced*. Formally we have the following definition:

Definition 1 (Balanced Partition). Given graph G, a *semi-matching* $M = \{M(u)\}_{u \in N_U}$ is a *partition* of the item-set such that $M(u) \subseteq \mathcal{N}(u)\ \forall u \in N_U$ (i.e., each set $M(u)$ is a subset of the neighbors of user u). Given a semi-matching M, we define the load of user u as $d_M(u) = |M(u)|$. Then a *balanced item-partition* M is a solution to the optimization problem

$$d^*(G) = \min_{\{M:\,\text{semi-matching}\}} \left[\max_{u \in N_U} d_M(u) \right].$$

The above problem is known in different communities as the *minimum makespan* problem (Graham 1966) or *optimal semi-matching* problem (Harvey et al. 2003)—we henceforth refer to $d^*(G)$ as the makespan of graph G. Efficient algorithms are known for finding a balanced item partition, with a complexity of $O(m\sqrt{n}\log n)$ (Fakcharoenphol et al. 2010), where $m = |E|$, $n = n_U + n_I$.

Given a balanced item-partition generation routine, we define the Balanced Partition Exploration Algorithm, or BPExp, which can be summarized as follows: we preselect a balanced item partition as an *exploration schedule*; for each arriving user, we independently allocate each recommendation slot to be an explore or exploit slot with probability $1/2$; for exploration, we display items picked uniformly at random (without replacement) from the user's items in the balanced item partition; for exploitation, we display the most valuable available pre-explored items. Formally, the algorithm is given in Algorithm 1.

Algorithm 1 (BPExp: Exploration via balanced item partitions)

1: Generate a *balanced item partition* M of G. Initialize set of explored items $N_I^{\text{expl}} = \varnothing$.
2: **for** arriving user $u \in N_U$ **do**
3: Choose $R_1(u) \sim Binomial(r, \frac{1}{2})$ slots for *exploration* and the rest $R_2(u) = r - R_1(u)$ slots for *exploitation*.
4: {**Exploration**}: Choose $R_1(u)$ items from the set $M(u)$ uniformly at random, without replacement.
5: {**Exploitation**}: Recommend the $R_2(u)$ highest-valued items from $\mathcal{N}(u) \cap N_I^{\text{expl}}$.
6: Update N_I^{expl} by adding the $R_1(u)$ items explored by u.
7: **end for**

THEOREM 1. *Given graph G, reward-function $V(i)$ and uniformly random user arrival pattern, using the BPExp algorithm* (*Algorithm 1*) *we get*

$$\gamma^{\text{BPExp}}(G, r) \geqslant \frac{1}{8} \min\left\{\frac{r}{d^*(G)}, 1\right\}.$$

PROOF OUTLINE. Consider the case with $r = 1$. Now for any user u, under the BPExp Algorithm, either she is shown her highest-valued item $i^*(u)$ via exploration or via exploitation. For the latter, the crucial observation is that *all of the following events must occur*:

1. The user $v(\neq u)$ responsible for exploring $i^*(u)$ arrives before u to the system,
2. BPExp chooses to use v's slot for exploration, and moreover, picks $i^*(u)$ for display, and
3. BPExp chooses to use u's slot for an exploitation.

By symmetry in user arrivals, the *first event occurs with probability* 1/2; subsequently, the definition of the BPExp algorithm allows the probability of the latter to be bounded. Finally, the probability of recommending $i^*(u)$ to u via exploitation is shown to be a lower bound for the probability of recommending $i^*(u)$ via exploitation. Combining these facts, we get the result. The complete proof is given in Section 7. □

Remarks:

- An immediate corollary of this result is that given *any graph G that contains a perfect matching*, the competitive ratio guaranteed by the BPExp algorithm is $\min\{\frac{r}{8}, \frac{1}{8}\}$. More generally, if G is a bi-regular graph (i.e., all nodes in N_U have the same degree, and similarly all nodes in N_I) with $n_I \geqslant n_U$, then $\gamma^{\text{BPExp}}(G, r) \geqslant rn_U/(8n_I)$.
- BPExp guarantees a linear scaling with r. However, note that we compare the reward earned by BPExp to the optimal reward for r recommendations. In settings where there are $\Omega(r)$ high-valued items, the optimal reward scales linearly with r—in such cases, BPExp's reward scales *quadratically*. In Section 4, we show that linear scaling of $\gamma(G, r)$ with r is in fact the best achievable by any algorithm.
- Consider a graph, where each user is connected to $d^*(G)$ items of degree 1—in this case, it is clear that the best possible competitive ratio is $r/d^*(G)$. This example is somewhat trivial as it offers no scope for using feedback—at the other extreme, in Section 4, we show that no algorithm can have a better competitive ratio than $rn_U/(2n_I)$ in the *complete bipartite graph*, where $d^*(G) = n_I/n_U$. The above theorem shows that, on the other hand, $\Omega(r/d^*(G))$ *is achievable in all graphs*.

2.3. Exploration via Inverse-Degree Sampling

Although it has a good competitive ratio, BPExp has several drawbacks associated with preprocessing to produce a balanced item partition—it is computationally expensive for large graphs, requires extensive centralized computation, and has to be updated if the network changes. We now present an alternative approach that avoids these problems (at the cost of lower performance) by using a *distributed and dynamic* exploration policy. The main idea is that a user, upon arrival, picks a neighboring item for exploration with a probability *inversely proportional to the degree of the item*. This can be done with minimal local knowledge of the graph (in fact, the degree information is often publicly available, e.g., followers on Twitter, friends on Facebook/Google+). The resulting competitive-ratio bounds, though, are weaker—in particular, the makespan $d^*(G)$ is now replaced by a quantity $Z_{\max}(G)$, defined as

$$Z_{\max}(G) := \max_{u \in N_U} Z(u), \quad \text{where } Z(u) := \sum_{i \in \mathcal{N}(u)} d_i^{-1}.$$

Note that $Z(u)$ is the normalization in inverse-degree sampling; i.e., when all neighboring items are unexplored, then user u samples item i with $p_{ui} = d_i^{-1}/Z(u)$.

One technical issue with inverse degree sampling is that when a user explores some $k > 1$ item, then it is difficult to derive the probability of sampling k neighboring items without replacement with probability proportional to their degrees. To avoid this problem, we instead first partition the neighboring items of *each* user into r sets and then sample one item from each set. We do so as follows:

DEFINITION 2 (GREEDY NEIGHBORHOOD PARTITIONING). For user u, given neighboring-items set $\mathcal{N}(u)$ with degrees $\{d_i\}$, sort the items in descending order of d_i^{-1} and generate partition $P_u = \{P_u^1, P_u^2, \ldots, P_u^r\}$ by iteratively assigning each item to the set $P_u^{k^*}$, where $k^* = \arg\min_{k \in [r]} \{\sum_{i \in P_u^k} 1/d_i\}$.

Note that the item partitioning is performed separately for each user—it is not a centralized operation. Given this preprocessing routine, we define the Inverse-Degree Exploration Algorithm, or IDExp, as follows:

Algorithm 2 (IDExp: Exploration via inverse-degree sampling)

1: **for** arriving user $u \in N_U$ **do**
2: Generate item-set partition $P_u = \{P_u^1, P_u^2, \ldots, P_u^r\}$ using Greedy Neighborhood Partitioning.
3: Choose $R_1(u) \sim Binomial(r, \frac{1}{2})$ *exploration* slots, and $R_2(u) = r - R_1(u)$ *exploitation* slots.
4: {**Exploration**}: Select R_1 sets without replacement from P_u, and from each selected set P_u^1, pick one item i with probability proportional to d_i^{-1} (ignoring whether or not i is pre-explored).
5: {**Exploitation**}: Recommend the $R_2(u)$ highest-valued items from $\mathcal{N}(u) \cap N_I^{\text{expl}}$.
6: Update N_I^{expl} by adding the $R_1(u)$ items explored by u.
7: **end for**

THEOREM 2. *Given graph G, reward-function $V(i)$ and uniformly random user arrivals, the IDExp algorithm* (*Algorithm 2*) *for recommending r items guarantees*:

$$\gamma^{IDExp}(G, r) \geqslant \frac{1}{4e} \min\left\{\frac{r}{2Z_{\max}(G)}, 1\right\}.$$

PROOF OUTLINE. Note that for any item with degree d, if each of its neighboring users tries to explore it with probability d^{-1}, then it is explored once on average—we can build on this idea to bound the probability that each item is explored by some user. From the point of view of any user u, its top item(s) are explored with some constant probability—further, because of random dynamics, there is a constant probability that the user arrives after the items are explored. We provide the complete proof in Section 7.2. □

Remarks:

• Compared to Theorem 1, the above guarantee is weaker by a factor of $d^*(G)/Z_{\max}(G)$ (ignoring constants). The two quantities, however, are almost equal for bi-regular graphs (in particular, $Z_{\max}(G) = n_I/n_U$, and $d^*(G) = \lceil n_I/n_U \rceil$).

