Performance Evaluation 161 (2023) 102351

Contents lists available at ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

Optimizing recommendations under abandonment risks: Models and algorithms

Xuchuang Wang [a], Hong Xie [b,*], Pinghui Wang [c], John C.S. Lui [a]

[a] Department of Computer Science and Engineering, The Chinese University of Hong Kong, New Territory, Shatin, Hong Kong
[b] Chongqing Key Laboratory of Software Theory and Technology, Chongqing University, No. 174 Shazhengjie, Shapingba, Chongqing, 400044, China
[c] MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, No. 28, West Xianning Road, Xi'an, 710049, Shannxi, China

ARTICLE INFO

Article history:
Received 14 December 2022
Received in revised form 11 June 2023
Accepted 13 June 2023
Available online 21 June 2023

Keywords:
Recommendation
Transfer learning
Reinforcement learning

ABSTRACT

User abandonment behaviors are quite common in recommendation applications such as online shopping recommendation and news recommendation. To maximize its total "reward" under the risk of user abandonment, the online platform needs to carefully optimize its recommendations for its users. Because inappropriate recommendations can lead to user abandoning the platform, which results in a short learning duration and reduces the cumulative reward. To address this problem, we formulate a new online decision model and propose an algorithmic framework to *transfer similar users' information* via parametric estimation, and employ this knowledge to *optimize later decisions*. The framework's theoretical guarantees depend on requirements for its transfer learning oracle and online decision oracle. We then design an online learning algorithm consisting of two components that fulfills each corresponding oracle's requirements. We also conduct extensive experiments to demonstrate our algorithm's performance.

1. Introduction

In this paper, we consider how a commercial platform optimizes recommendations to its multiple users who may abandon the platform for inappropriate recommendations [1,2]. Here we use *recommendation* as a general term which includes not only the types of advertisements (ads), news, etc., but also their quantities or frequencies. Let us consider a newsletter platform serving many users, which aims to optimize its ads recommending frequency [3,4]. At each time, new users subscribe to the platform while some existing users may unsubscribe to (abandon) it. The platform obtains rewards via recommending ads to subscribers. To maximize its profit, the platform may send ads frequently. However, too many "uninteresting" ads will annoy users who then abandon the platform [5,6], in which case no more reward can be obtained. From the user reactions towards platform ads, e.g., viewing time [7], the platform can access the user's patience/attitudes [8]. Since users' reactions towards ads are different, how to individually decide each subscriber's advertisement sending frequency so as to maximize the platform's total reward becomes relevant.

The problem contains two crucial settings: (1) the users may leave the platform, and (2) there are many users in the platform at the same time. Both appear in many applications, like advertising strategy in online social networks, and E-commence systems. Take "Twitter ad account and its followers" and "online merchandising and its customers" as two

* Corresponding author.
E-mail address: xiehong2018@foxmail.com (H. Xie).

https://doi.org/10.1016/j.peva.2023.102351
0166-5316/

specific examples. A Twitter ad account would like to disseminate news to its followers either to obtain rewards or attract new subscribers. However, the Twitter account operator should also take care of sending ads or news to its followers, in case of causing their abandonment. As for online merchandising and its customers, when customers shop online and select products, the website merchandiser would like to recommend its products to potential customers so to acquire more profit. However, too many recommendations may annoy costumers and cause them to terminate their shopping.

We address the optimizing recommendations under abandonment risks problem in online learning scenario. Many online learning algorithms (e.g., Q-learning) need a moderate length of sequential data to achieve good performance. However, the user abandonment behavior interrupts the learning and data collecting process, causing a lack-of-data challenge. It leads to either low accuracy in estimation or sub-optimal actions in decision. Example 1 illustrates the challenge.

Example 1. In a newsletter platform, an operator can attach $a \in \{1, 2, 3\}$ ads to news feed. On average, the operator earns \$10, \$20 or \$30 if 1, 2 or 3 advertisements are attached respectively. Let s denote a user's patience value towards the platform. Every time she receives news with a ads, her patience value may decrease by 1 w.p. $p(s-1|s, a)$, or increase by 1 w.p. $p(s+1|s, a)$, or she may abandon the platform w.p. $p(O|s, a)$. These probabilities are unknown to the operator. Suppose $p(O|1, 3) = 0.9$, i.e., the user's abandonment happens w.p. 0.9, if the user has a patience value $s = 1$ and receives $a = 3$ ads. In this case, the expected number of visit to the state–action pair $(s, a) = (1, 3)$ is at most $1.11(\approx 1/0.9)$. This number of visit is far from sufficient for online learning algorithms to make optimal decisions.

We model each user u as an *absorbing Markov Decision Process (MDP)* to capture the abandonment behavior. Users' states $\boldsymbol{s}$ are observable via their feedback. We characterize the *interactions* between a decision maker's action $\boldsymbol{a}$ and multiple users' rewards $\boldsymbol{r}$ with feedback $\boldsymbol{b}$ as a Markov decision model. Then, we propose an online learning problem: *without any prior knowledge of these MDPs (users), how should the decision maker optimize its actions from the interactions?*

We propose a general model-based algorithmic framework to address the above online learning problem. The framework relies on two components: (1) *a δ-accurate personalize transfer learning* (`PersonalizedTL`) *oracle*, and (2) *an (α, β)-stable warm start reinforcement learning* (`WarmStartRL`) *oracle*. The `PersonalizedTL` oracle requires that its output estimated model parameters have a *probably approximately correct (PAC)* accuracy depending on the function δ (see Definition 1 for detail). This oracle addresses the challenge highlighted by Example 1. The `WarmStartRL` oracle requires that the difference between its output estimated value function and the optimal value function is upper bounded by its input parameter's error that is quantified by two terms with coefficients α and β (see Definition 2 for detail).

We design two algorithms that satisfy both oracles' requirements respectively. Plugging both into our general framework, we obtain the LAR (*Learning under Abandonment Risk*) algorithm. The `PersonalizedTL` algorithm, based on users' *"homophily property"*, applies a *"similar user selection"* strategy to transfer useful data with the desired δ-accurate property. These transferred data from users chosen by the similar user selection strategy alleviates the lack-of-data challenge caused by user abandonment behavior. The `WarmStartRL` algorithm utilizes the `PersonalizedTL` algorithm's output to optimize its decisions. The (α, β)-stable property is shown by analyzing the value functions' Lipschitz continuity under the *non-discounted* total reward setting. Finally, we validate our algorithm's performance and compare it with the static policy [9], the Adaptive Real-Time Dynamic Programming (ARTDP) algorithm [10], and the Upper Confidence Bound (UCB) algorithm [11,12].

In summary, our contributions include: (a) formalizing the optimizing recommendations under abandonment risks problem; (b) developing a general algorithmic framework for solving the problem and providing its rigorous theoretical analysis; (c) designing a novel algorithm satisfying the framework's theoretical requirements and empirically validating its performance.

The remainder of this paper is organized as follows. In Section 2, we discuss related literature. In Section 3, we propose our model and define our online learning problem. Then, in Section 4, we design an algorithmic framework to solve the problem. Our LAR algorithm, as an instance of the framework, is represented in Section 5. Full proofs of propositions, lemmas, and theorems are provided in Appendix A. We present numerical experiments to further validate the LAR algorithm in Section 6. At last, we conclude our paper in Section 7.

2. Related works

The user abandonment setting has been studied via dynamic programming [9], multi-armed bandits (MAB) [11–16], and reinforcement learning [17–22]. Schmit and Johari [9] proposed a threshold mechanism to represent a user's abandonment behavior, in which a user abandons once the platform's action value exceeds the user's threshold. They applied dynamic programming to show that in their setting the static policy was optimal. Cao and Wei [11,15] introduced a "marketing fatigue" setting in a sequential choice bandit problem. A similar sequential abandonment setting was considered in cascading bandit (e.g., Kveton et al. [12]). In their works, when observing choices sequentially, a user (or operator) may select one choice in the sequence and stop observing the remaining (i.e. abandonment). Instead of users abandoning the platform, mortal bandits [13] and dying experts [16] considered the abandonment of platforms (e.g., bandits are unavailable to players). These multi-armed bandit models [11,12] assumed that the user's abandonment only depended on the platform's current recommendations and was independent of the user's type and state (e.g., history), while our model takes these realistic features into consideration. So, applying their upper confidence bound (UCB)

algorithms [11,12] to our setting would lead to poor performance. We also empirically validate this in Section 6. Safe reinforcement learning [17–20] aims to avoid undesirable behaviors (e.g., abandonment) and safety is one of its evaluation criteria, which is different from our maximizing total reward objective. Furthermore, different from these previous works, this paper models the abandonment with two novel settings: the user's abandonment behavior is described by absorbing MDPs, and the platform learns policies for multiple users simultaneously.

Our work also sits in utilizing reinforcement learning at recommendation systems, for example, Shani et al. [23], Rojanavasu et al. [24], Zheng et al. [25], Chen et al. [26], Zhao et al. [27], Gharibshah et al. [28], Zhao et al. [29]. Recently, deep Q-network (DQN) was broadly used in recommendations. For instance, Zheng et al. [25] proposed a deep reinforcement learning framework in news recommendation, Chen et al. [26] used generative adversarial model to simulate users to improve the system's long-term reward, and Zhao et al. [27] devised a recommendation-advertising jointly learning framework, etc. We refer interesting readers to the survey [30]. While these works focused on designing deep learning methods and improving their empirical performances, this paper studies the user abandonment behavior and online learning algorithms from a theoretical aspect. Especially, our algorithms have theoretical guarantees, e.g., Theorems 1, 2 and 4. If one model users via reinforcement neural networks, these theoretical results cannot be obtained.

