Reinforcement Learning to Rank in E-Commerce Search Engine: Formalization, Analysis, and Application

Yujing Hu
Alibaba Group
Hangzhou, China
yujing.hyj@alibaba-inc.com

Qing Da
Alibaba Group
Hangzhou, China
daqing.dq@alibaba-inc.com

Anxiang Zeng
Alibaba Group
Hangzhou, China
renzhong@taobao.com

Yang Yu
National Key Laboratory for Novel Software Technology, Nanjing University
Nanjing, China
yuy@nju.edu.cn

Yinghui Xu
Artificial Intelligence Department, Zhejiang Cainiao Supply Chain Management Co., Ltd.
Hangzhou, China
renji.xyh@taobao.com

ABSTRACT

In E-commerce platforms such as *Amazon* and *TaoBao*, ranking items in a search session is a typical multi-step decision-making problem. Learning to rank (LTR) methods have been widely applied to ranking problems. However, such methods often consider different ranking steps in a session to be independent, which conversely may be highly correlated to each other. For better utilizing the correlation between different ranking steps, in this paper, we propose to use reinforcement learning (RL) to learn an optimal ranking policy which maximizes the expected accumulative rewards in a search session. Firstly, we formally define the concept of search session Markov decision process (SSMDP) to formulate the multi-step ranking problem. Secondly, we analyze the property of SSMDP and theoretically prove the necessity of maximizing accumulative rewards. Lastly, we propose a novel policy gradient algorithm for learning an optimal ranking policy, which is able to deal with the problem of high reward variance and unbalanced reward distribution of an SSMDP. Experiments are conducted in simulation and *TaoBao* search engine. The results demonstrate that our algorithm performs much better than the state-of-the-art LTR methods, with more than 40% and 30% growth of total transaction amount in the simulation and the real application, respectively.

KEYWORDS

reinforcement learning; online learning to rank; policy gradient

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219846

ACM Reference Format:
Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement Learning to Rank in E-Commerce Search Engine: Formalization, Analysis, and Application. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219846

1 INTRODUCTION

Over past decades, shopping online has become an important part of people's daily life, requiring the E-commerce giants like *Amazon*, *eBay* and *TaoBao* to provide stable and fascinating services for hundreds of millions of users all over the world. Among these services, commodity search is the fundamental infrastructure of these E-commerce platforms, affording users the opportunities to search commodities, browse product information and make comparisons. For example, every day millions of users choose to purchase commodities through *TaoBao* search engine.

In this paper, we focus on the problem of ranking items in large-scale item search engines, which refers to assigning each item a score and sorting the items according to their scores. Generally, a search session between a user and the search engine is a multi-step ranking problem as follows:

(1) the user inputs a query in the blank of the search engine,
(2) the search engine ranks the items related to the query and displays the top K items (e.g., $K = 10$) in a page,
(3) the user makes some operations (e.g., click items, buy some certain item or just request a new page of the same query) on the page,
(4) when a new page is requested, the search engine reranks the rest of the items and display the top K items.

These four steps will repeat until the user buys some items or just leaves the search session. Empirically, a successful transaction always involves multiple rounds of the above process.

The operations of users in a search session may indicate their personal intentions and preference on items. From a statistical view, these signals can be utilized to learn a ranking function which satisfies the users' demand. This motivates the marriage of machine learning and information retrieval, namely the learning to rank

(LTR) methods [9, 19], which learns a ranking function by classification or regression from training data. The major paradigms of supervised LTR methods are pointwise [16, 22], pairwise [4, 5], and listwise [6]. Recently, online learning techniques such as regret minimization [2, 12, 15] have been introduced into the LTR domain for directly learning from user signals. Compared with offline LTR, online LTR avoids the mismatch between manually curated labels, user intent [32] and the expensive cost of creating labeled data sets. Although rigorous mathematical models are adopted for problem formalization [12, 32, 33] and guarantees on regret bounds are established, most of those works only consider a one-shot ranking problem, which means that the interaction between the search engine and each user contains only one round of ranking-and-feedback activity. However, in practice, a search session often contains multiple rounds of interactions and the sequential correlation between each round may be an important factor for ranking, which has not been well investigated.

In this paper, we consider the multi-step sequential ranking problem mentioned above and propose a novel reinforcement learning (RL) algorithm for learning an optimal ranking policy. The major contributions of this paper are as follows.