• In general, we have $d^*(G) \leqslant Z_{\max}(G)$. However, there is no $O(1)$-bound in the other direction, and one can construct graphs where $Z_{\max}(G)/d^*(G)$ is $\Omega(n_I)$. This shows that the IDEXP algorithm performs best when the graph is close to regular but may deteriorate with increasing nonregularity.

• The fact that $Z_{\max}(G)$ is large because of nonregularity can result in the above bound being weak; however, in real-world social network graphs, the performance of the IDEXP algorithm is often much better than the bound. This is because the above bound is for the *worst-case node*; in real-world graphs, removing a few nodes often improves $Z_{\max}(G)$ by a large amount.

• It is somewhat nonintuitive to explore items with a probability *inversely proportional* to its degree—for example, if an item has the same value for all neighboring users (i.e., $V(u,i) = V(i)$), then not exploring a high-degree item with a high value may seem costly. However, in Section 4, we show that inverse-degree randomization is the *only competitive approach* in the following strong sense: any algorithm that explores item i with a probability proportional to $d_i^{-1\pm\epsilon}$ has 0 competitive ratio.

3. The Infinite-Horizon Setting

3.1. System Model

We now consider a setting where the *system evolves in time with user/item arrivals and departures*—this is a more natural model for social-network news feeds and some content-curation sites like Digg/Reddit, where content is posted in a more continuous manner.

Access graph: We are given an underlying access graph $G(N_U, N_C, E)$ between *users* N_U and *item classes* N_C (with $|N_U| = n_U$, $|N_C| = n_C$). For example, for the problem of generating news feeds in social networks, the access to user-generated content is restricted by the "follower" graph—a user can only see updates from people whom she follows. Each user visits the platform periodically to view updates from people she follows; correspondingly, users also post updates from time to time.

User/item dynamics: We assume the system evolves in continuous time. Each user generates a series of *visit events* via an independent Poisson process of rate 1. Equivalently, by the aggregation property of Poisson processes, all user visits together constitute a marked Poisson process of rate n_U—each visit is denoted by a unique index $s \in \mathbb{N}_+$ (i.e., a running count of user visits) and has an associated random mark $U(s)$ corresponding to the identity of the visiting user. A user in each visit is presented r items, chosen from available items in her neighboring item classes; she accrues rewards from these, provides feedback, and leaves instantaneously.

On the other side of the platform, we have an infinite stream of items, where for each item class, individual *items* arrive according to independent Poisson(1) processes. As with the users, the item streams together constitute a marked Poisson process of rate n_C—each arriving item is denoted by a unique index $i \in \mathbb{N}_+$ and has three associated parameters: item class $C(i)$, reward function $V(i)$, and arrival time $T(i)$. Also each item expires after a *fixed lifetime* τ, which we assume is the same for all items.

Reward function: Each item has an arbitrary value—this can depend on the item class, but not on the specific sample path. Formally, each item class c has an associated (infinite) sequence of positive values V_c, and the kth item of class c arriving in the system has associated value $V_c(k)$. Note that in any given sample path ω, the kth item of class c will have associated index $I_\omega \geqslant k$, depending on when it arrives in the system—by our previous notation, we have $C(I_\omega) = c$ and $V(I_\omega) = V_c(k)$.

We say an item sequence $\mathcal{J} = \{C(i), V(i), T(i), \tau\}_{i\in\mathbb{N}_+}$ is *valid* if it satisfies the above assumptions. At any time t, we define $N_I^{\text{expl}}(t)$ to be the set of pre-explored items (i.e., presented during at least one prior visit) currently in the system—for brevity, we suppress the dependance on t. As before, we assume that an item's value becomes known to the algorithm the first time it is presented.

Objective: Given a valid item sequence $\mathcal{J}$ and a visit s, we define $R_r^*(s)$ as the *optimal offline reward for visit* s; note that this is a random variable that depends on which user $U(s)$ corresponds to visit s, which items are in the system, etc. Similarly, for a given algorithm $\mathcal{A}$, we can define $R_r^{\mathcal{A}}(s)$. Combining these, we can define the *competitive-ratio* of algorithm $\mathcal{A}$ (given graph G, r recommendations) as

$$\gamma^{\mathcal{A}}(G,r) = \inf_{\mathcal{J}} \inf_{s\in\mathbb{N}_+} \mathbb{E}\left[\frac{R_r^{\mathcal{A}}(s)}{R_r^*(s)}\right],$$

where $\mathcal{J}$ represents the set of all valid item sequences.

3.2. Uniform Latest-Item Exploration

Given our results for the finite-horizon setting, a first idea for the infinite-horizon setting would be to apply the IDExp algorithm on the set of available items. This, however, does

not guarantee a positive competitive ratio as the number of unexplored items only decreases by at most r after each user visit and so subsequent users may have too many items to explore. The main idea in designing an exploration policy in this setting is that exploration should be *biased toward more recent items* while discarding older unexplored items. Let T_s and T_i be the arrival times of user visit s and item i, respectively. Next, for item $i \in \mathbb{N}$, we can define its *first neighbor* $S_1(i)$ as the index of the first user visit after T_i by a neighboring user (i.e., $S_1(i) = \min\{s \mid T_s \geqslant T_i,\ i \in \mathcal{N}(s)\}$). Correspondingly, for visit s, we can define the set of *latest items* $L(s)$ as the set of available items, for which it is the first neighbor (i.e., $L(s) = \{i \mid s = S_1(i),\ T_s < T_i + \tau\}$). Using these definitions, we now propose the Uniform Latest-Item Exploration Algorithm (ULExp):

Algorithm 3 (ULExp: Uniform latest-item exploration)
1: **for** session $s \in \mathbb{N}_+$ **do**
2: Determine $L(s)$, the set of *latest items*.
3: Choose $R_1(u) \sim Binomial(r, \frac{1}{2})$ *exploration* slots and $R_2(u) = r - R_1(u)$ *exploitation* slots.
4: {**Exploration**}: Pick R_1 items from $L(s)$ uniformly at random, and without replacement.
5: {**Exploitation**}: Recommend the $R_2(s)$ highest-valued neighboring items in N_I^{expl}.
6: Update N_I^{expl} by adding the $R_1(s)$ items explored by u.
7: **end for**
8: Remove items from N_I^{expl} when they leave the system.

Recall in Section 2, we defined $Z_{\max}(G) := \max_{u \in N_U} Z(u)$, where $Z(u) := \sum_{i \in \mathcal{N}(u)} d_i^{-1}$. We now have the following theorem for the competitive ratio of the ULExp algorithm:

THEOREM 3. *Given graph G, with both users and items arriving according to independent Poisson(1) processes, using the ULExp Algorithm, we have*

$$\gamma^{\text{ULExp}}(G, r) \geqslant \frac{r}{4(5Z_{\max} + 2)}.$$

PROOF OUTLINE. Using the reversibility of the Poisson processes, we can show that for any visit s, the average number of items in its latest-item set is bounded by $Z_{\max}$—to show this, we argue that for visit s and any neighboring item class c, the number of items in $L(s)$ of class c is one less than a $Geometric(d_i/(d_i + 1))$ random variable. This property suggests that uniform latest-item exploration ensures that any item is explored with high enough probability.