Our LAR algorithm contributes to the field on model-based reinforcement learning [10,31–35] and sample based transfer learning [36–38]. Both the model based estimated interval (MBEI) algorithm [33] and the upper confidence reinforcement learning (UCRL) algorithm [31,34] are similar to the ARTDP algorithm [10]. These three algorithms all require either one infinite long sample path, or a multi-episode learning scheme. Since the abandonment setting restricts them to work under one short-time episode only, they may perform poorly in our problem. In analyzing our algorithm, we study the value functions' Lipschitz continuity that differs from previous studies [32,35] where our statement and proof are based on *non-discounted* setting. In designing the transfer learning algorithm, instead of applying relevance [37] or importance weight [38] as sample distance metric, we use Euclidean distance to select similar users and transfer samples by empirical mean. This results in a simple and general TL method for the multi-user scenario with theoretical guarantees. Besides the works of applying TL to RL, there is another line of work on online clustering of bandits [39–43]. The online clustering of contextual linear bandits was initially studied by Gentile et al. [39], where their algorithm utilized the bandits parameters of users to do clustering. Later, Li et al. [42] extended the clustering method of Gentile et al. [39] to collaborative filtering, and Gentile et al. [44] extended the model of Gentile et al. [39] to contextual-aware case. Recently, Liu et al. [43] further extended the clustering of bandits model to the federated learning scenario for privacy-friendly and decentralized systems. Our algorithm shares some high-level ideas of these references in clustering users and sharing similar user data, however, none of them can be easily applied to solve the state (context) dependent setting, e.g., the MDP model, let along address the user abandonment risk (MDPs with absorbing states) studied in this paper.

The multi-user setting has also been studied in multi-agent reinforcement learning (MARL, e.g., [45]) and multi-agent system [46]. Different from multi-agent setting, we consider users' abandonment and the interactions between one centralized decision maker and multi-user instead of multi-agent, where *agents* have their own objectives like maximizing their own rewards while *users* in our model do not. Therefore, results in MARL with transfer learning (e.g., [47]) are not applicable in our setting.

3. Model and problem formulation

We first present an overview of our model. Then, we formulate absorbing MDPs to model users' abandonment behavior and present a Markov decision model to characterize the interactions between a platform operator and users. Lastly, we formulate the online learning problem.

3.1. The model overview

Consider a decision maker, i.e., the platform operator, who makes decisions in discrete time $t \in \mathbb{N}$. In each time slot t, a finite set $\mathcal{U}_t \subset \mathbb{N}$ of active users is on the platform. The set $\mathcal{U}_t$ changes over time, which captures that users may join or abandon the platform. Let $a_{t,u} \in \mathcal{A}$ denote the decision maker's action to a user $u \in \mathcal{U}_t$ in time slot t, where $\mathcal{A}$ denotes a finite set of actions. Let $b_{t,u} \in \mathcal{B}$ denote user u's feedback to the decision maker's action $a_{t,u}$, where $\mathcal{B}$ denotes a finite set of feedback. Different kinds of feedback are associated with different "rewards", and the decision maker's action $a_{t,u}$ influences the feedback $b_{t,u}$ (we will define this later). Feedback values across different users are independent. To maximize total rewards, the operator aims to select appropriate actions for each user in each time slot.

3.2. Modeling user behavior via MDP

We use an absorbing Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, p_u, r_u)$ to model user u's abandonment behavior, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space we mentioned above, p_u denotes user u's transition probability function, and r_u denotes user u's reward function. The functions p_u and r_u are unknown to the decision maker.

State space $\mathcal{S}$. Define $\mathcal{S} \triangleq \{0, 1, \ldots, S-1, S\} \cup \{O\}$, where O denotes an absorbing state capturing that a user abandons the platform. Let $s_{t,u}$ denote the state of user $u \in \mathcal{U}_t$ in time slot t. When a user first joins the platform at time slot t_u,

its state is initialized as S, i.e., $s_{t_u,u} = S$.[1] The $s_{t,u} \in \{0, 1, \ldots, S-1, S\}$ can be interpreted as the *patience value* of user u. A larger state value means that user u is more patient. User u abandons the platform when $s_{t,u}$ first hits the absorbing state O. Let $T_u = \min_t\{t : s_{t,u} = O\}$ denote the time that user u abandons the platform. The platform *knows* when the user u abandons the platform. For example, when a user unsubscribes from a newsletter, the user needs to change his preference on the platform.

Feedback and state transition. Assume that each user has two kinds of feedback, i.e., $\mathcal{B} = \{-1, 1\}$, where 1 (-1) means that user u is satisfied (unsatisfied) with the decision maker's action. Positive feedback (i.e., $b_{t,u} = 1$) increases user u's patience value by 1, while negative feedback decreases it by 1. To motive the feedback setting, consider a newsletter system operator disseminating news and advertisements to users. User ignoring or blocking the disseminated news can be regarded as *negative feedback*, while opening the news and clicking links within it are considered as *positive feedback*. Users' attitudes towards the platform (i.e., *users' states*) are accumulations of their feedback.

Hence, each user's state $s_{t,u}$ is equal to the accumulation of its feedback values before t and the initial state S. That is

$$s_{t,u} = S + \sum_{k=t_u}^{t-1} b_{k,u} = s_{t-1,u} + b_{t-1,u}, \tag{1}$$

where we recall that t_u is the time slot that user u joins the platform. We note that all users' states are *observable* to the platform as feedback is visible. When $s_{t,u}$ is great than S (alternatively, less than 0), it is truncated as S (as 0). Specially, when the user abandons the platform, we set $s_{t,u} = O$ and denote the current time slot t as the user's termination time T_u.

Transition probability function p_u. We define user $u \in \mathcal{U}_t$'s transition probabilities as $p_u(s'|s, a)$ for s' in $\{s+1, s-1, O\}$, where $p_u(s+1|s, a)$ corresponds to the probability that the patience state increases by one (with positive feedback), $p_u(s-1|s, a)$ is the probability that the patience state decreases by one (with negative feedback), and $p_u(O|s, a)$ denotes the probability that user u abandons the platform from state s and action a, specially, $p_u(O|O, a) = 1$. The transition probability $p_u(\cdot|\cdot, a)$ is indexed by u to indicate that different users can have different transition probabilities under the same action a.

Reward Function r_u. Let $r_u(s_{t,u}, a_{t,u})$ denote the expected reward that the decision maker earns by taking the action $a_{t,u}$ from user $u \in \mathcal{U}_t$ with patience state $s_{t,u}$. It may vary across different users. The reward $r_u(\cdot, \cdot)$ can be expressed as a random variable whose value depends on feedback $b_{t,u}$:

$$r_u(s, a) = \begin{cases} \theta_u(s, a), & \text{if } b_{t,u} = -1 \\ \vartheta_u(s, a), & \text{if } b_{t,u} = 1, \end{cases}$$

where $\theta_u(\cdot, \cdot)$ is the reward if the feedback is negative and $\vartheta_u(\cdot, \cdot)$ is the positive counterpart. Note that if a user leaves the platform (i.e., absorbed to state O), the platform cannot obtain any reward. So, multiplied by a zero reward, the term corresponding to transit to the absorbing state, i.e., $0 \times p_u(O|s, a)$, vanishes in the reward formulation. The reward function is bounded, i.e., exist $c \in \mathbb{R}_+$ such that $|r_u(s, a)| \leqslant c$, for all user u, state s and action a.

3.3. Platform operator's decision model

The objective of the decision maker, through selecting appropriate actions, is to maximize the total cumulative reward from all users. Since users' feedback values are independent, it is equivalent to maximize the reward earned from each user. The total amount of reward earned from user u is

$$R_u \triangleq \sum_{t=t_u}^{T_u} r_u(s_{t,u}, a_{t,u}), \tag{2}$$

where t_u is the time slot that the user u joins the platform, and T_u is random variable representing the time slot that the user u abandons the platform. The reward R_u may be unbounded if T_u goes to infinity, i.e., a user never abandons. To properly define R_u, we make the following assumption and its inferring proposition.

[1] Here we assume that all users share the identical initial state, which unifies our notations. We emphasize that assumption is just for notation simplicity. For example, consider two users with different initial states S_1, S_2 $(S_1 < S_2)$, i.e., two users' state spaces are $\mathcal{S}_1 = \{0, 1, \ldots, S_1\} \cup \{O\}$ and $\mathcal{S}_2 = \{0, 1, \ldots, S_2\} \cup \{O\}$ respectively. Then, we add $S_2 - S_1$ dummy states to the state space $\mathcal{S}_1$ and rewrite it as follows,

$$\begin{aligned} \mathcal{S}_1' &= \underbrace{\{0, 1, \ldots, -1 + (S_2 - S_1)}_{S_2 - S_1 \text{ dummy states}}, \underbrace{0 + (S_2 - S_1), 1 + (S_2 - S_1), \ldots, S_1 + (S_2 - S_1)\}}_{=\mathcal{S}_1} \cup \{O\} \\ &= \{0, 1, \ldots, S_2\} \cup \{O\} \\ &= \mathcal{S}_2, \end{aligned}$$

where, as those dummy states are never reached, we set the transition probabilities from true states to dummy states as zero. More generally, we can set the largest initial state S among all users as a criterion and rewrite all users' state spaces such that—from the aspect of notations—they have the same initial state. In other words, this assumption just *aligns* the state notations of all kinds of users such that the S represents the initial state for all users, and, it does not add any constraint.

Assumption 1 (*Terminating Assumption*)**.** For any policy π, any user's absorbing state is almost surely reachable. That is, under any policy π, there exists a path of non-zero probability connecting all states $\{0, 1, \ldots, S\}$ to the absorbing state O.

Proposition 1. *Under Assumption 1, for any action sequence the decision maker takes, the user will almost surely enter the absorbing state O after a finite number of transitions.*

Assumption 1 states that any user u ends to abandon the platform. Proposition 1 states that each user would eventually abandon the platform in a finite time horizon. Assumption 1, known as the terminating assumption, is a common setting in MDP with absorbing state models for assuring that the total reward in Eq. (2) is well defined (cf., [48,49]).