- We formally define the concept of search session Markov decision process (SSMDP) to formulate the multi-step ranking problem, by identifying the state space, reward function and state transition function.
- We theoretically prove that maximizing accumulative rewards is necessary, indicating that the different ranking steps in a session are tightly correlated rather than independent.
- We propose a novel algorithm named deterministic policy gradient with full backup estimation (DPG-FBE), designed for the problem of high reward variance and unbalanced reward distribution of SSMDP, which could be hardly dealt with even for existing state-of-the-art RL algorithms.
- We empirically demonstrate that our algorithm performs much better than online LTR methods, with more than 40% and 30% growth of total transaction amount in the simulation and the *TaoBao* application, respectively.

The rest of the paper is organized as follows. Section 2 introduces the background of this work. The problem description, analysis of SSMDP and the proposed algorithm are stated in Section 3, 4, 5, respectively. The experimental results are shown in Section 6, and Section 7 concludes the paper finally.

2 BACKGROUND

In this section, we briefly review some key concepts of reinforcement learning and the related work in the online LTR domain. We start from the reinforcement learning part.

2.1 Reinforcement Learning

Reinforcement learning (RL) [28] is a learning technique that an agent learns from the interactions between the environment by trial-and-error. The fundamental mathematical model of reinforcement learning is Markov decision process (MDP).

Definition 2.1 (Markov Decision Process). A Markov decision process is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space of the agent, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition function and $\gamma \in [0, 1]$ is the discount rate.

The objective of an agent in an MDP is to find an optimal policy which maximizes the expected accumulative rewards starting from any state s (typically under the infinite-horizon discounted setting), which is defined by $V^*(s) = \max_\pi \mathbb{E}^\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| s_t = s \right\}$, where $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ denotes any policy of the agent, $\mathbb{E}^\pi$ stands for expectation under policy π, t is the current time step, k is a future time step, and r_{t+k} is the immediate reward at the time step $(t + k)$. This goal is equivalent to finding the optimal state-action value $Q^*(s, a) = \max_\pi \mathbb{E}^\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| s_t = s, a_t = a \right\}$ for any state-action pair (s, a). In finite-horizon setting with a time horizon T, the objective of an agent can be reinterpreted as the finding the optimal policy which maximizes the expected T-step discounted return $\mathbb{E}^\pi \left\{ \sum_{k=0}^{T} \gamma^k r_{t+k} \middle| s_t = s \right\}$ or undiscounted return $\mathbb{E}^\pi \left\{ \sum_{k=0}^{T} r_{t+k} \middle| s_t = s \right\}$[1] in the discounted and undiscounted reward cases, respectively.

An optimal policy can be found by computing the optimal state-value function V^* or the optimal state-action value function Q^*. Early methods such as dynamic programming [28] and temporal-difference learning [30] rely on a table to store and compute the value functions. However, such tabular methods cannot scale up in large-scale state/action space problems due to the curse of dimensionality. Function approximation is widely used to address the scalability issues of RL. By using a parameterized function (e.g., linear functions [20], neural networks [21, 25]) to represent the value function or the policy (a.k.a value function approximation and policy gradient method respectively), the learning problem is transformed to optimizing the function parameters according to reward signals. In recent years, policy gradient methods [24, 26, 29] have drawn much attention in the RL domain. The explicit parameterized representation of policy enables the learning agent to directly search in the policy space and avoids the policy degradation problem of value function approximation.

2.2 Related Work

Early attempt of online LTR can be dated back to the evaluation of RankSVM in online settings [9]. As claimed by Hofmann et al., balancing exploitation and exploration should be a key ability of online LTR methods [8]. The theoretical results in the online learning community (typically in the bandit problem domain) [2, 15] provide rich mathematical tools for online LTR problem formalization and algorithms for efficient exploration, which motivates a lot of online LTR methods. In general, these methods can be divided into two groups. The first is to learn the best ranking function from a function space [8, 32]. For example, Yue and Joachims [32] define a dueling bandit problem in which actions are pairwise comparisons between documents and the goal is to learn a parameterized retrieval function which has sublinear regret performance. The second groups of online LTR methods directly learn the best list under some model of user interactions [23, 27], which can be treated as an assumption on how users act to a ranked list. Representative

[1] The undiscounted return is a special case in discount setting with $\gamma = 1$.