The technical difficulty arises because for any visit, we want such a guarantee for the corresponding highest-valued item for that visit—this item is selected based on the sample path and the sequence of rewards, which is *arbitrary*. Note that the rewards can affect which item is the highest valued—for example, if the sequence of item rewards is strictly decreasing, then the most valuable item is the oldest available item. Thus we cannot directly argue that the probability of the highest-valued item being explored is the same as that of a typical item. We present a more refined counting argument that accounts for this conditioning—the complete proof is given in Section 7. □

Remarks:

- We do not need to assume that all the Poisson processes have the same rate—in fact, in Section 7.3, we prove the result for general $\{\lambda_u, \lambda_c\}$. Note that we do not need to know these rates for the algorithm (though the competitive-ratio bound does depend on them).
- On the practical side, recommendation via showing the *latest* items, as done on the Twitter news feed, can be thought to be a form of uniform latest exploration—our result suggests that for good recommendation, this should be appropriately mixed with items that are popular ("trending"). Twitter has been experimenting with similar policies (the "While you were away" feature according to Rosania 2015).

4. Converse Results

We now present some converse results, which put in perspective the performance of our algorithms. We present two types of results—upper bounds on the competitive ratio *over all possible online algorithms* and negative results for *specific algorithms*. All results in this sections are for the finite-horizon setting.

Upper bounds: For our upper bounds, we consider G to be the complete bipartite access graph, with *binary rewards*, i.e., where each item has value $V(i) \in \{0, 1\}$. In this setting, we show that *no algorithm can achieve a competitive ratio better than* $n_U/(2n_I)$. Note that in this setting, $Z_{\max}(G) = n_U/n_I$, and $d^*(G) = \lceil n_U/n_I \rceil$—this shows that over all graphs, *our competitive-ratio bounds are tight up to constant factors*.

Formally we have the following theorem (the proof is in the electronic companion, available as supplemental material at http://dx.doi.org/10.1287/opre.2016.1508):

THEOREM 4. *Given any $\epsilon > 0$ and n_U, there exists a sufficiently large n_I such that for an $n_U \times n_I$ complete bipartite access graph, no recommendation algorithm can achieve $\gamma(G, 1) > n_U/(2n_I) + \epsilon$.*

Moreover, for r item recommendations, we show that *no algorithm can achieve superlinear scaling in r*:

COROLLARY 1. *Given any $\epsilon > 0$, n_U and r, there exists a sufficiently large n_I such that for a $n_U \times n_I$ complete bipartite access graph, no algorithm that is allowed to show at most r recommendations per user can achieve $\gamma(G, r) > rn_U/(2n_I) + \epsilon$.*

Negative results: In the IDExp algorithm, we chose items for exploration with a probability inversely proportional to their degree. This choice is somewhat nonintuitive—a more natural choice would seem to be to bias toward higher

degree items (as they can reward more users in the universal rewards setting). However, it turns out that a sampling distribution that is proportional to *any other polynomial in the degree* in fact has poor competitive ratio.

THEOREM 5. *Given any algorithm $\mathcal{A}$ that choses item i for exploration with probability proportional to $d_i^{-1\pm\epsilon}$ for any $\epsilon > 0$, then $\exists$ a sequence of graphs G_n (with $Z_{\max}(G_n) = 2$) and a corresponding collection of item values $V \in \mathbb{R}_+^{n_I}$ such that for $r = 1$, the competitive ratio $\gamma^{\mathcal{A}}(G_n, 1)$ goes to 0.*

The formal construction of G_n and proof is presented in our electronic companion. We note that for the graph family G_n used in the above result, IDExp achieves a competitive ratio of $r/(8e)$.

Finally, in all our algorithms, we split the recommendation slots between explore and exploit recommendations uniformly at random. It is not clear if we need this randomization—however, we can show that some simple intuitive schemes for deciding between explore and exploit are noncompetitive.

THEOREM 6. *Given a recommendation algorithm $\mathcal{A}$ with exploit/explore decision obeying*

- *Exploit-when-possible*: *Exploit whenever there is a non-0 valued available item, else explore*; *OR*
- *Exploit-above-threshold-t*: *Exploit when any pre-explored item gives a reward greater than t, for some fixed threshold $t > 0$.*

Next, independent of the choice of exploration policy, there exists a sequence of graphs G_n (with $Z_{\max}(G_n) = 2$) such that $\gamma^{\mathcal{A}}(G_n, 1)$ goes to 0 as $n \to \infty$.

Proof outline for converse results: All the above results are based on appropriate use of *Yao's minimax principle* (see Motwani and Raghavan 1997): essentially, this principle states that the competitive ratio of the *optimal deterministic algorithm* for a given *randomized input* (where the measure over inputs is known to the algorithm) is an upper bound for the competitive ratio. For example, for Theorems 4 and 1, we consider a setting where we choose an item i^* uniformly at random from N_I, and set $V(i^*) = 1$, and $V(i) = 0$ for all $i \neq i^*$—under this assumption, we show that no deterministic algorithm achieves better than the stated guarantees. Because of lack of space, we defer the complete proofs to our electronic companion.

5. Discussion and Extensions

5.1. Inferring Item Values from Multiple Ratings

Till now we have assumed that the algorithm knows the value of an item once it has been explored by at least one user. We now generalize this assumption by considering a setting where once an item is viewed by at least f users, its value is known to the algorithm within a multiplicative factor of $(1 \pm \delta(f))$ for some $\delta(f) \in [0, 0.5)$. The case considered so far corresponds to $f = 1$ and $\delta(1) = 0$—we now show how to modify our algorithms to handle this more general setting.

Unlike before, we now define an item to be pre-explored if it has been presented to at least f users (the set of pre-explored items is still denoted N_I^{expl}). Moreover, we modify our algorithms as follows:

- For every user u (or user visit s), the algorithm chooses $R_1(u) \sim Binomial(r, f/(f+1))$ exploration slots and $R_2(u) = r - R_1(u)$ exploitation slots.
- The exploration policies are modified to ensure each item gets explored up to f times (see below).
- For exploitation, the algorithm picks the top pre-explored items, based on the noisy estimates of $V(i)$.

Now suppose for a user u, its top item $i_1^*(u)$ has been explored by at least f users before u arrives, and u decides to exploit at least one item—then either item $i_1^*(u)$ is chosen or another item $i' \neq i_1^*(u)$ such that $(1 + \delta(f))V(i') \geqslant (1 - \delta(f))V^*(i_k^*(u))$. Consequently, the competitive ratio is reduced by a factor that is at most $1 - 2\delta(f)$.

We now briefly discuss how the exploration step is modified for the ULExp algorithm—the arguments for the finite-horizon setting are similar. Recall that in the ULExp algorithm, each user upon visiting explored items chosen uniformly from the set of latest items. We now modify this as follows: each item has a counter, initialized to zero, which is incremented whenever a neighboring user makes a visit (note: the item may or may not be presented during the visit). Once the counter reaches f, the item is declared to be pre-explored if it was presented to all its f visiting neighbors; else it is discarded. Finally, during each visit s, given $R_1(s)$ slots for exploration, the algorithm first chooses $R_1(s)$ numbers $\{l_1, l_2, \ldots, l_{R_1(s)}\}$ uniformly at random (with replacement) from $\{0, 1, \ldots, f-1\}$, and then, for each l_j, chooses an item uniformly at random from amongst neighboring items whose counter equals l_j. We henceforth refer to this as the ULExp-f algorithm; for $f = 1$, we get back the ULExp algorithm. Now we have the following theorem:

THEOREM 7. *In the infinite-horizon setting, given graph G, users and items arriving according to Poisson(1) processes and given that for any item, its value is known to within $(1 \pm \delta(f))$ after it is explored f times; then the ULExp-f algorithm guarantees*

$$\gamma^{\text{ULExp-}f}(G, r) \geqslant \frac{1}{(f+1)^{f+1}} \cdot \left(\frac{r}{5Z_{\max} + 2}\right)^f \cdot (1 - 2\delta(f))$$

Note that substituting $f = 1$ and $\delta(f) = 0$ gives us back the result from Theorem 1.