Proposition 1 also implies that there exists at least one deterministic policy that attains the maximum reward R_u [50, Lemma 7.1.2]. A deterministic stationary policy $\pi : \mathcal{S} \to \mathcal{A}$ prescribes an action for each state. The action under policy π is $a^{\pi}_{t,u} \triangleq \pi(s_{t,u})$. Denote $V^{\pi}_u(s)$ as the expected total reward one can earn from user u via applying policy π from state s, i.e., the value function of a MDP. Then, the expected total reward from initial state S is

$$V^{\pi}_u(S) \triangleq \mathbb{E}[R_u|\pi] = \mathbb{E}\left[\sum_{t=t_u}^{T_u} r_u(s_{t,u}, \pi(s_{t,u}))\right],$$

where the expectation is taken over the whole state transition sequence's randomness. So, the problem is to locate user u's optimal policy, i.e., $\arg\max_{\pi\in\Pi} V^{\pi}_u(S)$, where Π denotes a set of all deterministic stationary policies.

3.4. The online learning problem under abandonment risks

As any user u's state transition probability p_u and the reward function r_u are unknown, the decision maker is in an online learning scenario needing to estimate model parameters and search for the optimal policy simultaneously. Let $\mathcal{H}_t$ denote the interaction history between the decision maker and users up to time slot t:

$$\mathcal{H}_t \triangleq \{(\boldsymbol{a}_1, \boldsymbol{b}_1, \boldsymbol{s}_1, \boldsymbol{r}_1, \mathcal{U}_1) \ldots, (\boldsymbol{a}_t, \boldsymbol{b}_t, \boldsymbol{s}_t, \boldsymbol{r}_t, \mathcal{U}_t)\},$$

where $\boldsymbol{a}_t \triangleq (a_{t,u})_{u\in\mathcal{U}_t}$, $\boldsymbol{b}_t \triangleq (b_{t,u})_{u\in\mathcal{U}_t}$, $\boldsymbol{s}_t \triangleq (s_{t,u})_{u\in\mathcal{U}_t}$, $\boldsymbol{r}_t \triangleq (r_u(s_{t,u}, a_{t,u}))_{u\in\mathcal{U}_t}$ denote the action, feedback, state, and reward vectors in time slot t respectively. The online learning task is as follows.

Problem 1. Design an online learning algorithm Algo that takes the interaction history $\mathcal{H}_{t-1}$ as input, and outputs the action vector $(a_{t,u})_{u\in\mathcal{U}_t} = \text{Algo}(\mathcal{H}_{t-1})$ in each time slot.[2] The objective is to maximize the expected total reward, i.e., for all user u,

$$\max_{\text{Algo}} V^{\text{Algo}}_u(S) \triangleq \max_{\text{Algo}} \mathbb{E}\left[\sum_{t=t_u}^{T_u} r_u(s_{t,u}, a_{t,u})\right],$$

where the initial state is $s_{t_u,u} = S$, and the actions in each time slot are selected by Algo, i.e., $a_{t,u} = (\text{Algo}(\mathcal{H}_{t-1}))_u$.

4. General algorithmic framework

We first present the high-level idea of our algorithmic framework, which consists of a `PersonalizedTL` oracle, and a `WarmStartRL` oracle. Then we establish sufficient conditions for both oracles to guarantee the framework's theoretical performance in solving Problem 1.

Note that the `PersonalizedTL` oracle and the `WarmStartRL` oracle – as abstract building blocks in the framework – respectively stand for any unspecific algorithms that fulfill their requirements (define later in Definitions 1 and 2). In Section 5, we design two specific algorithms as two possible realizations of `PersonalizedTL` and `WarmStartRL` oracles. These two specific realizations are called the `PersonalizedTL` algorithm and `WarmStartRL` algorithm. Together with the framework proposed in this section, they form the LAR algorithm that can solve Problem 1.

4.1. Algorithmic framework design

Algorithm 1 outlines the framework. It consists of two components. The first one is a `PersonalizedTL` oracle in line 4, where only the historical observations of users similar to u (see Definition 3) are *transferred*. This oracle resolves the user data insufficiency challenge. It outputs the estimated model parameters for user u, i.e., $(\hat{p}_{t,u}, \hat{r}_{t,u})$. The second component is a `WarmStartRL` oracle. With newly estimated $(\hat{p}_{t,u}, \hat{r}_{t,u})$ as warm start parameters, its `Update` part (line 5) refreshes `WarmStartRL` record $\boldsymbol{X}_{t,u}$ that depends on the specific RL algorithm. Note that $\boldsymbol{X}_{t,u}$ is an explicit record of `WarmStartRL` oracle that aggregates historical information of previous rounds, and will be utilized and further updated in future rounds.

[2] The learning algorithm Algo is different from a policy π. Because the policy π maps the state to action while the algorithm maps the whole interaction history (up to the current time slot) to action.

For example, the $\boldsymbol{X}_{t,u}$ can be the value function and Q-function according to specific `WarmStartRL` oracle. The `Play` part (line 6) takes the estimated model parameters $(\hat{p}_{t,u}, \hat{r}_{t,u})$, the current state $s_{t,u}$, and the latest `WarmStartRL` record $\boldsymbol{X}_{t,u}$ as input, and optimize the next action $a_{t,u}$ as output.

Algorithm 1 General Algorithmic Framework (illustrated in Fig. 1)

1: **Initialization:** Interaction history $\mathcal{H}_0 = \emptyset$ and users' `WarmStartRL` record $\boldsymbol{X}_{t_u-1,u} = \emptyset$.
2: **for** $t = 1, \ldots, \infty$ **do**
3: **for** $u \in \mathcal{U}_t$ **do**
4: $(\hat{p}_{t,u}, \hat{r}_{t,u}) \leftarrow$ `PersonalizedTL`$(\mathcal{H}_{t-1}, u)$.
5: $\boldsymbol{X}_{t,u} \leftarrow$ `WarmStartRL.Update`$(\hat{p}_{t,u}, \hat{r}_{t,u}, \boldsymbol{X}_{t-1,u})$.
6: $a_{t,u} \leftarrow$ `WarmStartRL.Play`$(s_{t,u}, \hat{p}_{t,u}, \hat{r}_{t,u}, \boldsymbol{X}_{t,u})$.
7: Perform $a_{t,u}$ to user u, receive feedback $b_{t,u}$, reward $r_{t,u}$, and update state $s_{t+1,u}$ via Eq. (1).
8: **end for**
9: $\mathcal{H}_t \leftarrow \mathcal{H}_{t-1} \cup \left(\cup_{u\in\mathcal{U}_t}\{(a_{t,u}, b_{t,u}, s_{t,u}, r_{t,u}, u)\}\right)$.
10: **end for**

The remaining issues are: (1) *How to design appropriate* `PersonalizedTL`, `WarmStartRL` *oracles?* (2) *What is the theoretical performance guarantee for the above algorithm?* To answer both questions, we next present a general theoretical framework that guides to design oracles and analyze the algorithmic framework in Algorithm 1.

4.2. General theoretical analysis framework

To simplify notations, we define two norm distances between user u and w's reward functions r_u and r_w, and between their transition probability functions p_u and p_w as follows,

$$\|r_u - r_w\| \triangleq \|\boldsymbol{R}_u - \boldsymbol{R}_w\|_\infty, \quad \|p_u - p_w\| \triangleq \|\boldsymbol{P}_u - \boldsymbol{P}_w\|_\infty,$$

where $\|\cdot\|_\infty$ is the infinity matrix norm and

$$\boldsymbol{R}_u \triangleq [r_u(s,a)]_{(s,a)\in\mathcal{S}\times\mathcal{A}}, \boldsymbol{P}_u \triangleq [p_u(s'|s,a)]_{(s',s,a)\in\mathcal{S}^2\times\mathcal{A}}$$

are the reward function and transition probability's matrix forms. We define a metric to quantify the `PersonalizedTL` oracle's accuracy.

Definition 1. A personalized TL oracle *`PersonalizedTL`*$(\mathcal{H}_{t-1}, u)$ for user u with history $\mathcal{H}_{t-1}$ is δ-accurate, if its output $(\hat{p}_{t,u}, \hat{r}_{t,u})$ satisfies the following probability inequality,

$$\mathbb{P}\left[\|\hat{r}_{t,u} - r_u\| \leqslant \epsilon, \|\hat{p}_{t,u} - p_u\| \leqslant \eta\right] \geqslant 1 - \delta(\mathcal{H}_{t-1}, \epsilon, \eta),$$

where $\epsilon, \eta \in \mathbb{R}_+$ are tunable parameters and function $\delta(\mathcal{H}_{t-1}, \epsilon, \eta) \in [0, 1]$ depends on the specific *`PersonalizedTL`* algorithm.

The expression in Definition 1 is analogous to the concentration inequality commonly used in the analysis of RL algorithms (cf. [51]), and, thus, can be obtained as an intermediate result from the analysis of these RL algorithms. We note that the accuracy in Definition 1 only depends on the function $\delta(\cdot,\cdot,\cdot)$, and the parameters ϵ and η used in this definition are only formal variables of function $\delta(\cdot,\epsilon,\eta)$ and they are independent of model and algorithm. The $\delta(\mathcal{H}_{t-1},\epsilon,\eta)$ is a key element in characterizing a `PersonalizedTL` oracle. In general, $\delta(\mathcal{H}_{t-1},\epsilon,\eta)$ is decreasing in both ϵ and η, capturing that one can have a higher confidence through relaxing the error ϵ and η. The element also decreases with respect to time slot t, implying that utilizing more information increases the estimation's confidence. The form of δ depends on the specific algorithm (see Appendix A.4 as an example). Let us define a metric to quantify the `WarmStartRL` oracle's stability.