Figure 1: A typical search session in TaoBao. A user starts a session from a query, and has multiple actions to choose, including clicking into an item description, buying an item, turning to the next page, and leaving the session.

models include the cascade model [12, 13, 17, 34], the dependent-click model [10], and the position-based model [14]. Since no single model can entirely capture the behavior of all users, Zoghi et al. [33] recently propose a stochastic click learning framework for online LTR in a broad class of click models.

Our work in this paper is more similar to the first group of online LTR methods which learn ranking functions. However, while most of previous works consider a one-shot ranking problem, we focus on learning a ranking policy in a multi-step ranking problem, which contains multiple rounds of interactions and typically occurs in E-commerce scenarios.

3 PROBLEM FORMULATION

As we mentioned in previous sections, in E-commerce platforms such as *TaoBao* and *TMall*, ranking items given a query is a multi-step decision-making problem, where the search engine should take a ranking action whenever an item page is requested by a user. Figure (1) shows a typical search session between the search engine and a mobile app user in *TaoBao*. In the beginning, the user inputs a query "Cola" into the blank of the search engine and clicks the "Search" button. Then the search engine takes a ranking action and shows the top items related to "Cola" in page 1. The user browses the displayed items and clicks some of them for the details. When no items interest the user or the user wants to check more items for comparisons, the user requests a new item page. The search engine again takes a ranking action and displays page 2. After a certain number of such ranking rounds, the search session will finally end when the user purchases items or just leaves the search session.

3.1 Search Session Modeling

Before we formulate the multi-step ranking problem as an MDP, we define some concepts to formalize the contextual information and user behaviours in a search session, which are the basis for defining the state and state transitions of our MDP.

Definition 3.1 (Top K List). For an item set $\mathcal{D}$, a ranking function f, and a positive integer K ($1 \le K \le |\mathcal{D}|$), the top K list $\mathcal{L}_K(\mathcal{D}, f)$ is an ordered item list $(\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_K)$ which contains the top K items when applying the rank function f to the item set $\mathcal{D}$, where $\mathcal{I}_k$ ($1 \le k \le K$) is the item in position k and for any $k' \ge k$, it is the case that $f(\mathcal{I}_k) > f(\mathcal{I}_{k'})$.

Definition 3.2 (Item Page). For each step t ($t \ge 1$) during a session, the item page p_t is the top K list $\mathcal{L}_K(\mathcal{D}_{t-1}, a_{t-1})$ resulted by applying the ranking action a_{t-1} of the search engine to the set of unranked items $\mathcal{D}_{t-1}$ in the last decision step $(t-1)$. For the initial step $t = 0$, $\mathcal{D}_0 = \mathcal{D}$. For any decision step $t \ge 1$, $\mathcal{D}_t = \mathcal{D}_{t-1} \setminus p_t$.

Definition 3.3 (Item Page History). In a search session, let q be the input query. For the initial decision step $t = 0$, the initial item page history $h_0 = q$. For each later decision step $t \ge 1$, the item page history up to t is $h_t = h_{t-1} \cup \{p_t\}$, where h_{t-1} is the item page history up to the step $(t-1)$ and p_t is the item page of step t.

The item page history h_t contains all information the user observes at the decision step t ($t \ge 0$). Since the item set $\mathcal{D}$ is finite, there are at most $\lceil \frac{|\mathcal{D}|}{K} \rceil$ item pages, and correspondingly at most $\lceil \frac{|\mathcal{D}|}{K} \rceil$ decision steps in a search session. In *TaoBao* and *TMall*, users may choose to purchase items or just leave at different steps of a session. If we treat all possible users as an environment which samples user behaviors, this would mean that after observing any item page history, the environment may terminate a search session with a certain probability of transaction conversion or abandonment. We formally define such two types of probability as follows.

Definition 3.4 (Conversion Probability). For any item page history h_t ($t > 0$) in a search session, let $B(h_t)$ denote the conversion event that a user purchases an item after observing h_t. The conversion probability of h_t, which is denoted by $b(h_t)$, is the averaged probability that $B(h_t)$ occurs when h_t takes place.

Definition 3.5 (Abandon Probability). For any item page history h_t ($t > 0$) in a search session, let $L(h_t)$ denote the abandon event that a user leaves the search session after observing h_t. The abandon probability of h_t, which is denoted by $l(h_t)$, is the averaged probability that $L(h_t)$ occurs when h_t takes place.