PROOF OUTLINE. By expanding the latest-item set to items that have seen $< f$ neighboring-user visits, we show that the (expected) size of the latest item set (for the top item corresponding to visit s) can now be bounded by $f \cdot (5Z_{\max} + 2)$. Thus for any visit, its top item is explored by *all of the first f neighboring users* with probability at least $((fr/(f+1)) \cdot$

$(1/f(5Z_{\max}+2)))^f$. Furthermore, the visiting user exploits with probability $1/(f+1)$, and if her top item $I_1^*(s)$ is pre-explored, it is either presented or substituted by another pre-explored item with a true value greater than $(1-2\delta(f))V(I_1^*(s))$. Combining these, we get the result. The formal proof is given in our electronic companion. □

5.2. More General Reward Models

Finally, we mention a few other extensions to our setting under which the competitive-ratio guarantees of our algorithms are preserved:

Personalization: Item i has intrinsic value $V(i)$ but gives neighboring user u a reward of $V(u,i)=f_{ui}(V(i))$, where $f_{ui}(\cdot)$ are (nonnegative, invertible) functions known to the algorithm. For example, we can have $V(u,i)=w(u,i)\cdot V(i)$, where $w(u,i)$ is a (known) preference weight user u has for item class i.

Ordinal preferences: In many settings, an item's value may not be quantifiable, but instead users may have ordinal rankings over items. In such settings, our identifiability assumption translates to being able to infer a ranking of pre-explored items from past feedback. This is reminiscent of the secretary problem (Babaioff et al. 2008) and also allows us to use learning-to-rank techniques (Jagabathula and Shah 2011).

Probabilistic predictability: In many cases, we may only be able to guarantee identification of the top item for a user *with some probability* P_{pred}; for example, for collaborative-filtering algorithms such as matrix completion (Keshavan et al. 2010). In this case, all our competitive-ratio bounds get scaled by P_{pred}.

6. Related Work

Static recommendation: Learning from feedback in large-scale recommender systems has been extensively studied in static settings (Keshavan et al. 2010, Jagabathula and Shah 2011)—these works, however, do not consider user-item dynamics and the explore-exploit trade-off. In contrast, our model captures that the algorithm must select what user data can be collected and that this selection affects the performance.

Standard bandit models: Bandit models refer to settings where choosing an action (or *arm*) from a set of actions both yields a reward as well as some feedback about the system, which then affects future control decisions. One such model is that of Markovian bandits (Gittins 1979), wherein each arm has an underlying state, and playing an arm results in a reward that depends on the state as well as a possible transition to another state (the other arms remaining unaffected). Both rewards and state transition matrices are assumed to be known, and the aim is to maximize the discounted sum reward. Obtaining optimal policies for Markovian bandits models is often difficult in complex settings.

In contrast, in finite-time bandit models, the focus is on controlling the additive loss (or *regret*) with respect to the best policy in a smaller class of allowed policies. These models were first proposed by Lai and Robbins (1985), and subsequent authors have considered several forms of these problems, with different assumptions on the reward-generation process. In most of these settings, prior work has supplied simple algorithms with near-optimal regret scaling (Bubeck and Cesa-Bianchi 2008). To obtain these regret bounds, however, algorithms typically need to sample each arm a minimum number of times, which scales sublinearly in the number of item views. In content-rich settings where the number of content pieces is of the same order as the number of content views, it is infeasible that all arms get shown more than a constant number of times. For example, consider a setting with n users, n items, and a complete bipartite access graph, where one item has a value of 1 and the rest 0, and each user is shown one item (i.e., $r=1$)—then a typical bandit algorithm (for example, the standard UCB algorithm of Auer et al. 2002) will sample all items at least once, thereby getting a competitive ratio of $\gamma=O(1/n)\to 0$. On the other hand, it is easy to get a constant competitive ratio in this setting (in particular, the optimal algorithm gets a competitive ratio of $1/2$, while our algorithms achieve a competitive ratio of $1/8$). Thus using existing bandit algorithms in these settings leads to 0 competitive ratios.

Structured bandit frameworks: Our work is more closely related to structured bandit frameworks, which incorporate constraints on user-item recommendations. For example, access graph constraints are incorporated in the contextual bandits (Dudík et al. 2011) and sleeping bandits (Kleinberg et al. 2010) models—however, here, the graph and user dynamics are assumed to be arbitrary and are not used to inform the algorithm design. Other variants on the bandit setting incorporate some underlying structure between the arms—for example, the linear bandits (Rusmevichientong and Tsitsiklis 2010) (where the reward is the inner product between a unknown vector of weights and an known quality vector, which can be chosen from some underlying metric space) and combinatorial bandits (Audibert et al. 2011) (a discrete analog of the linear bandit setting). Our model differs from these in various dimensions—most importantly, however, we focus on going beyond the typical bandit scaling to understand what is achievable in more unstructured and higher-dimensional settings.

Online matching and its variants: Our setting is also close in spirit to certain online optimization problems on graphs. Online auction design problems (Mehta et al. 2005) incorporate that an item can be displayed to multiple users, constrained by an underlying graph. However, in such problems the node weights (bids) are known, which often allows greedy algorithms to be constant-factor competitive. Related problems include the generalized secretary problem (Babaioff et al. 2008) and online transversal-matroid selection (Dimitrov and Plaxton 2008); both are based on

a bipartite graph between a "static set" and an "online set of nodes," where node weights (of online nodes) are automatically revealed upon arrival. In our problem, the reward function is unknown and becomes known only via exploration—this may affect many future users in a nontrivial manner.

7. Proofs of Competitive-Ratio Guarantees

We now prove our performance guarantees. Note that our algorithms all share the following structure:

- For each user, we divide the r slots uniformly at random between *explore* and *exploit* recommendations (i.e., we have $R_1(u) \sim Binomial(r, \frac{1}{2})$ exploration slots and $R_2(u) = r - R_1(u)$ for exploitation).
- For the exploitation step, the algorithm leverages the identifiability assumption; i.e., for any user the highest-valued pre-explored items can always be identified.
- For the exploration step, we use three different exploration policies (in Algorithms 1–3). These are designed to leverage the graph topology and randomness in user arrivals to ensure *balanced exploration*: for *any neighboring user-item pair*, we can lower bound the probability that the item is explored before the user arrives to the system.

Some notation: We use $\mathbb{R}_+$ to denote the sets of nonnegative reals ($x \geqslant 0$) and $\mathbb{N}_+$ for natural numbers ($x \in \{1, 2, \ldots\}$). For any $n \in \mathbb{N}_+$, we define $[n] = \{1, 2, \ldots, n\}$. We use the notation $a \vee b = \max\{a, b\}$, $a \wedge b = \min\{a, b\}$, and $\mathbb{1}_E$ to denote an indicator random variable for an event E (and $\mathbb{1}_E^{\mathscr{A}}$ as the indicator random variable for event E under algorithm $\mathscr{A}$).

7.1. A Preliminary Lemma

We first state and prove a structural lemma, which shows that for an algorithm to have a good competitive ratio, it should *ensure that every user is shown its highest-valued items with near-equal probability*.

For ease of exposition, we state the lemma for the finite-horizon setting. Given algorithm $\mathscr{A}$, recall that we define its competitive-ratio as

$$\gamma^{\mathscr{A}}(G, r) = \inf_V \inf_{u \in N_U} \frac{\mathbb{E}[R_r^{\mathscr{A}}(u)]}{R_r^*(u)}$$

(for graph G, and r-recommendations per user). Note that $R_r^*(u)$ *is not random* given G and reward-function V—the expectation is over randomness in user arrival-pattern as well as any randomization used by algorithm $\mathscr{A}$.