Definition 2. Fix any user $u \in \mathcal{U}_t$. Let $\hat{\pi}_{t,u}$ denote a policy associated with the *`WarmStartRL`* oracle, i.e.,

$$\hat{\pi}_{t,u}(s) = \texttt{WarmStartRL.Play}(s, \hat{p}_{t,u}, \hat{r}_{t,u}, \boldsymbol{X}_{t,u}),$$

for any s in $\mathcal{S}$. Then, a *`WarmStartRL`* oracle is (α, β)-stable, if the expected total reward $V_u^{\hat{\pi}_{t,u}}(S)$ corresponding to policy $\hat{\pi}_{t,u}$ satisfies, for any state s

$$\left|V_u^{\pi_u^*}(s) - V_u^{\hat{\pi}_{t,u}}(s)\right| \leqslant \alpha\epsilon + \beta\eta,$$

where $V_u^{\pi_u^*}(\cdot)$ is the value function of user u corresponding to its optimal policy $\pi_u^* \triangleq \arg\max_{\pi\in\Pi_{\text{DM}}} V_u^{\pi}(S)$,[3] parameters ϵ and η need to fulfill the constraints $\epsilon \geqslant \|\hat{r}_{t,u} - r_u\|$ and $\eta \geqslant \|\hat{p}_{t,u} - p_u\|$, and $\alpha, \beta \in \mathbb{R}_+$ are parameters depending on the specific *`WarmStartRL`* algorithm.

[3] The Π_{DM} represents the set of all deterministic Markovian policies.

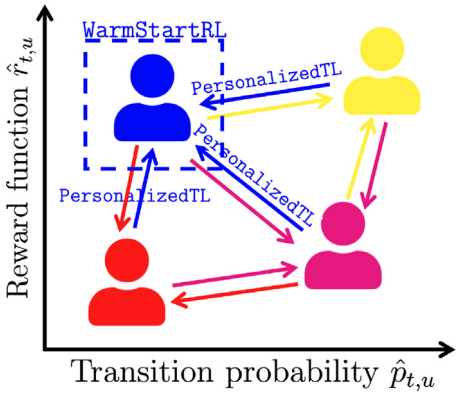


Fig. 1. Algorithm illustration of Section 5: this figure contains 4 users located according to their estimated transition probabilities $\hat{p}_{t,u}$ and reward functions $\hat{r}_{t,u}$. A directed arrow means the user at end point is similar to users at source point, and `PersonalizedTL` transfers the data of the user at source point to the user at the end point. Then, each user (e.g., the blue one highlighted by the dashed square) utilizes these transferred data via `WarmStartRL`.

Definition 2 requires that the `WarmStartRL` oracle's output error is bounded by a linear combination of its input's error. Smaller α and β imply a stabler algorithm. We note that the simulation lemma [52] and Definition 2 are different: the simulation lemma says that the same policy has similar performance on two similar MDPs, while Definition 2 means that for the same MDP, the estimated optimal policy derived from an estimate of this MDP (via the `WarmStartRL` oracle) has a similar performance with the true optimal policy. Then, we provide the framework's theoretical result in Theorem 1.

Theorem 1. *Given a δ-accurate* `PersonalizedTL` *oracle and an (α, β)-stable* `WarmStartRL` *oracle, the Algorithm 1's output expected total reward $V_u^{Algo}(S)$ satisfies a probability inequality as follows,*

$$\mathbb{P}\left[V_u^{\pi^*}(S) - V_u^{Algo}(S) \leqslant \sum_{t=t_u}^{\infty}(1-\gamma)^t\left[(\alpha\epsilon_t + \beta\eta_t) + (\alpha\epsilon_{t+1} + \beta\eta_{t+1})\right]\right] \geqslant 1 - \sum_{t=t_u}^{\infty}\mathbb{E}[\delta(\mathcal{H}_{t-1}, \epsilon_t, \eta_t)],$$

where the γ is the smallest abandonment probability of all state–action pairs, i.e., $\gamma = \min_{(s,a)\in\mathcal{S}\times\mathcal{A}} p_u(O|s,a)$, and parameters $\epsilon_t, \eta_t \in \mathbb{R}^+$ fulfill constraints $\epsilon_t \geqslant \|\hat{r}_{t,u} - r_u\|$ and $\eta_t \geqslant \|\hat{p}_{t,u} - p_u\|$ for all time slot t.

Theorem 1 states sufficient conditions to quantify Algorithm 1's accuracy. Essentially, it requires one to design (1) a δ-accurate `PersonalizedTL` oracle, and (2) an (α, β)-stable `WarmStartRL` oracle. We note that the choice of paramteres ϵ_t and η_t of Theorem 1 depends on the `PersonalizedTL` oracle. For these `PersonalizedTL` oracles that can give constant upper bounds for $\|\hat{r}_{t,u} - r_u\|$ and $\|\hat{p}_{t,u} - p_u\|$ as the values of ϵ_t and η_t, Theorem 1 provides the standard probably approximately correct (PAC) bounds. For example, with the `PersonalizedTL` oracle in Section 5.1, ϵ_t and η_t are both constants (see Theorem 2 for detail). We also note that Theorem 1 for a single user u can be easily extended to all users, e.g., via a union bound and summing up all users' cumulative rewards.

5. The LAR algorithm

In this section, we design a `PersonalizedTL` algorithm and a `WarmStartRL` algorithm as instances to fit our framework's oracles. They constitute our *Learning under Abandonment Risks* (LAR) algorithm.

We note that the `PersonalizedTL` and `WarmStartRL` algorithms proposed in this section are only two possible realizations of the `PersonalizedTL` and `WarmStartRL` oracles. One can also design other algorithms that fulfill the oracles' requirements in Definitions 1 and 2, thanks to the generality of our framework in Algorithm 1 (see Fig. 1).

5.1. The `PersonalizedTL` algorithm

We propose a `PersonalizedTL` algorithm. It utilizes other users' data under a similar user selection strategy. We analyze the algorithm and show it satisfies the requirements in Definition 1. The algorithm contains two parts: (1) *similar user selection*; and (2) *interaction history transfer*. The latter utilizes the selected users' history to estimate a certain user's parameters.

Similar user selection. Since each user u is characterized by p_u and r_u, we define a metric to quantify the similarity between two users in terms of their transition probability functions and reward functions as follows.

Definition 3. Users u and w are (ξ, ζ)-similar, if $\|r_u - r_w\| \leqslant \xi$, $\|p_u - p_w\| \leqslant \zeta$, where $\xi, \zeta \in \mathbb{R}_+$ are tunable parameters.

Smaller ξ or ζ implies more similar users. Challenges in selecting similar users for u from the interaction history $\mathcal{H}_{t-1}$ are threefold: (1) *For each user u, we do not know the exact value of (p_u, r_u), but some observations only;* (2) *The number of such observations is usually small due to the abandonment behavior;* (3) *Furthermore, (p_u, r_u)'s parameter space $[0,1]^{|\mathcal{S}|^2|\mathcal{A}|} \times [0,c]^{|\mathcal{S}||\mathcal{A}|}$ is of high dimension.* To address above challenges, we first provide Definition 4 to quantify an important property between users in the same population.

Definition 4. The variation of the abandonment behavior of the user population $\cup_t \mathcal{U}_t$ is (ρ, ϱ)-bounded, if for any two users u and w, we have (1) $\|r_u - r_w\| \leqslant \rho \|\boldsymbol{q}_u - \boldsymbol{q}_w\|_\infty$ and (2) $\|p_u - p_w\| \leqslant \varrho \|\boldsymbol{q}_u - \boldsymbol{q}_w\|_\infty$, where ρ and ϱ are two positive constants, and the vector $\boldsymbol{q}_u \triangleq [p_u(s|S,0)]_{s\in\mathcal{S}}$ denotes the probability mass function of transition from the initial state S under action 0.[4]

We first motivate why estimating $\boldsymbol{q}_u \triangleq [p_u(s|S,0)]_{s\in\mathcal{S}}$ is important as Definition 4 shows. In real-world scenarios, many e-shops employ welcome coupons as a conservative action (denoted as action 0 in footnote 4) to incentivize new users to stay on their platforms (with the highest patience S). These coupons not only encourage users to remain engaged with the e-shop but also provide valuable information about the preferences of new customers. Consequently, the user's repeated visits to the initial state S allow these e-shops to gain insights into the customer's preferences (e.g., estimate $p_u(s|S,0)$). In this context, the observation from Eq. (3) that frequent visits to state S are necessary for obtaining a good estimate of $p_u(s|S,0)$ aligns perfectly with the real-world strategy employed by e-shops.

Smaller ρ or ϱ implies that the user population $\cup_t \mathcal{U}_t$ has a smaller variation in the abandonment behavior. Definition 4 captures the *homophily* in the abandonment behavior among the user population, i.e., users with similar behavior in early subscription phase tend to have similar abandonment behavior (cf. McPherson et al. [53]). Thus, we bound the variation in terms of the transition probability distribution of the initial state. Definition 4 is not restrictive, as it allows ρ and ϱ to go to infinity, capturing large variation.

Suppose that the abandonment behavior of the user population $\cup_t \mathcal{U}_t$ is (ρ, ϱ)-bounded. We can select similar users from their transition behaviors in the initial state S under the most conservative action 0 (i.e. the state–action pair with the smallest abandonment probability), for it is safe to perform more explorations in estimating $\boldsymbol{q}_u$ for higher accuracy. Formally, we apply the sample mean to estimate its sth entry $\hat{\boldsymbol{q}}_{t,u}(s)$ as follows,

$$\hat{\boldsymbol{q}}_{t,u}(s) = \frac{\sum_{l=t_u}^{\tilde{T}_{t,u}} \mathbb{1}\{(s_{l+1,u}, s_{l,u}, a_{l,u}) = (s,S,0)\}}{\sum_{l=t_u}^{\tilde{T}_{t,u}} \sum_{x\in\mathcal{S}} \mathbb{1}\{(s_{l+1,u}, s_{l,u}, a_{l,u}) = (x,S,0)\}}, \tag{3}$$

where $\tilde{T}_{t,u} \triangleq \min\{T_u, t-1\}$, and $\mathbb{1}\{A\}$ is the indicator function that is equal to 1 when event A happens, or 0 otherwise. We use $\hat{\boldsymbol{q}}_{t,u}$ as the key to instruct the selection. Denote a set of users similar to u within distance d as:

$$\mathcal{C}_{t,u}(d) \triangleq \left\{ w \in \cup_{i\leqslant t-1}\mathcal{U}_i \backslash \{u\} : \|\hat{\boldsymbol{q}}_{t,u} - \hat{\boldsymbol{q}}_{t,w}\|_\infty \leqslant d \right\}. \tag{4}$$

Smaller d leads to higher accuracy in user selection but a fewer number of similar users (a smaller amount of data). We will quantify this trade-off in Theorem 2. In practice, instead of setting d, we choose the scale of $\left|\mathcal{C}_{t,u}(d)\right|$ as the selection criterion which can control the accuracy more directly.