Since h_t is the direct result of the agent's action a_{t-1} in the last item page history h_{t-1}, the conversion probability $b(h_t)$ and the abandon probability $l(h_t)$ define how the state of the environment (i.e., the user population) will change after a_{t-1} is taken in h_{t-1}: (1) terminating the search session by purchasing an item in h_t with probability $b(h_t)$; (2) leaving the search session from h_t with probability $l(h_t)$; (3) continuing the search session from h_t with probability $(1 - b(h_t) - l(h_t))$. For convenience, we also define the continuing probability of an item page history.

Definition 3.6 (Continuing Probability). For any item page history h_t ($t \geq 0$) in a search session, let $C(h_t)$ denote the continuation event that a user continues searching after observing h_t. The continuing probability of h_t, which is denoted by $c(h_t)$, is the averaged probability that $C(h_t)$ occurs when h_t takes place.

Obviously, for any item page history h, it holds that $c(h) = 1 - b(h) - l(h)$. Specially, the continuation event of the initial item page history h_0 which only contains the query q is a sure event (i.e., $c(h_0) = 1$) as neither a conversion event nor a abandon event can occur before the first item page is displayed.

3.2 Search Session MDP

Now we are ready to define the instantiated Markov decision process (MDP) for the multi-step ranking problem in a search session, which we call a search session MDP (SSMDP).

Definition 3.7 (Search Session MDP). Let q be a query, $\mathcal{D}$ be the set of items related to q, and K ($K > 0$) be the number of items that can be displayed in a page, the search session MDP (SSMDP) with respect to q, $\mathcal{D}$ and K is a tuple $\mathcal{M} = \langle T, \mathcal{H}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$, where

* $T = \lceil \frac{|\mathcal{D}|}{K} \rceil$ is the maximal decision step of a search session,
* $\mathcal{H} = \bigcup_{t=0}^{T} \mathcal{H}_t$ is the set of all possible item page histories, $\mathcal{H}_t$ is the set of all item page histories up to t ($0 \leq t \leq T$).
* $\mathcal{S} = \mathcal{H}_C \bigcup \mathcal{H}_B \bigcup \mathcal{H}_L$ is the state space, $\mathcal{H}_C = \{C(h_t) | \forall h_t \in \mathcal{H}_t, 0 \leq t < T\}$ is the nonterminal state set that contains all continuation events, $\mathcal{H}_B = \{B(h_t) | \forall h_t \in \mathcal{H}_t, 0 < t \leq T\}$ and $\mathcal{H}_L = \{L(h_t) | \forall h_t \in \mathcal{H}_t, 0 < t \leq T\}$ are two terminal state sets which contain all conversion events and all abandon events, respectively.
* $\mathcal{A}$ is the action space which contains all possible ranking functions of the search engine.
* $\mathcal{R} : \mathcal{H}_C \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function.
* $\mathcal{P} : \mathcal{H}_C \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition function. For any step t ($0 \leq t < T$), any item page history $h_t \in \mathcal{H}_t$, any action $a \in \mathcal{A}$, let $h_{t+1} = (h_t, \mathcal{L}_K(\mathcal{D}_t, a))$. The transition probability from the nonterminal state $C(h_t)$ to any state $s' \in \mathcal{S}$ after taking action a is

$$\mathcal{P}(C(h_t), a, s') = \begin{cases} b(h_{t+1}) & \text{if } s' = B(h_{t+1}), \\ l(h_{t+1}) & \text{if } s' = L(h_{t+1}), \\ c(h_{t+1}) & \text{if } s' = C(h_{t+1}), \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

In an SSMDP, the agent is the search engine and the environment is the population of all possible users. The states of the environment are indication of user status in the corresponding item page histories (i.e., contiuation, abandonment, or transaction conversion). The action space $\mathcal{A}$ can be set differently (e.g., discrete or continuous) according to specific ranking tasks. The state transition function $\mathcal{P}$ is directly based on the conversion probability and abandon probability. The reward function $\mathcal{R}$ highly depends on the goal of a specific task, we will discuss our reward setting in Section 4.2.

4 ANALYSIS OF SSMDP

Before we apply the search session MDP (SSMDP) model in practice, some details need to be further clarified. In this section, we first identify the Markov property of the states in an SSMDP to show that SSMDP is well defined. Then we provide a reward function setting for SSMDP, based on which we perform an analysis on the reward discount rate and show the necessity for a search engine agent to maximize long-time accumulative rewards.