Lemma 1. *Given a graph G and algorithm $\mathscr{A}$ displaying r items, for any nonnegative reward function V (i.e., with $V(i) \geqslant 0\ \forall i$) and for any pair (u, i) where $i \in \mathcal{N}(u)$, let $\mathbb{1}_{u \to i}^{\mathscr{A}}$ be the indicator that user u is shown item i. Then we have*

$$\mathbb{E}[R_r^{\mathscr{A}}(u)] \geqslant \Big(\min_{k \in [r]} \mathbb{E}[\mathbb{1}_{u \to i_k^*(u)}^{\mathscr{A}}]\Big) R_r^*(u)$$

Consequently, from the definition of the competitive ratio, for any G and r, we have

$$\gamma^{\mathscr{A}}(G, r) \geqslant \inf_{u \in N_U} \min_{k \in [r]} \mathbb{E}[\mathbb{1}_{u \to i_k^*(u)}^{\mathscr{A}}]$$

Proof. Given nonnegative reward-function V, we can bound the reward earned by user u under $\mathscr{A}$ as

$$\begin{aligned}
\mathbb{E}[R_r^{\mathscr{A}}(u)] &\geqslant \mathbb{E}\Big[\sum_{k \in [r]} V(i_k^*(u)) \mathbb{1}_{u \to i_k^*(u)}^{\mathscr{A}}\Big] \\
&= \sum_{k \in [r]} V(i_k^*(u))\, \mathbb{E}[\mathbb{1}_{u \to i_k^*(u)}^{\mathscr{A}}] \\
&\geqslant \Big(\min_{k \in [r]} \mathbb{E}[\mathbb{1}_{u \to i_k^*(u)}^{\mathscr{A}}]\Big) \sum_{k \in [r]} V(i_k^*(u)) \\
&= \Big(\min_{k \in [r]} \mathbb{E}[\mathbb{1}_{u \to i_k^*(u)}^{\mathscr{A}}]\Big) R_r^*(u)
\end{aligned}$$

Note that the above inequality does not reference the reward function V. Moreover, recall that $\gamma^{\mathscr{A}}(G, r) = \inf_V \inf_{u \in N_U} (\mathbb{E}[R_r^{\mathscr{A}}(u)]/R_r^*(u))$. Taking infimum over $u \in N_U$, we get the result. □

In other words, to achieve a good competitive ratio, it is sufficient to *balance the welfare of all users*. In the infinite-horizon setting, a similar result holds with the infimum over users replaced with infimum over user visits $s \in \mathbb{N}_+$ and conditioning the expectation on the items currently in the system during visit s.

7.2. Performance Analysis: Finite-Horizon Setting

Some notation: we are given access graph $G(N_U, N_I, E)$ between n_U users and n_I items N_I. For a user $u \in N_U$, we define its neighboring item set $\mathcal{N}(u) \triangleq \{i \in N_I \mid (u, i) \in E\}$, $d_u = |\mathcal{N}(u)|$ (similarly for items). When a user arrives, the algorithm $\mathscr{A}$ recommends r items $\{i_1^{\mathscr{A}}(u), \ldots, i_r^{\mathscr{A}}(u)\} \subseteq \mathcal{N}(u)$—given reward-function V, the total reward earned by u is $\sum_{k=1}^r V(i_k^{\mathscr{A}}(u))$. Moreover, for a given user u, we can order her neighboring items (in decreasing order of values) as $\{i_k^*(u)\}_{k=1}^{d_u}$.

Performance analysis for BPExp algorithm:

Proof of Theorem 1. Suppose we are given a reward function V, balanced item partition M, and a *user arrival order* $\pi \in S_{n_U}$ (where S_{n_U} denotes the set of permutations of users N_U), which, as mentioned before, is assumed to be chosen *uniformly at random*.

From Lemma 1, we know that to obtain a lower bound on the competitive ratio, it is sufficient to bound $\inf_{u \in N_U} \min_{k \in [r]} \mathbb{E}[\mathbb{1}_{u \to i_k^*(u)}^{\text{BPExp}}]$. One problem in bounding this is that the event that a (pre-explored) item $i_k^*(u), k \in [r]$ is exploited by user u depends on how many higher-valued neighboring items of u are also pre-explored. To remove this dependence, we first propose a modified algorithm ModBPExp, whose reward is stochastically dominated from above by that of BPExp.

In ModBPexp, the choice of $R_1(u)$ and exploration rule is the same as in BPExp; the difference is in the exploitation rule. Note first that choosing $R_1 \sim Binomial(r, \frac{1}{2})$ is equivalent to independently choosing slots $\{1, 2, \ldots, r\}$ for either exploration or exploitation with probability 1/2. For a user u, suppose the slots we allocate for exploitation are $\{k_1, k_2, \ldots, k_{R_2}\} \subseteq [r]$—now, instead of recommending the top $R_2(u)$ pre-explored items (as in BPExp), ModBPExp recommends items $\{i^*_{k_1}(u), i^*_{k_2}(u), \ldots, i^*_{k_{R_2}}(u)\}$ if they are pre-explored, else it leaves the slot empty. A coupling argument shows that the resulting reward for ModBPExp is stochastically dominated by BPExp (since we may recommend a less valuable item, or even no item, while a higher valued item is present). We now show that ModBPExp gives a uniform lower bound *over all users* $u \in N_U$, and $\forall k \in [r]$, of $\mathbb{E}[\mathbb{1}^{\text{ModBPExp}}_{u \to i^*_k(u)}] \geqslant r/(8(d^*(G) \vee r))$ (we henceforth suppress the superscript).

First, for any user u and any item i, let p_{ui} denote the probability that u explores i under ModBPExp. For this to happen, we require that $(u, i) \in M$ *and* that i is one of the $R_1(u)$ items explored by u. From the definition of the ModBPExp algorithm and the makespan $d^*(G)$, if $(u, i) \in M$, then we have that u explores i with probability at least $(R_1(u)/d^*(G) \wedge 1)$—moreover, since $R_1(u) \leqslant r$ (and hence $R_1(u)/d^*(G) \wedge 1 \geqslant R_1(u)/(d^*(G) \vee r)$), we have that *for any neighboring user-item pair* (u, i), and independent of the user arrival pattern π,

$$p_{ui} \geqslant \left[\frac{R_1(u)}{d^*(G) \vee r}\right] \cdot \mathbb{1}_{\{(u,\, i) \in M\}}$$

Next, for any item $i \in N_I$, let $u_M(i)$ be the (unique) user connected to it in the item partition M. As before, let $\mathbb{1}^{\text{BPExp}}_{u \to i}$ denote the indicator that user u is recommended item i by BPExp. Now for any user $u \in N_U$, and any item $i^*_k(u), k \in [r]$ (i.e., any of the top r items for u), we want to bound $\mathbb{E}[\mathbb{1}^{\text{BPExp}}_{u \to i^*_k(u)}]$. To do so, we consider two cases: (i) when $u_M(i^*_k(u)) = u$ and (ii) when $u_M(i^*_k(u)) \neq u$. Note that these are mutually exclusive and completely determined given reward function V and item partition M:

(i) $u_M(i^*_k(u)) = u$ and u *is shown* $i^*_k(u)$ *via exploration*: This requires that u explores $i^*_k(u)$. Using the definition of $p_{u,\, i}$ we have that

$$\begin{aligned}\mathbb{E}[\mathbb{1}^{\text{BPExp}}_{u \to i^*_k(u)} \,|\, u_M(i^*_k(u)) = u] &= \mathbb{E}[p_{u,\, i^*_k(u)} \,|\, u_M(i^*_k(u)) = u] \\ &\geqslant \mathbb{E}\left[\frac{R_1(u)}{d^*(G) \vee r}\right] = \frac{r}{2(d^*(G) \vee r)}\end{aligned}$$

(ii) $u_M(i^*_k(u)) = w \neq u$ and u *is shown* $i^*_k(u)$ *via exploitation*: This requires that w arrive before u in arrival pattern π, w explores $i^*_k(u)$, and u exploits $i^*_k(u)$. We now lower bound these probabilities.