History data transfer. We transfer $\mathcal{C}_{t,u}(d)$'s data to estimate $(\hat{p}_{t,u}, \hat{r}_{t,u})$ as follows,

$$\hat{p}_{t,u}(s'|s,a) = \frac{\sum_{w\in\mathcal{C}_{t,u}(d)} \sum_{l=t_w}^{\tilde{T}_{t,w}} \mathbb{1}\{(s_{l+1,w}, s_{l,w}, a_{l,w}) = (s',s,a)\}}{\sum_{w\in\mathcal{C}_{t,u}(d)} \sum_{l=t_w}^{\tilde{T}_{t,w}} \sum_{x\in\mathcal{S}} \mathbb{1}\{(s_{l+1,w}, s_{l,w}, a_{l,w}) = (x,s,a)\}}, \tag{5}$$

$$\hat{r}_{t,u}(s,a) = \frac{\sum_{w\in\mathcal{C}_{t,u}(d)} \sum_{l=t_w}^{\tilde{T}_{t,w}} \sum_{x\in\mathcal{S}} r_{t,w} \mathbb{1}\{(s_{l+1,w}, s_{l,w}, a_{l,w}) = (x,s,a)\}}{\sum_{w\in\mathcal{C}_{t,u}(d)} \sum_{l=t_w}^{\tilde{T}_{t,w}} \sum_{x\in\mathcal{S}} \mathbb{1}\{(s_{l+1,w}, s_{l,w}, a_{l,w}) = (x,s,a)\}}. \tag{6}$$

Combining both steps, we have Algorithm 2 as `PersonalizedTL` for our LAR algorithm.

Algorithm 2 `PersonalizedTL`$(\mathcal{H}_{t-1}, u)$

Input: history data $\mathcal{H}_{t-1}$ and user u.
Initialize: distance d.
Calculate all users' $\hat{\boldsymbol{q}}_u$ by Eq. (3).
Select similar users into set $\mathcal{C}_{t,u}(d)$ by Eq. (4).
Estimate $(\hat{p}_{t,u}, \hat{r}_{t,u})$ from $\mathcal{C}_{t,u}(d)$'s history by Eq. (5), (6).
Output: user u's estimated parameters $(\hat{p}_{t,u}, \hat{r}_{t,u})$.

[4] We assume that the action 0 represents a conservative recommendation, e.g., a newsletter with high-value coupons. Besides, due to the state transition in Eq. (1), only two entries of vector $\boldsymbol{q}_u$ (i.e., $p_u(S|S,0)$ and $p_u(S-1|S,0)$) are non-zero. This fact further simplifies the estimate of vector $\boldsymbol{q}_u$ because one only needs to estimate one of the last two entries instead of all entries of the vector.

Theoretical guarantee. We analyze Algorithm 2's theoretical properties. Lemma 1 quantifies the accuracy of *similar user selection*.

Lemma 1. *Suppose that the abandonment behavior of the user population $\cup_t \mathcal{U}_t$ is (ρ, ϱ)-bounded. Then, the probability that user u and w in $\mathcal{C}_{t,u}(d)$ are similar satisfies:*

$$\mathbb{P}[u \text{ and } w \text{ are } \big(\rho(2\xi + d), \varrho(2\xi + d)\big)\text{-similar }] \geqslant 1 - 2\,|\mathcal{S}| \exp\left(-2n_u(S,0)\xi^2\right) - 2\,|\mathcal{S}| \exp\left(-2n_w(S,0)\xi^2\right),$$

where $\xi \in \mathbb{R}_+$ is a tunable parameter, and $n_u(S,0)$ is the visiting times of state–action pair $(S,0)$ of user u.

Lemma 1 suggests that the visiting times of the state–action pair $(S,0)$ in user's history is the bottleneck for improving the selection's accuracy. That is coherent with our Definition 4's $\boldsymbol{q}_u$ vector. The selection of d and ξ has a trade-off: if the number of visit $n_u(s,a)$ and $n_w(s,a)$ are large, we can use smaller d and ξ for a smaller error bound, or *vice versa*, larger d and ξ for a higher confidence. The next theorem is the main analytical result of the `PersonalizedTL` algorithm in Algorithm 2.

Theorem 2. *Algorithm 2 conforms to a PAC-bound: the probability that the estimated parameters $(\hat{p}_{t,u}, \hat{r}_{t,u})$ are similar to the true ones satisfies*

$$\begin{aligned}
&\mathbb{P}[\|\hat{r}_{t,u} - r_u\| \leqslant \epsilon + 2\rho(2\xi + d) \text{ and } \|\hat{p}_{t,u} - p_u\| \leqslant \eta + 2\varrho(2\xi + d)] \\
\geqslant &1 - \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} 2\exp\left(-2\epsilon^2 \sum_{w\in\mathcal{C}_{t,u}(d)} \frac{n_w(s,a)}{c^2}\right) - \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} 2\exp\left(-2\eta^2 \sum_{w\in\mathcal{C}_{t,u}(d)} n_w(s,a)\right) \\
&- 2\,|\mathcal{S}|\,|\mathcal{C}_{t,u}(d)| \exp\left(-2n_u(S,0)\xi^2\right) - 2\,|\mathcal{S}| \sum_{w\in\mathcal{C}_{t,u}(d)} \exp\left(-2n_w(S,0)\xi^2\right),
\end{aligned} \tag{7}$$

where ϵ, η, ξ are positive tunable parameters, $|\mathcal{C}_{t,u}(d)|$ is the cardinality of similar user set, and $n_u(s,a)$ denotes the visiting times of state–action pair (s,a) of user u.

The complex exponential bound of Theorem 2 is due to *history data transfer*, i.e., incorporating not a single but a group of users' history into estimation. A larger d would increase the number of similar users in $\mathcal{C}_{t,u}(d)$ and enlarges the variation between selected users. Similar to ξ's selection trade-off in Lemma 1, if many users are in $\mathcal{C}_{t,u}(d)$, we can use smaller ϵ and η for higher accuracy, or otherwise larger ϵ and η for a fair confidence probability.

Theorem 2 shows that Algorithm 2 achieves the `PersonalizedTL` oracle in Definition 1. We can take the four exponential parts in Eq. (7) as an expression of $\delta(\mathcal{H}_{t-1}, \epsilon + 2\rho(2\xi + d), \eta + 2\varrho(2\xi + d))$. Notice that ξ, d are set in algorithm and ρ, ϱ depend on scenario, so only ϵ and η are variables. We derive the $\delta(\mathcal{H}_{t-1}, \epsilon, \eta)$ function with other proofs in Appendix A. That δ's formula decreases in the visiting times of any state–action pair encourages us to add explorations in the next `WarmStartRL` algorithm, e.g., ε-greedy policy.

5.2. The `WarmStartRL` algorithm

Next, we design a `WarmStartRL` algorithm satisfying the stable property in Definition 2. Denote user u's estimated value function at time t as $\hat{V}_{t,u} : \mathcal{S} \to \mathbb{R}$ and Let $V_u^{\pi}(s)$ be the expected total reward from state s under policy π.

We take the value iteration algorithm of MDP with N fixed rounds as our `WarmStartRL` algorithm, where the `WarmStartRL` record $\boldsymbol{X}_{t,u}$ is the estimated value function $\hat{V}_{t,u}$. Its `Update` part outputs $\hat{V}_{t,u}$ via iterating $\hat{V}_{t-1,u}$ for N times with user u's estimated parameters $(\hat{p}_{t,u}, \hat{r}_{t,u})$ in time slot t. The `Play` part greedily selects the action with the highest expected reward according to current $\hat{V}_{t,u}$ and parameters $(\hat{p}_{t,u}, \hat{r}_{t,u})$.

Theoretical guarantees. We quantify the convergence of the absorbing MDP's value function in the following lemma.

Lemma 2. *For each given N, the value iteration's output $\hat{V}_{t,u}$ satisfies the following inequality, for any s in $\mathcal{S}$,*

$$\left|V_{\tilde{u}}^{\pi_{\tilde{u}}^*}(s) - \hat{V}_{t,u}(s)\right| \leqslant \kappa^N \tau_u(S) \max_{s'\in\mathcal{S}} V_{\tilde{u}}^{\pi_{\tilde{u}}^*}(s'), \tag{8}$$

where $V_{\tilde{u}}^{\pi_{\tilde{u}}^*}$ is the optimal value function for a user $\tilde{u}$ (a MDP)[5] with transition probability and reward functions $(\hat{p}_{t,u}, \hat{r}_{t,u})$ and the same state and action spaces as the user u, and $\tau_u(s)$ is the maximum expected number of steps starting from state s to absorbing state O (of all possible policies $\pi \in \Pi$), or formally, defining random variable $N_u^{\pi}(s)$ as the number steps that the user u—starting from state s with policy π—reaches the absorbing state O,

$$\tau_u(s) \triangleq \max_{\pi\in\Pi} \mathbb{E}_{\pi}[N_u^{\pi}(s)],$$

and $\kappa \triangleq \max_{s\in\mathcal{S}} (\tau_u(s) - 1)/\tau_u(s)$.