4.1 Markov Property

The Markov property means that a state is able to summarize past sensations compactly in such a way that all relevant information is retained [28]. Formally, the Markov property refers to that for any state-action sequence $s_0, a_0, s_1, a_1, s_2, ..., s_{t-1}, a_{t-1}, s_t$ experienced in an MDP, it holds that

$$\Pr(s_t | s_0, a_0, s_1, a_1, ..., s_{t-1}, a_{t-1}) = \Pr(s_t | s_{t-1}, a_{t-1}). \quad (2)$$

That is to say, the occurring of the current state s_t is only conditional on the last state-action pair (s_{t-1}, a_{t-1}) rather than the whole sequence. Now we show that the states of a search session MDP (SSMDP) also have the Markov property.

Proposition 4.1. *For the search session MDP $\mathcal{M} = \langle T, \mathcal{H}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$ defined in Definition 3.7, any state $s \in \mathcal{S}$ is Markovian.*

Proof. We only need to prove that for any step t ($0 \leq t \leq T$) and any possible state-action sequence $s_0, a_0, s_1, a_1, ..., s_{t-1}, a_{t-1}, s_t$ with respect to t, it holds that

$$\Pr(s_t | s_0, a_0, s_1, a_1, ..., s_{t-1}, a_{t-1}) = \Pr(s_t | s_{t-1}, a_{t-1}).$$

Note that all states except s_t in the sequence $s_0, a_0, s_1, a_1, ..., s_{t-1}, a_{t-1}, s_t$ must be non-terminal states. According to the state definition, for any step t' ($0 < t' < t$), there must be an item page history $h_{t'}$ corresponding to the state $s_{t'}$ such that $s_{t'} = C(h(t'))$. So the state-action sequence can be rewritten as $C(h_0), a_0, C(h_1), a_1, ..., C(h_{t-1}), a_{t-1}, s_t$. For any step t' ($0 < t' < t$), it holds that

$$h_{t'} = (h_{t'-1}, \mathcal{L}_K(\mathcal{D}_{t'-1}, a_{t'-1})),$$

where $\mathcal{L}_K(\mathcal{D}_{t'-1}, a_{t'-1})$ is the top K list (i.e., item page) with respect to the unranked item set $\mathcal{D}_{t'-1}$ and ranking action $a_{t'-1}$ in step $(t'-1)$. Given $h_{t'-1}$, the unranked item set $\mathcal{D}_{t'-1}$ is deterministic. Thus, $h_{t'}$ is the necessary and unique result of the state-action pair $(C(h_{t'-1}), a_{t'-1})$. Therefore, the event $(C(h_{t'-1}), a_{t'-1})$ can be equivalently represented by the event $h_{t'}$, and the following derivation can be conducted:

$$\begin{aligned} &\Pr(s_t | s_0, a_0, s_1, a_1, ..., s_{t-1}, a_{t-1}) \\ =&\Pr(s_t | C(h_0), a_0, C(h_1), a_1, ..., C(h_{t-1}), a_{t-1}) \\ =&\Pr(s_t | h_1, h_2, ..., h_{t-1}, C(h_{t-1}), a_{t-1}) \\ =&\Pr(s_t | h_{t-1}, C(h_{t-1}), a_{t-1}) \\ =&\Pr(s_t | C(h_{t-1}), a_{t-1}) \\ =&\Pr(s_t | s_{t-1}, a_{t-1}). \end{aligned}$$

The third step of the derivation holds because for any step t' ($0 < t' < t$), $h_{t'-1}$ is contained in $h_{t'}$. Similarly, the fourth step holds because $C(h_{t-1})$ contains the occurrence of h_{t-1}. □

4.2 Reward Function

In a search session MDP $\mathcal{M} = \langle T, \mathcal{H}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$, the reward function $\mathcal{R}$ of is a quantitative evaluation of the action performance in

each state. Specifically, for any nonterminal state $s \in \mathcal{H}_C$, any action $a \in \mathcal{A}$, and any other state $s' \in \mathcal{S}$, $\mathcal{R}(s, a, s')$ is the expected value of the immediate rewards that numerically characterize the user feedback when action a is taken in s and the state is changed to s'. Therefore, we need to translate user feedback to numeric reward values that a learning algorithm can understand.