First, by a similar calculation as above, we have that the probability of w exploring $i^*_k(u)$ is $\geqslant (r/(2(d^*(G) \vee r)))$. Next, *since* π *is chosen uniformly at random from* S_{n_U}, we have that w arrives before u with probability 1/2. Finally, we need to bound the probability that u exploits $i^*_k(u)$—under ModBPExp, this is identical to the event that u's display slot k is chosen for exploitation, which happens with probability 1/2. Putting the three bounds together, we have

$$\begin{aligned}\mathbb{E}[\mathbb{1}_{u \to i^*_k(u)}] &= \mathbb{E}[\mathbb{1}_{\{u' \text{ arrives before } u\}} \mathbb{1}_{\{u' \text{ explores } i^*_k(u)\}} \mathbb{1}_{\{u \text{ exploits } i^*_k(u)\}}] \\ &\geqslant \frac{1}{2} \cdot \left(\frac{r}{2(d^*(G) \vee r)}\right) \cdot \frac{1}{2} = \frac{r}{8(d^*(G) \vee r)}. \qquad \square\end{aligned}$$

Performance analysis for IDExp algorithm: Next, we obtain the competitive ratio bound for IDExp; refer to Section 2.3 for details. We first need a lemma characterizing greedy neighborhood partitioning:

Lemma 2. *If* $\{P_u\}_{u \in N_U}$ *is generated by independently applying greedy neighborhood partitioning (Definition 2) for each user* $u \in N_U$; *then* $Z_{\max}(G, r, \{P\}) \triangleq \max_{u \in N_U} \max_{k \in [r]} \sum_{i \in P^k_u} d_i^{-1}$ *obeys*

$$Z_{\max}(G, r, \{P\}) \leqslant \frac{2Z_{\max}(G)}{r}.$$

Proof. First, note that by definition, $\sum_{i \in \mathcal{N}(u)} d_i^{-1}$ is bounded by $Z_{\max}(G)$ for all u. Further, $d_i^{-1} \leqslant 1$ for all i. Together, this implies that the *optimal* balanced neighborhood partition $\{\tilde{P}^k_u\}_{k \in [r]}$ for any user u has the property that $\max_{k \in [r]} \sum_{i \in \tilde{P}^k_u} d_i^{-1} \leqslant Z_{\max}(G)/r$. The above claim now follows from the existing result of Graham (1966), which shows that greedy set partitioning has an approximation ratio of 1/2. $\square$

Proof of Theorem 2. The proof follows a very similar line of arguments as the proof of Theorem 1. First, as before, we use Lemma 1 to claim that it is sufficient to show that *for all users* $u \in N_U$ and $\forall k \in [r]$, IDExp achieves a uniform bound: $\mathbb{E}[\mathbb{1}_{u \to i^*_k(u)}] \geqslant (1/(4e)) \cdot ((r/(2Z_{\max}(G))) \wedge 1)$.

As with BPExp, we again need to define a fictitious algorithm ModIDExp to simplify the dependence between explored items. First, suppose for each user u, greedy neighborhood partitioning returns partition $\{P^1_u, P^2_u, \ldots, P^r_u\}$. Now we define ModIDExp as follows: for user u, we allocate each of its r recommendation slots for exploration or exploitation with probability 1/2; if slot $k \in [r]$ is chosen for exploration, we pick an item i from P^k_u with probability proportional to d_i^{-1}, *irrespective of whether* i *was explored before*; if slot $k \in [r]$ is chosen for exploitation, then we recommend item $i^*_u(k)$ if it is pre-explored or else leave the slot empty. Note that the first two steps are identical to IDExp, while the third can be can be combined with a coupling argument to show that IDExp stochastically dominates ModIDExp in terms of rewards.

We henceforth focus on analyzing ModIDExp and as before define p_{ui} to be the probability that user u explores item i—from definition of IDExp, we have $p_{ui} \leqslant d_i^{-1}$. Let $Z \triangleq (2Z_{\max}(G)/r) \vee 1$. Lemma 2 shows that for every

user u, and every neighborhood partition $P_u^k, k \in [r]$, the quantity $\sum_{i \in P_u^k} d_i^{-1}$ is bounded by Z. Now for any user u, with $R_1(u) \sim Binomial(r, 1/2)$ explore slots, for any item i, we have that u explores i if all of the following occur: $i \in \mathcal{N}(u)$, the partition P_u^k containing i is picked for one of the $R_1(u)$ explore slots, and i is the item chosen from P_u^k—thus

$$p_{ui} \geqslant \frac{d_i^{-1}}{Z} \cdot \frac{1}{2} \cdot \mathbb{1}_{\{i \in \mathcal{N}(u)\}}$$

Now given reward function V and user arrival pattern π chosen uniformly at random from S_{n_U}, consider any user $u \in N_U$ with associated highest-valued r items $i_k^*(u), k \in [r]$. Consider item $i_k^*(u)$, and let A_t be the event that there are $t \in \{0, 1, \ldots, d_i - 1\}$ neighbors of $i_k^*(u)$ who arrive in the system before u in arrival pattern π; we denote these users as $\{a_k\}_{k=1}^t$. Conditioned on A_t, under the IDExp algorithm, $\mathbb{1}_{u \to i_k^*(u)} = 1$ iff $i_k^*(u)$ is explored by u, or it is explored by one of the t neighbors of $i_k^*(u)$ who arrived before u and exploited by u. Note that unlike for BPExp, these events are now not mutually exclusive. Now as before we have that for ModIDExp, the probability that $i_k^*(u)$ is exploited by u when it is pre-explored is at least 1/2. Thus we have (using i as shorthand for $i_k^*(u)$)

$$\begin{aligned}\mathbb{E}[\mathbb{1}_{u \to i_k^*(u)} \mid A_t] &\geqslant \frac{1}{2}\left[1 - (1 - p_{ui}) \prod_{k=1}^{t} (1 - p_{a_k i})\right] \\ &\geqslant \frac{1}{2}\Big[p_{a_1 i} + (1 - p_{a_1 i}) p_{a_2 i} + (1 - p_{a_1 i}) \\ &\qquad \cdot (1 - p_{a_2 i}) p_{3i} + \cdots + p_{ui} \prod_{k=1}^{t} (1 - p_{a_k i})\Big].\end{aligned}$$

From before, we know that for all w s.t. $i \in \mathcal{N}(w)$, we have $d_i^{-1}/(2Z) \leqslant p_{wi} \leqslant d_i^{-1}$. Now we have

$$\begin{aligned}\mathbb{E}[\mathbb{1}_{u \to i_k^*(u)} \mid A_t] &\geqslant \frac{1}{2} \sum_{k=0}^{t} \left(1 - \frac{1}{d_i}\right)^k \frac{d_i^{-1}}{2Z} \\ &= \frac{1}{2Z}\left(1 - \left(1 - \frac{1}{d_i}\right)^{t+1}\right)\end{aligned}$$

Since π is drawn uniformly at random from S_{n_U}, we have that $\mathbb{P}[A_t] = 1/d_i$. Thus

$$\begin{aligned}\mathbb{E}[\mathbb{1}_{u \to i_1(u)}] &\geqslant \frac{1}{d_i} \cdot \frac{1}{4Z} \sum_{t=1}^{d_i} \left(1 - \left(1 - \frac{1}{d_i}\right)^t\right) \\ &= \frac{1}{4Z}\left(\frac{1}{d_i} + \left(1 - \frac{1}{d_i}\right)^{d_i + 1}\right) \\ &\geqslant \frac{1}{4Z}\left(\frac{1}{d_i} + \frac{1}{e}\left(1 - \frac{1}{d_i}\right)^2\right),\end{aligned}$$

since $(1 - 1/d)^d \geqslant 1/e - 1/(ed)\ \forall d \in \mathbb{N}_+$. Thus we have

$$\mathbb{E}[\mathbb{1}_{u \to i_1^*(u)}] \geqslant \frac{1}{4Z}\left(\frac{1}{e} + \frac{1}{d_i}\left(1 - \frac{2}{e}\right) + \frac{1}{ed_i^2}\right) \geqslant \frac{1}{4eZ},$$

where $Z = (2Z_{\max}(G)/r) \vee 1$. This completes the proof. □

7.3. Performance Analysis: Infinite-Horizon Setting

Finally, we turn to the infinite-horizon setting. Recall that we have a graph between users N_U and item classes N_C, with user visits $x \in \mathbb{N}_+$, and items $i \in \mathbb{N}_+$. Each item i has item class $C(i)$, lifetime τ (same for all items), and a reward function $V(i)$. For each item class c, we define V_c to be an arbitrary sequence of reward functions such that the kth item of class c has value $V_c(k)$. Finally we define the competitive ratio of a recommendation algorithm over visits as $\gamma^{\mathcal{A}}(G, r) = \inf_V \inf_{s \in \mathbb{N}_+} \mathbb{E}[R_r^{\mathcal{A}}(s)/R_r^*(s)]$; note that now the optimal reward is a random variable.