[5] The tilde of $\tilde{u}$ emphasizes that the user is only conceptual and not match the subscriptions of $\hat{r}_{t,u}, \hat{p}_{t,u}$.

Table 1
Time and space complexities of two oracles in Section 5: the $|\mathcal{S}|$, $|\mathcal{A}|$, and $|\mathcal{U}|$ denote the number of states, actions, and users respectively, and the N denotes the number of iterations of our `WarmStartRL` algorithm.

	Time Complexity	Space Complexity
`PersonalizedTL` (Section 5.1)	$O(\lvert\mathcal{U}\rvert \lvert\mathcal{S}\rvert^2 \lvert\mathcal{A}\rvert)$ per time slot	$O(\lvert\mathcal{U}\rvert \lvert\mathcal{S}\rvert^2 \lvert\mathcal{A}\rvert)$
`WarmStartRL` (Section 5.2)	$O(N \lvert\mathcal{U}\rvert \lvert\mathcal{S}\rvert^2 \lvert\mathcal{A}\rvert)$ per time slot	$O(\lvert\mathcal{U}\rvert \lvert\mathcal{S}\rvert^2 \lvert\mathcal{A}\rvert)$

Lemma 2 shows that the estimated $\hat{V}_{t,u}$ can approximate the true $V^*_{\tilde{u}}$ in *"any degree"* of accuracy. The error of estimation (RHS of Eq. (8)) vanishes at a geometric rate of N for Proposition 1 ascertains κ is strictly less then 1. Next, we quantify the Lipschitz continuity of value function with respect to the MDP model parameters.

Theorem 3. *Let V^π_u and V^π_w denote the value functions of user u and w under the same stationary policy π. Then, the following inequality holds, for any $s \in \mathcal{S}$,*

$$|V^\pi_u(s) - V^\pi_w(s)| \leqslant k_1 \|r_u - r_w\| + k_2 \|\boldsymbol{P}^\pi_u - \boldsymbol{P}^\pi_w\|_\infty \leqslant k_3 \|r_u - r_w\| + k_4 \|p_u - p_w\|,$$

where $\boldsymbol{P}^\pi_u \triangleq [p_u(j|i, \pi(i))]_{(i,j)\in\mathcal{S}^2}$ is the transition matrix under policy π, $k_1 = \|(\boldsymbol{I} - \boldsymbol{P}^\pi_w)^{-1}\|_\infty$, $k_2 = k_1 \max_{s'\in\mathcal{S}} V^\pi_w(s')$, $k_3 = \max_{\pi'\in\Pi} \|(\boldsymbol{I} - \boldsymbol{P}^{\pi'}_w)^{-1}\|_\infty$, and $k_4 = k_3 \max_{\pi'\in\Pi} \max_{s'\in\mathcal{S}} V^{\pi'}_w(s')$.

Theorem 3 states that under the same policy π, the difference between total expected rewards from user u and w are bounded by a linear combination of the distance between their reward functions and the distance between their transition probability functions. Theorem 3's statement and proof are based on the *non-discounted* reward setting, which is new and different from those Lipschitz continuities derived from *discounted* setting [32,35]. Lemma 2 and Theorem 3 together deduce the stable property for our `WarmStartRL` algorithm.

Theorem 4. *For any state s in $\mathcal{S}$, our `WarmStartRL` algorithm employs the greedy policy $\hat{\pi}_{t,u}(s)$ as follows*

$$\arg\max_{a\in\mathcal{A}} \left(\hat{r}_{t,u}(s,a) + \sum_{s'\in\mathcal{S}} \hat{p}_{t,u}(s,a,s') \hat{V}_{t,u}(s) \right).$$

Let $V^{\hat{\pi}_{t,u}}_u$ denote the value function of user u under the policy $\hat{\pi}_{t,u}$. Then, the following inequality holds,

$$|V^{\pi^*_u}_u(s) - V^{\hat{\pi}_{t,u}}_u(s)| \leqslant 2k_3 \|r_u - \hat{r}_{t,u}\| + 2k_4 \|p_u - \hat{p}_{t,u}\| + k_5 \kappa^N,$$

where $k_5 = \tau_u(S) \max_{s'\in\mathcal{S}} V^{\pi^*_{\tilde{u}}}_{\tilde{u}}(s')$, κ and N follow Lemma 2, and k_3 and k_4 follow Theorem 3.

Theorem 4 signifies that the `WarmStartRL` algorithm is asymptotically $(2k_3, 2k_4)$-stable, for $k_5\kappa^N$ vanishes as N goes to infinity. Similar results also holds after adding an ε-greedy exploration, as the convergence of value iteration in Lemma 2 holds when its transition matrix only has ε scale perturbation. The section shows that our LAR algorithm fits our general framework well and achieves its required theoretical guarantees.

Impact of the number of states $|\mathcal{S}|$. In general, the larger the number of states $|\mathcal{S}|$, the more number of samples that is required for an algorithm to achieve a good performance. For our `PersonalizedTL` oracle in Section 5.1, the probability that the estimated MDP parameters is close to the ground truth MDP parameters decreases with respect to the number of states $|\mathcal{S}|$ (see Theorem 2's RHS). For our `WarmStartRL` oracle in Section 5.2, the upper bound of the differences between the estimated and the ground truth value functions increases with respect to $|\mathcal{S}|$ (see Theorem 4's RHS) (see Table 1).

6. Simulations

In this section, we evaluate the empirical performance of our LAR algorithm. We compare it with five baselines: (1) the static policy that selects the optimal fixed action, (2) the ARTDP algorithm, (3) the upper confidence bound (UCB) algorithm, (4) Round-Robin (RR) policy, and (5) Random policy. The static policy is proposed by Schmit and Johari [9] via modeling abandonment setting as a threshold mechanism. Comparing with this method shows that the LAR algorithm suits our user abandonment formulation. As the LAR algorithm is model-based, we also compare it with the classic model-based RL algorithm, ARTDP [10], which shows the necessity of transferring other users' information in the abandonment setting. The UCB algorithm is adapted from the multi-armed bandit literatures [11,12] that studies the abandonment setting.[6] Comparing LAR to the UCB algorithm illustrates the advantage of modeling users as MDPs over multi-armed

[6] Designing UCB algorithm for our MDP is challenging. Each policy of our MDP is associated with an absorbing Markov chain. This absorbing nature connects to challenges in the open problem of risk of ruin in multiarmed bandits [54]. Thus, we consider a naïve adaption of UCB to our setting. Specifically, in our implementation, the UCB algorithm maintains $|\mathcal{A}|$ UCB indexes each of which corresponds to one action $a \in \mathcal{A}$, (completely ignores the information of state,) and chooses the action with the highest UCB index in each time slot.

bandits. The Random-Robin chooses actions among action space in a circular manner. The random policy uniformly randomly chooses actions among action space.

Algorithms setting: Our performance metric is the average V_u^{Algo}/V_u^* ratio (with standard variance error bar) over 2000 users and 30 rounds, where V_u^* is the total reward under the optimal policy and V_u^{Algo} corresponds to simulated policies. For our `PersonalizedTL` algorithm, knowing all empirical estimates of user parameters, the algorithm chooses a d such that $\left|\mathcal{C}_{t,u}(d)\right| = 30$, i.e., it selects the closest 30 users as similar ones. For our `WarmStartRL` algorithm, we set the iteration round as $N = 100$ in its `Update` part, and apply the ε-greedy algorithm in its `Play` part, where $\varepsilon = 0.01$. The ARTDP algorithm also employs the same ε-greedy algorithm. The static policy selects the action $a = 0$ which has the highest output total reward among all static policies.

6.1. Synthetic user MDP setting

All users have the same state and action space in each simulation: $\mathcal{S} = \{0, 1, 2, 3, \ldots, k_s-1\} \cup \{O\}$, $\mathcal{A} = \{0, 1, 2, \ldots, k_a\}$, where k_s and k_a represent the number of states and actions. Also, users have the same reward function: the positive feedback reward is $\vartheta_u(s, a) = 4(a + 1)$ and the negative one is $\theta_u(s, a) = a + 1$. Let k denote the number of user types. Users with different types have different transition probability functions. We define a user feature constant e_u. It depends on the distance (dist) between users of different types and the user's type index: $e_u = 0.1 + \text{dist} \times \text{index}$. In each time slot t, we generate a user whose type index is t mod k. Let $q_u(s, a) = 0.1e_u(a + s) + 0.2$. Then, the transition probability function is as follows,

$$p_u(s'|s, a) = \begin{cases} 1, & s = s' = O \\ \frac{0.1s}{1+0.1s}, & s' = O, s \neq O \\ \frac{q_u(s,a)}{1+0.1s}, & s' = \min\{S, s + 1\}, s \neq O \\ \frac{1-q_u(s,a)}{1+0.1s}, & s' = \max\{0, s - 1\}, s \neq O. \end{cases} \tag{9}$$

Our default settings for control variables are 6 states, 4 actions, 5 user types with distance 0.125.

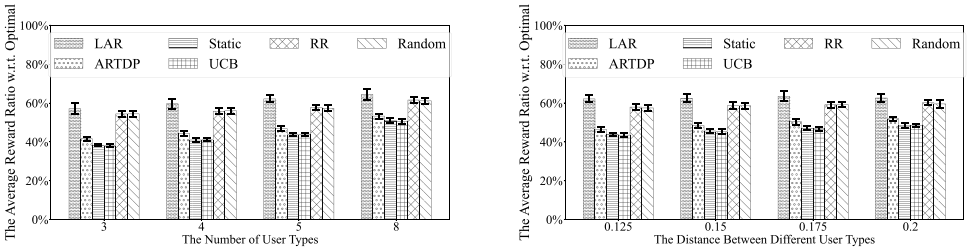
Fig. 2 shows that the performance of our LAR algorithm surpasses all other baselines in various settings. Fig. 2(a) shows how the number of user types k influences these algorithms' performances. Fig. 2(b) shows the impact of distance between $k = 5$ types of users. Increasing the number of user types or enlarging the distance between different types rises the heterogeneity of the user population. Even with a large population variation, the performance gap between LAR and other algorithms is still obvious. That implies that the LAR is more efficient under a small user population heterogeneity. For example, the LAR algorithm largely outperforms the other two when dist=0.125 in Fig. 2(b). In Figs. 2(c) and 2(d), we see that all four algorithms have similar performance in different state space or action space.