In the online LTR domain, user clicks are commonly adopted as a reward metric [10, 14, 33] to guide learning algorithms. However, in E-commerce scenarios, successful transactions between users (who search items) and sellers (whose items are ranked by the search engine) are more important than user clicks. Thus, our reward setting is designed to encourage more successful transactions. For any decision step t ($0 \le t < T$), any item page history $h_t \in \mathcal{H}_t$, and any action $a \in \mathcal{A}$, let $h_{t+1} = (h_t, \mathcal{L}_K(D_t, a))$. Recall that after observing the item page history h_{t+1}, a user will purchase an item with a conversion probability $b(h_{t+1})$. Although different users may choose different items to buy, from a statistical view, the deal prices of the transactions occurring in h_{t+1} must follow an underlying distribution. We use $m(h_{t+1})$ to denote the expected deal price of h_{t+1}. Then for the nonterminal state $C(h_t)$ and any state $s' \in \mathcal{S}$, the reward $\mathcal{R}(C(h_t), a, s')$ is set as follows:

$$\mathcal{R}(C(h_t), a, s') = \begin{cases} m(h_{t+1}) & \text{if } s' = B(h_{t+1}), \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $B(h_{t+1})$ is the terminal state which represents the conversion event of h_{t+1}. The agent will receive a positive reward from the environment only when its ranking action leads to a successful transaction. In all other cases, the reward is zero. It should be noted that the expected deal price of any item page history is most probably unknown beforehand. In practice, the actual deal price of a transaction can be directly used as the reward signal.

4.3 Discount Rate

The discount rate γ is an important parameter of an MDP which defines the importance of future rewards in the objective of the agent (defined in Section 2.1). For the search session MDP (SSMDP) defined in this paper, the choice of the discount rate γ brings out a fundamental question: "Is it necessary for the search engine agent to consider future rewards when making decisions?" We will find out the answer and determine an appropriate value of the discount rate by analyzing how the objective of maximizing long-time accumulative rewards is related to the goal of improving the search engine's economic performance.

Let $\mathcal{M} = \langle T, \mathcal{H}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$ be a search session MDP with respect to a query q, an item set $\mathcal{D}$ and an integer K ($K > 0$). Given a fixed deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ of the agent[2], denote the item page history occurring at step t ($0 \le t \le T$) under π by h_t^π. We enumerate all possible states that can be visited in a search session under π in Figure 2. For better illustration, we show all item page histories (marked in red) in the figure. Note that they are not the states of the SSMDP $\mathcal{M}$. Next, we will rewrite $C(h_t^\pi)$, $c(h_t^\pi)$, $b(h_t^\pi)$, and $m(h_t^\pi)$ as C_t^π, c_t^π, b_t^π, and m_t^π for simplicity.

[2]More accurately, the policy π is a mapping from the nonterminal state set $\mathcal{H}_C$ to the action space $\mathcal{A}$. Our conclusion in this paper also holds for stochastic policies, but we ommit the discussion due to space limitation.

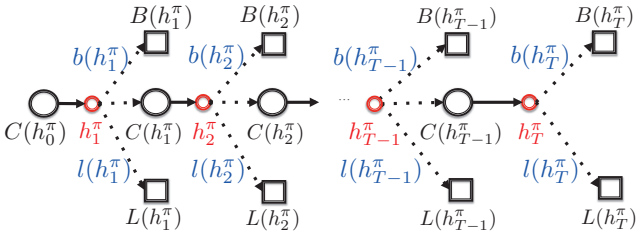

Figure 2: All states that can be visited under policy π. The black circles are nonterminal states and the black squares are terminal states. The red circles are item page histories. The solid black arrow starting from each nonterminal state represents the execution of the policy π. The dotted arrows from each item page history are state transitions, with the corresponding transition probabilities marked in blue.

Without loss of generality, we assume the discount rate of the SSMDP $\mathcal{M}$ is γ ($0 \le \gamma \le 1$). Denote the state value function (i.e., expected accumulative rewards) under γ by V_γ^π . For each step t ($0 \le t < T$), the state value of the nonterminal state C_t^π is

$$\begin{aligned} V_\gamma^\pi(C_t^\pi) &= \mathbb{E}^\pi\Big\{ \sum_{k=1}^{T-t} \gamma^{k-1} r_{t+k} \Big| C_t^\pi \Big\} \\ &= \mathbb{E}^\pi\Big\{ r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-t-1} r_T \Big| C_t^\pi \Big\}, \end{aligned} \tag{4}$$