To define the uniform latest item exploration strategy (ULExp), let $S_1(i)$ to be index of the first visit by a neighboring user $u \in \mathcal{N}(C(i))$ after item i arrives and before it expires. Complementary to this, for user visit s, we define the latest-item set $L(s)$ to be the set of items for which s is the first visit. Now ULExp is based on randomly picking $R_1(s) \sim Binomial(r, 1/2)$ items from $L(s)$ without replacement, for exploration during visit s. To get an intuition behind the competitive ratio bound for ULExp, note that for any *typical* item, the expected size of the latest-item set can be bounded by $2Z_{\max}(G) + 1$. This suggests that the probability that any typical item is explored is greater than $1/(2Z_{\max}(G) + 1)$—however, is insufficient to obtain our result because, for any given user, we are interested not in the latest-item set of a typical item but rather the latest-item set *as seen by its corresponding highest-valued items*. This is determined by reward function V, and the main technical challenge in the proof is to account for this dependence.

The main idea of our proof is as follows—for any visit s arriving at time T_s, we consider the arrivals in the time interval $[T_s - \tau, T_s]$—since each item has a lifetime of τ, the items in this interval are the only ones which matter. We argue that the statistics of these arrivals are unaffected by the reward function V, and further, using the ULExp scheme, we can control the probability with which a particular item is explored. For this, we first require the following combinatorial lemma:

LEMMA 3. *Given a uniform permutation of R red and B blue balls, for each $i \in [R]$ let $N_{\text{cons}}(i)$ be the length of the interval of consecutive red balls (bounded on either side by either a blue ball or a boundary) containing the ith red ball. Then we have*

$$\max_i \mathbb{E}[N_{\text{cons}}(i)] \leqslant \frac{4R}{B+1} + 2$$

Remarks: We defer the complete proof to our electronic companion—however, a brief note on why this result is nontrivial: The main difficulty in showing this bound is accounting for the boundary conditions. To get a sense for this, note that the maximum value of $\mathbb{E}[N_{\text{cons}}(i)]$ over i is not only greater than the expected number of consecutive red balls (which by symmetry is $R/(B+1)$) but also greater than the expected number of consecutive red

balls *as seen by a uniform random red ball*. For example, for $B = 1$, a simple calculation gives that the expected consecutive sequence seen by a random ball is $(1/R) \cdot (1/(R+1)) \cdot \sum_{k=0}^{R}(k^2 + (R-k)^2) = (2R+1)/3$—however, we prove that $\max_i \mathbb{E}[N_{\text{cons}}(i)]$ in this case is $(3R+1)/4$. On the other hand, using Jensen's inequality to exchange the max and the expectation does not give a strong enough bound—for example, when $B = R = n$, our bound gives $\max_i \mathbb{E}[N_{\text{cons}}(i)] \leqslant 6$, while a standard balls-in-bins argument gives $\mathbb{E}[\max_i N_{\text{cons}}(i)] = \Theta(\log n / \log\log n)$.

Proof of Theorem 3. From Lemma 1, we have that for any visit s:

$$\mathbb{E}[R_r^{\text{ULExp}}(s)] \geqslant \mathbb{E}\left[\min_{k\in[r]} \mathbb{E}[\mathbb{1}_{s\to I_k^*(s)}^{\text{ULExp}}] R_r^*(s)\right]$$

Here the inner expectation is over the randomness in the algorithm, and the outer expectation is over randomness in the sample path. We henceforth suppress the superscript.

To complete the proof, for any user visit s and for all $k \in [r]$, we need to show that the corresponding kth top item, denoted $I_k^*(s)$, satisfies $\mathbb{E}[\mathbb{1}_{s\to i_k^*(s)}^{\text{ULExp}}] \geqslant r/(4(5Z_{\max}(G)+2))$. We will in fact prove a more general result—suppose items of any item class c arrive at rate λ_c, and each user u visits at rate λ_u. We now obtain that $\mathbb{E}[\mathbb{1}_{s\to I_k^*(s)}] \geqslant r/(4(5Z+2))$, where

$$Z = \max_{u\in N_U} \sum_{c\in\mathcal{N}(u)} \frac{\lambda_c}{\sum_{u'\in\mathcal{N}(c)} \lambda_{u'}}.$$

When $\lambda_u = \lambda_c = 1\ \forall u, c$, we get the claimed result.

Now suppose we denote $|L(I_k^*(s))|$ to be the size of the latest-item set of the first-neighbor of item $I_k^*(s)$ (formally, in our notation, $L(S_1(I_k^*(s)))$—we use $L(I_k^*(s))$ as a shorthand for this). Then we have that the probability of $I_k^*(s)$ being explored by $S_1(I_k^*(s))$ under the ULExp policy is given by

$$\begin{aligned}&\mathbb{P}[I_k^*(s) \text{ is explored by } S_1(I_k^*(s))]\\ &\quad= \mathbb{E}\left[\frac{R_1(S_1(I_k^*(s)))}{|L(I^*(s))|}\right] \geqslant \frac{r}{2\,\mathbb{E}[|L(I^*(s))|]},\end{aligned}$$

where for the last inequality, we have used that $R_1(S_1(I_k^*(s)))$ is independent of $L(I_k^*(s))$ and further bounded it via Jensen's inequality. Further, via similar arguments as in Theorems 1 and 2, we have that

$$\begin{aligned}\mathbb{E}[\mathbb{1}_{s\to i_k^*(s)}] &\geqslant \frac{1}{2}\,\mathbb{P}[I_k^*(s)] \text{ is explored by } S_1(I_k^*(s))]\\ &\geqslant \frac{r}{4}\frac{1}{\mathbb{E}[|L(I_k^*(s))|]}. \qquad (2)\end{aligned}$$

Thus a lower bound on the competitive ratio involves upper bounding $\mathbb{E}[|L(I_k^*(s))|] = \mathbb{E}[\mathbb{E}[|L(s')| \mid I_k^*(s), S_1(i)]]$. Note that the conditioning depends on the bipartite graph G and also on the reward function sequence V; also, note that because of the conditioning on $I_k^*(s)$, we cannot write $\mathbb{E}[|L(I_k^*(s))|] = \mathbb{E}[|L(s')|]$ for some "typical" visit s'. Instead, we need to exactly characterize and bound the dependence on the graph and reward function. We do so as follows: given user s arrives at time T_s, we consider all sample paths of the process parametrized by three random variables:

- $\mathbf{I}_l(s) = \{I_l(s,c)\}_{c\in N_C}$ are the indices of the most recent items for each item class.
- $\mathbf{R}_s = \{R_c\}_{c\in N_C}$ are the number of items of each class that arrived in the interval $[T_s - \tau, T_s]$.
- $\mathbf{B}_s = \{B_u\}_{u\in N_U}$ are the number of visits by each user in the same time interval.

Since all items have a lifetime of τ, it is clear that $I_k^*(s)$ must have arrived in the interval $[T_s - \tau, T_s]$. Further, given $\mathbf{I}_l(s), \mathbf{R}_s$, $I_k^*(s)$ is completely determined, and so we can now define $c^* = C(I_k^*(s)) \in \mathcal{N}(U(s))$ and i^* to be the index (or position) of $I_k^*(s)$ among all the items of class c^* arriving in the interval (i.e., $i^* \in \{1, 2, \ldots, R_{c^*}\}$). The crucial observation is that conditioning on $\{\mathbf{R}_s, \mathbf{B}_s\}$ item/user visits arriving in the interval implies that *any ordering of these* $\{\mathbf{R}_s, \mathbf{B}_s\}$ *events is equally likely*—further, this remains unchanged given $\mathbf{I}_l(s)$. This now puts us in a position where we can use Lemma 3.