6.2. Real-world user MDP setting:

MovieLens 1M dataset [55]. We utilize the real-world MovieLens 1M dataset to generate user types and MDP parameters. Its 6040 users are pruned and divided to 10 types according to user features, e.g., age, gender, and occupation. We pick movies with more than 800 ratings – 15 in total – as our recommendations (action space) and set the state space size as 15 including an absorbing state. These 10 types of users' ratings on the 15 movies (between $1 - 5$) are set to be positive if rating is no less than 4 and negative otherwise. For each type of users, the frequency of negative feedback for a movie, multiplied by $15/(15 + s)$, is set to be $q_u(s, a)$ for generating the user transition probability function p_u via Eq. (9).

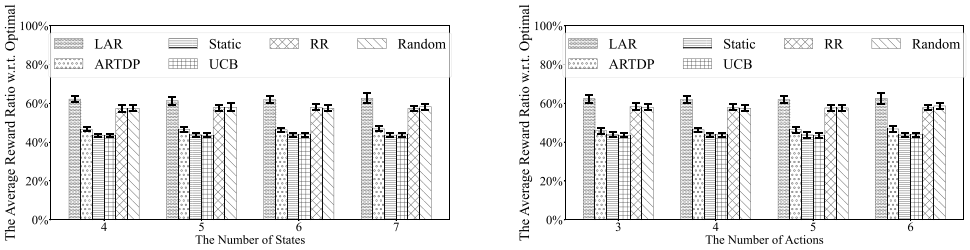
TalkingData AdTracking dataset [56] (16 user types). The dataset contains the log of whether a user downloads a mobile app after clicking the app's ad from TalkingData (the largest independent big data service platform in China), which is composed of 184 million ad clicking interactions. We process the data as follows. We take devices (e.g., iphone 7, huawei mate 7, etc.) as different users in the Dataset and exclude devices whose total clicking time is more than 1000 (due to fraud clicking) or less than 100. In total, we have 16 types of devices (i.e., user types). Then, we consider a small action set $\mathcal{A} = \{0, 1\}$ where 0 stands for not displaying the ad to user and 1 is for displaying the ad, and assume each user have 10 states including the absorbing one. When displaying the ad, we calculate the frequency of downloading the mobile app (as the probability of positive feedback) based on the dataset, and multiply them by $15/(15 + s)$ as $1 - q_u(s, a)$ for generating the user transition probability function p_u via Eq. (9). When not displaying the ad, we assume $p_u(s - 1|s, a) = 0.99$ for all $s > 0$ since not displaying ad is a conservation action. We set the reward of zero action (no ad) as 0, the reward of negative feedback (click but not download) as 0.1, and the reward of positive feedback (click and download) as 1.

Ad display/click dataset on taobao.com [57] (20 actions). This dataset involves a log of whether customers click an ad when they buy products in Taobao (a big e-commerce company in China), which consists of 26 million records from 1.14 million users during 8 days. We process the dataset as follows. We first separate these users into 4 groups according to their ages as different user types in the platform. Then, we choose ad groups with more than 1000 displaying times as the action set, in total 20 actions, and assume the state space is 10 (including the absorbing state). The click-through-rate (CTR) of displaying an ad from ad group a to a user from user group u calculated from the dataset, multiplying $20/(s + 1)$, is set to be $q_u(s, a)$ in Eq. (9) for generating the user transition probability function $q_u(s, a)$.



(a) Impact of #user types (user distance 0.125)

(b) Impact of user distance (5 user types)



(c) Impact of state numbers (4 actions)

(d) Impact of action numbers (6 states)

Fig. 2. Impact of different user population settings.

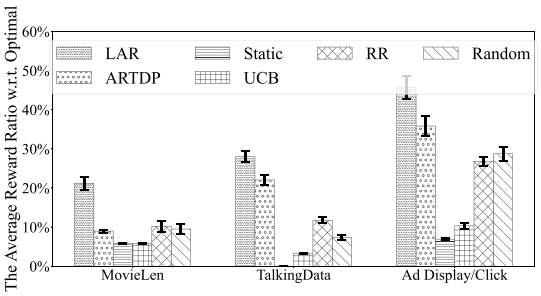


Fig. 3. Algorithms' performance comparison in three real-world datasets' user population.

Fig. 3 shows that the LAR algorithm outperforms other baselines in all three datasets. Specifically, LAR algorithm gains more than twice the reward of the ARTDP algorithm, and four times the reward of the Static and UCB policies.

Comparing the performance of LAR and ARTDP validates the advantage of utilizing other users' history when learning under the risk of user abandonment. Secondly, that our LAR algorithm outperforms the static policy and the UCB algorithm implies that our Markov decision model is more realistic and practical than the threshold model [9] and the multi-armed bandits model [12].

7. Conclusion

This paper formalizes the optimizing recommendation under abandonment risks problem and develops an online learning framework to address its challenges. The framework's theoretical guarantees requires two premises: (1) *the personalized transfer learning oracle* that estimates users' parameters with a PAC accuracy; (2) *the warm start reinforcement learning oracle* whose output error is linearly upper bounded by its input's error. We design the LAR algorithm as a realization of our proposed framework. Its `PersonalizedTL` algorithm implements a sample selection strategy and estimates model parameters via empirical mean estimator. Initialized by the `PersonalizedTL` algorithm's estimation, our `WarmStartRL` algorithm executes fixed rounds of value iterations to optimize the next action. The theoretical guarantee for the `PersonalizedTL` algorithm is achieved by exploiting the "homophily phenomenon" of user population's abandonment behavior, while that for the `WarmStartRL` algorithm is proved through analyzing its value function's convergence rate and Lipschitz continuity. Simulations show that our LAR algorithm outperforms the static policy, the UCB algorithm, and the ARTDP algorithm.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

The work of John C.S. Lui was supported in part by the GRF 14200321. Hong Xie was supported by Chongqing Talents: Exceptional Young Talents Project (cstc2021ycjhbgzxm0195). This work was supported in part by the National Key R&D Program of China (2021YFB1715600), and the National Natural Science Foundation of China (U22B2019).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.peva.2023.102351.

References

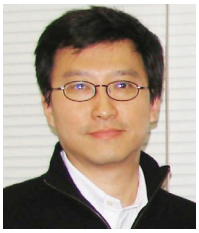
[1] H. Nam, H. Schulzrinne, H. Nam, K.-H. Kim, H. Schulzrinne, M. Varela, H. Nam, H. Schulzrinne, T. Mäki, H. Nam, et al., Youslow: What Influences User Abandonment Behavior for Internet Video? Columbia University Rcp, 2016.
[2] B. Furneaux, L. Rieser, User motivation in application abandonment: A four-drives model, Int. J. Electron. Commer. 26 (1) (2022) 49–89.
[3] G. Broussard, How advertising frequency can work to build online advertising effectiveness, Int. J. Mark. Res. 42 (4) (2000) 1–13.
[4] S. Schmidt, M. Eisend, Advertising repetition: A meta-analysis on effective frequency in advertising, J. Advert. 44 (4) (2015) 415–428.
[5] Y. Shono, Y. Takada, N. Komoda, H. Oiso, A. Hiramatsu, Y. Fukaya, Customer analysis of monthly-charged mobile content aiming at prolonging subscription period, in: Second IEEE International Conference on Computational Cybernetics, 2004, ICCC 2004, IEEE, 2004, pp. 279–284.
[6] T. Iwata, K. Saito, T. Yamada, Recommendation method for extending subscription periods, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 574–579.
[7] J. Parsons, P. Ralph, K. Gallagher, Using viewing time to infer user preference in recommender systems, 2004.
[8] K. Mao, J. Niu, X. Liu, S. Tang, L. Liao, T.S. Chua, A patience-aware recommendation scheme for shared accounts on mobile devices, IEEE Trans. Knowl. Data Eng. 34 (12) (2021) 5571–5585.
[9] S. Schmit, R. Johari, Learning with abandonment, in: International Conference on Machine Learning, 2018, pp. 4516–4524.
[10] A.G. Barto, S.J. Bradtke, S.P. Singh, Learning to act using real-time dynamic programming, Artificial Intelligence 72 (1–2) (1995) 81–138.
[11] J. Cao, W. Sun, Dynamic learning of sequential choice bandit problem under marketing fatigue, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 3264–3271.
[12] B. Kveton, C. Szepesvari, Z. Wen, A. Ashkan, Cascading bandits: Learning to rank in the cascade model, in: International Conference on Machine Learning, 2015, pp. 767–776.
[13] D. Chakrabarti, R. Kumar, F. Radlinski, E. Upfal, Mortal multi-armed bandits, in: Advances in Neural Information Processing Systems, 2009, pp. 273–280.
[14] M. Zoghi, T. Tunys, M. Ghavamzadeh, B. Kveton, C. Szepesvari, Z. Wen, Online learning to rank in stochastic click models, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR.org, 2017, pp. 4199–4208.
[15] J. Cao, W. Sun, Z.-J.M. Shen, M. Ettl, Fatigue-aware bandits for dependent click models, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 04, 2020, pp. 3341–3348.
[16] H. Shayestehmanesh, S. Azami, N.A. Mehta, Dying experts: Efficient algorithms with optimal regret bounds, in: Advances in Neural Information Processing Systems, 2019, pp. 9983–9992.
[17] M. Pecka, T. Svoboda, Safe exploration techniques for reinforcement learning–an overview, in: International Workshop on Modelling and Simulation for Autonomous Systems, Springer, 2014, pp. 357–375.
[18] J. Garcıa, F. Fernández, A comprehensive survey on safe reinforcement learning, J. Mach. Learn. Res. 16 (1) (2015) 1437–1480.
[19] F. Berkenkamp, M. Turchetta, A. Schoellig, A. Krause, Safe model-based reinforcement learning with stability guarantees, in: Advances in Neural Information Processing Systems, 2017, pp. 908–918.
[20] P.S. Thomas, B.C. da Silva, A.G. Barto, S. Giguere, Y. Brun, E. Brunskill, Preventing undesirable behavior of intelligent machines, Science 366 (6468) (2019) 999–1004.
[21] A. Rosenberg, A. Cohen, Y. Mansour, H. Kaplan, Near-optimal regret bounds for stochastic shortest path, in: International Conference on Machine Learning, PMLR, 2020, pp. 8210–8219.
[22] L. Chen, H. Luo, A. Rosenberg, Policy optimization for stochastic shortest path, in: Conference on Learning Theory, PMLR, 2022, pp. 982–1046.
[23] G. Shani, D. Heckerman, R.I. Brafman, C. Boutilier, An MDP-based recommender system., J. Mach. Learn. Res. 6 (9) (2005).
[24] P. Rojanavasu, P. Srinil, O. Pinngern, New recommendation system using reinforcement learning, Special Issue Int. J. Comput. Internet Manag. 13 (SP 3) (2005).
[25] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N.J. Yuan, X. Xie, Z. Li, DRN: A deep reinforcement learning framework for news recommendation, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 167–176.
[26] X. Chen, S. Li, H. Li, S. Jiang, Y. Qi, L. Song, Generative adversarial user model for reinforcement learning based recommendation system, in: International Conference on Machine Learning, PMLR, 2019, pp. 1052–1061.
[27] X. Zhao, X. Zheng, X. Yang, X. Liu, J. Tang, Jointly learning to recommend and advertise, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 3319–3327.
[28] Z. Gharibshah, X. Zhu, A. Hainline, M. Conway, Deep learning for user interest and response prediction in online display advertising, Data Sci. Eng. 5 (1) (2020) 12–26.
[29] Y. Zhao, Y.-H. Zhou, M. Ou, H. Xu, N. Li, Maximizing cumulative user engagement in sequential recommendation: An online optimization perspective, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 2784–2792.

[30] Y. Lin, Y. Liu, F. Lin, P. Wu, W. Zeng, C. Miao, A survey on reinforcement learning for recommender systems, 2021, arXiv preprint arXiv:2109.10665.
[31] P. Auer, R. Ortner, Logarithmic online regret bounds for undiscounted reinforcement learning, in: Advances in Neural Information Processing Systems, 2007, pp. 49–56.
[32] B.C. Csáji, L. Monostori, Value function based reinforcement learning in changing Markovian environments, J. Mach. Learn. Res. 9 (Aug) (2008) 1679–1709.
[33] A.L. Strehl, M.L. Littman, An analysis of model-based interval estimation for Markov decision processes, J. Comput. System Sci. 74 (8) (2008) 1309–1331.
[34] T. Jaksch, R. Ortner, P. Auer, Near-optimal regret bounds for reinforcement learning, J. Mach. Learn. Res. 11 (Apr) (2010) 1563–1600.
[35] K. Asadi, D. Misra, M. Littman, Lipschitz continuity in model-based reinforcement learning, in: International Conference on Machine Learning, 2018, pp. 264–273.
[36] M. Taylor, N. Jong, P. Stone, Transferring instances for model-based reinforcement learning, in: Machine Learning and Knowledge Discovery in Databases, Springer, 2008, pp. 488–505.
[37] A. Lazaric, M. Restelli, A. Bonarini, Transfer of samples in batch reinforcement learning, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 544–551.
[38] A. Tirinzoni, A. Sessa, M. Pirotta, M. Restelli, Importance weighted transfer of samples in reinforcement learning, in: International Conference on Machine Learning, 2018, pp. 4943–4952.
[39] C. Gentile, S. Li, G. Zappella, Online clustering of bandits, in: International Conference on Machine Learning, PMLR, 2014, pp. 757–765.
[40] T.T. Nguyen, H.W. Lauw, Dynamic clustering of contextual multi-armed bandits, in: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, 2014, pp. 1959–1962.
[41] J. Kawale, H.H. Bui, B. Kveton, L. Tran-Thanh, S. Chawla, Efficient thompson sampling for online matrix-factorization recommendation, Adv. Neural Inf. Process. Syst. 28 (2015).
[42] S. Li, A. Karatzoglou, C. Gentile, Collaborative filtering bandits, in: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2016, pp. 539–548.
[43] X. Liu, H. Zhao, T. Yu, S. Li, J.C. Lui, Federated online clustering of bandits, in: Uncertainty in Artificial Intelligence, PMLR, 2022, pp. 1221–1231.
[44] C. Gentile, S. Li, P. Kar, A. Karatzoglou, G. Zappella, E. Etrue, On context-dependent clustering of bandits, in: International Conference on Machine Learning, PMLR, 2017, pp. 1253–1262.
[45] M.L. Littman, Markov games as a framework for multi-agent reinforcement learning, in: Machine Learning Proceedings 1994, Elsevier, 1994, pp. 157–163.
[46] Y. Shoham, K. Leyton-Brown, Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations, Cambridge University Press, 2008.
[47] G. Boutsioukis, I. Partalas, I. Vlahavas, Transfer learning in multi-agent reinforcement learning domains, in: European Workshop on Reinforcement Learning, Springer, 2011, pp. 249–260.
[48] S. Osaki, H. Mine, Some remarks on a Markovian decision problem with an absorbing state, J. Math. Anal. Appl. 23 (2) (1968) 327–333.
[49] S. Carpin, Y.-L. Chow, M. Pavone, Risk aversion in finite Markov decision processes using total cost criteria and average value at risk, in: 2016 IEEE International Conference on Robotics and Automation, IEEE, 2016, pp. 335–342.
[50] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, John Wiley & Sons, 2014.
[51] C. Jin, Z. Allen-Zhu, S. Bubeck, M.I. Jordan, Is Q-learning provably efficient? Adv. Neural Inf. Process. Syst. 31 (2018).
[52] M. Kearns, S. Singh, Near-optimal reinforcement learning in polynomial time, Mach. Learn. 49 (2002) 209–232.
[53] M. McPherson, L. Smith-Lovin, J.M. Cook, Birds of a feather: Homophily in social networks, Annu. Rev. Sociol. 27 (1) (2001) 415–444.
[54] F.S. Perotto, M. Bourgais, B.C. Silva, L. Vercouter, Open problem: Risk of ruin in multiarmed bandits, in: Conference on Learning Theory, PMLR, 2019, pp. 3194–3197.
[55] F.M. Harper, J.A. Konstan, The movielens datasets: History and context, ACM Trans. Interact. Intell. Syst. 5 (4) (2015) 1–19.
[56] Kaggle, TalkingData AdTracking Fraud Detection Challenge; Competition, URL https://www.kaggle.com/competitions/talkingdata-adtracking-fraud-detection/data,
[57] Kaggle, Ad Display/Click Data on Taobao.com; Version 1, URL https://www.kaggle.com/datasets/pavansanagapati/ad-displayclick-data-on-taobaocom.

Xuchuang Wang received the B.Eng. degree from the School of Electronic and Information Engineering, Xi'an Jiaotong University in 2019. He is currently a Ph.D student in the Department of Computer Science and Engineering, The Chinese University of Hong Kong, under the supervision of Prof. John. C.S. Lui. His research interests include online learning and sequential decision making.

Xie Hong is currently a research professor at College of Computer Science, Chongqing University, supported by one hundred talents program of Chongqing University. He completed his Ph.D. degree in the Department of Computer Science and Engineering at The Chinese University of Hong Kong in 2015, proudly under the supervision of Prof. John C.S. Lui. He received his B.Eng. degree from the School of Computer Science and Technology at The University of Science and Technology of China in 2010. He was a postdoctoral research fellow at Department of Computing Science and Engineering, The Chinese University of Hong Kong working closely with Prof. John C.S. Lui. He was also a postdoctoral research fellow working closely with Prof. Richard T.B. Ma, at School of Computing, National University of Singapore. His research interests include online learning, data science and networking. He is a member of IEEE and a member of ACM.

Pinghui Wang received the B.S. and Ph.D. degrees from Xi'an Jiaotong University in 2006 and 2012 respectively. He was a postdoctoral researcher with the Department of Computer Science and Engineering at The Chinese University of Hong Kong and School of Computer Science at McGill University, and was a researcher with Huawei Noah's Ark lab, Hong Kong. He is currently a professor with MOE Key Laboratory for Intelligent Networks and Network Security at Xi'an Jiaotong University, and is also with Shenzhen Research Institute of Xi'an Jiaotong University. His research interests include Internet traffic measurement and modeling, traffic classification, abnormal detection, and online social network measurement.



John C.S. Lui received the Ph.D degree in computer science from the University of California at Los Angeles. He was a Chairman with the CSE Department from 2005 to 2011. He is currently the Choh-Ming Li Chair Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His current research interests include communication networks, system security (e.g., cloud security, mobile security, etc.), network economics, network sciences, largescale distributed systems, and performance evaluation theory. He is an Elected Member of the IFIP WG 7.3, a fellow of the IEEE, and a Croucher Senior Research Fellow. He was a recipient of the various departmental teaching awards and the CUHK Vice-Chancellors Exemplary Teaching Award. He was also a co-recipient of the IFIP WG 7.3 Performance 2005 and IEEEIFIP NOMS 2006 Best Student Paper Awards. He serves in the Editorial Board of the IEEE/ACM TRANSACTIONS ON NETWORKING, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the Journal of Performance Evaluation, and the International Journal of Network Security.