Recall that we want an upper bound on $\mathbb{E}[|L(I_k^*(s))|]$—as we argued, given the conditioning presented above, this corresponds to the item of class c^* with index i^* among all items of that class in the interval $[T_s - \tau, T_s]$. Now we define $L(I_k^*(s), c)$ to be the number of latest items of item class c encountered by $S_1(I_k^*(s))$—thus $L(I_k^*(s)) = \sum_{c\in\mathcal{N}(u(s))} L(I_k^*(s), c)$. Note that the first visit of $I_k^*(s)$ could correspond to any neighboring user; from Lemma 3, we thus have

$$\mathbb{E}[|L(I_k^*(s), c^*)|] \leqslant \mathbb{E}\left[\frac{4R_{c^*}}{1 + \sum_{u\in\mathcal{N}(c^*)} B_u} + 2\right] + \mathbb{E}[L_{\text{prior}}],$$

where L_{prior} is the number of additional items that arrived before $T_s - \tau$ but were potentially in $L(I_k^*(s))$. Now note that $R_{c^*} \sim Poisson(\lambda_{c*}\tau)$, and further, for its neighboring users, $\sum_{u\in\mathcal{N}(c^*)} B_u \sim Poisson(\sum_{u\in\mathcal{N}(c^*)} \lambda_u \tau)$ (since they are a sum of independent Poisson processes). Thus we have

$$\begin{aligned}&\mathbb{E}[|L(I_k^*(s), c^*)|]\\ &\leqslant \mathbb{E}[4R_{c^*}]\,\mathbb{E}\left[\frac{1}{\sum_{u\in\mathcal{N}(c^*)} B_u + 1}\right] + 2 + \frac{\lambda_{c^*}}{\sum_{u\in\mathcal{N}(c^*)} \lambda_u}\\ &\leqslant \frac{4\lambda_{c^*}\tau(1 - \exp(-\tau\sum_{u\in\mathcal{N}(c^*)} \lambda_u))}{\sum_{u\in\mathcal{N}(c^*)} \lambda_u \tau} + 2 + \frac{\lambda_{c^*}}{\sum_{u\in\mathcal{N}(c^*)} \lambda_u}\\ &\leqslant \frac{5\lambda_{c^*}}{\sum_{u\in\mathcal{N}(c^*)} \lambda_u} + 2\end{aligned}$$

To complete the proof, we need to get a bound on $\mathbb{E}[|L(I_k^*(s), c)|]\ \forall\, c \neq c^*$. Consider any visit s' in the interval $[T_s - \tau, T_s]$—conditioning only on $\{\mathbf{R}_s, \mathbf{B}_s\}$, it follows from symmetry that

$$\mathbb{E}[|L(s')| \,|\, \{\mathbf{R}_s, \mathbf{B}_s\}] \leqslant \sum_{c\in\mathcal{N}(U(s'))} \frac{R_c}{1 + \sum_{u\in\mathcal{N}(c)} B_u} + \frac{\lambda_c}{\sum_{u\in\mathcal{N}(c)} \lambda_u},$$

where the second term accounts for the arrivals prior to $T_s - \tau$. In our case, we are interested in $L(S_1(I_k^*(s)), c)$—so we need to take into account the condition that visit s' saw $I_k^*(s)$ in its latest-item set. However, in case of items of class c^*, we have that the number of items in the latest-item set of $S_1(I_k^*(s))$ can at most increase by a factor of 4. Thus, via similar arguments as above, we can show that

$$\mathbb{E}[|L(I_k^*(s), c)|] \leqslant \mathbb{E}\left[\frac{4R_c}{1+\sum_{u\in\mathcal{N}(c)} B_u}\right] + \frac{\lambda_c}{\sum_{u\in\mathcal{N}(c)} \lambda_u},$$

and thus we have for all user visits s and for all $k \in [r]$,

$$\begin{aligned}
&\mathbb{E}[|L(I_k^*(s))|] \\
&\quad\leqslant \mathbb{E}[|L(I_k^*(s), c^*)|] + \mathbb{E}\left[\sum_{c\in\mathcal{N}(S_1(I_k^*(s))),\, c\neq c^*} |L(I_k^*(s), c^*)|\right] \\
&\quad\leqslant 2 + \mathbb{E}\left[\sum_{c\in\mathcal{N}(S_1(I_k^*(s)))} \frac{4R_c}{1+\sum_{u\in\mathcal{N}(c)} B_u} + \frac{\lambda_c}{\sum_{u\in\mathcal{N}(c)} \lambda_u}\right] \\
&\quad\leqslant 2 + \max_{u\in N_U} \sum_{c\in\mathcal{N}(u)} \left[\frac{5\lambda_c}{\sum_{u'\in\mathcal{N}(c)} \lambda_{u'}}\right] = 2 + 5Z
\end{aligned}$$

Now we can substitute this in Equation (2), to get the result. □

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/opre.2016.1508.

Acknowledgments

This work was supported by the National Science Foundation [Grants CNS-1017525, CNS-1320175]; Army Research Office [Grants W911NF-11-1-0265, W911NF-14-1-0387]; and the U.S. Department of Transportation supported D-STOP Tier 1 UTC.

References

Audibert J-Y, Bubeck S, Lugosi G (2011) Minimax policies for combinatorial prediction games. *Proc. 24th Ann. Conf. Learn. Theory, COLT'11*, 107–132.

Auer P, Cesa-Bianchi N, Fischer N (2002) Finite-time analysis of the multiarmed bandit problem. *Machine Learn.* 47(2–3):235–256.

Babaioff M, Immorlica N, Kempe D, Kleinberg D (2008) Online auctions and generalized secretary problems. *ACM SIGecom Exchanges* 7(2): Article 7.

Bubeck S, Cesa-Bianchi N (2008) Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations Trends Machine Learn.* 5(1):1–122.

Dimitrov NB, Plaxton CG (2008). Competitive weighted matching in transversal matroids. *Algorithmica* 62(1–2):333–348.

Dudík M, Hsu D, Kale S, Karampatziakis N, Langford J, Reyzin L, Zhang T (2011) Efficient optimal learning for contextual bandits. *Proc. 27th Conf. Uncertainty Artificial Intelligence, UAI'11* (AUAI Press, Corvallis, OR), 169–178.

Fakcharoenphol J, Laekhanukit B, Nanongkai B (2010) Faster algorithms for semi-matching problems. *ACM Trans. Algorithms* 10(3): Article 14.

Gittins JC (1979) Bandit processes and dynamic allocation indices. *J. Roy. Statist. Soc. Ser. B (Methodological)* 41(2):148–177.

Graham RL (1966) Bounds for certain multiprocessing anomalies. *Bell. System Tech. J.* 45(9):1563–1581.

Harvey N, Ladner R, Lovász L, Tamir T (2003) Semi-matchings for bipartite graphs and load balancing. *J. Algorithms* 59(1):53–78.

Jagabathula S, Shah D (2011) Inferring rankings under constrained sensing. *IEEE Trans. Inform. Theory* 57(11):7288–7306.

Keshavan RH, Montanari A, Oh S (2010) Matrix completion from noisy entries. *J. Machine Learn. Res.* 11:2057–2078.

Kleinberg R, Niculescu-Mizil A, Sharma Y (2010) Regret bounds for sleeping experts and bandits. *Machine Learn.* 80(2–3):245–272.

Lai TL, Robbins H (1985) Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.* 6(1):4–22.

Mehta A, Saberi A, Vazirani U, Vazirani V (2005) Adwords and generalized on-line matching. *46th Ann. IEEE Sympos. Foundations Comput. Sci., FOCS '05* (IEEE Computer Society, Washington, DC), 264–273.

Motwani R, Raghavan P (1997) *Randomized Algorithms* (Cambridge University Press, Cambridge, UK).

Rosania P (2015) While you were away…[blog post]. Retrieved April 15, 2016, https://blog.twitter.com/2015/while-you-were-away-0.

Rusmevichientong P, Tsitsiklis JN (2010) Linearly parameterized bandits. *Math. Oper. Res.* 35(2):395–411.

Szabo G, Huberman BA (2010) Predicting the popularity of online content. *Comm. ACM* 53(8):80–88.

Yang J, Leskovec J (2011) Patterns of temporal variation in online media. *4th ACM Internat. Conf. Web Search and Data Mining, WSDM '11* (ACM, New York), 177–186.

Siddhartha Banerjee is an assistant professor in the School of Operations Research and Information Engineering (ORIE) at Cornell, where he works on stochastic modeling and the design of algorithms and incentives for large-scale systems.

Sujay Sanghavi is an associate professor in the Department of Electrical and Computer Engineering at The University of Texas at Austin. His research interests lie at the intersection of two central challenges in modern systems: large-scale networking and high-dimensional data analysis, with a focus on algorithm design and evaluation. He received the NSF CAREER award in January 2010.

Sanjay Shakkottai is a professor in the Department of Electrical and Computer Engineering at The University of Texas at Austin. He received the NSF CAREER award in 2004 and was elected an IEEE Fellow in 2014. His current research interests include network architectures, algorithms and performance analysis for wireless networks, and learning and inference over social networks.