where for any k ($1 \le k \le T-t$), r_{t+k} is the immediate reward received at the future step $(t+k)$ in the item page history h_{t+k}^π. According to the reward function in Equation (3), the expected value of the immediate reward r_{t+k} under π is

$$\mathbb{E}^\pi\big\{ r_{t+k} \big\} = b_{t+k}^\pi m_{t+k}^\pi, \tag{5}$$

where $m_{t+k}^\pi = m(h_{t+k}^\pi)$ is the expected deal price of the item page history h_{t+k}^π. However, since $V_\gamma^\pi(C_t^\pi)$ is the expected discounted accumulative rewards on condition of the state C_t^π, the probability that the item page history h_{t+k}^π is reached when C_t^π is visited should be taken into account. Denote the reaching probability from C_t^π to h_{t+k}^π by $\Pr(C_t^\pi \to h_{t+k}^\pi)$, it can be computed as follows according to the state transition function in Equation (1):

$$\Pr(C_t^\pi \to h_{t+k}^\pi) = \begin{cases} 1.0 & k = 1, \\ \Pi_{j=1}^{k-1} c_{t+j}^\pi & 1 < k \le T-t. \end{cases} \tag{6}$$

The reaching probability from C_t^π to h_{t+1}^π is 1 since h_{t+1}^π is the directly result of the state action pair $(C_t^\pi, \pi(C_t^\pi))$. For other future item page histories, the reaching probability is the product of all continuing probabilities along the path from C_{t+1}^π to C_{t+k-1}^π. By taking Equations (5) and (6) into Equation (4), $V_\gamma^\pi(C_t^\pi)$ can be further computed as follows:

$$\begin{aligned} V_\gamma^\pi(C_t^\pi) &= \mathbb{E}^\pi\big\{ r_{t+1} \big| C_t^\pi \big\} + \gamma \mathbb{E}^\pi\big\{ r_{t+2} \big| C_t^\pi \big\} + \cdots \\ &\quad + \gamma^{k-1} \mathbb{E}^\pi\big\{ r_{t+k} \big| C_t^\pi \big\} + \cdots + \gamma^{T-t-1} \mathbb{E}^\pi\big\{ r_T \big| C_t^\pi \big\} \\ &= \sum_{k=1}^{T-t} \gamma^{k-1} \Pr(C_t^\pi \to h_{t+k}^\pi) b_{t+k}^\pi m_{t+k}^\pi \\ &= b_{t+1}^\pi m_{t+1}^\pi + \sum_{k=2}^{T-t} \gamma^{k-1} \Big(\big(\Pi_{j=1}^{k-1} c_{t+j}^\pi \big) b_{t+k}^\pi m_{t+k}^\pi \Big). \end{aligned} \tag{7}$$

With the conversion probability and the expected deal price of each item page history in Figure 2, we can also derive the expected gross merchandise volume (GMV) lead by the search engine agent in a search session under the policy π as follows:

$$\begin{aligned}\mathbb{E}^{\pi}_{\text{gmv}} &= b^{\pi}_1 m^{\pi}_1 + c^{\pi}_1 b^{\pi}_2 m^{\pi}_2 + \cdots + \big(\Pi^{T}_{k=1} c^{\pi}_k\big) b^{\pi}_T m^{\pi}_T \\ &= b^{\pi}_1 m^{\pi}_1 + \sum_{k=2}^{T} \big(\Pi^{k-1}_{j=1} c^{\pi}_j\big) b^{\pi}_k m^{\pi}_k. \end{aligned} \tag{8}$$

By comparing Equations (7) and (8), it can be easily found that $\mathbb{E}^{\pi}_{gmv} = V^{\pi}_{\gamma}(C^{\pi}_0)$ when the discount rate $\gamma = 1$. That is to say, when $\gamma = 1$, maximizing the expected accumulative rewards directly leads to the maximization of the expected GMV. However, when $\gamma < 1$, maximizing the value function V^{π}_{γ} cannot necessarily maximize $\mathbb{E}^{\pi}_{gmv}$ since the latter is an upper bound of $V^{\pi}_{\gamma}(C^{\pi}_0)$.

Proposition 4.2. *Let $\mathcal{M} = \langle T, \mathcal{H}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}\rangle$ be a search session MDP. For any deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ and any discount rate γ ($0 \le \gamma \le 1$), it is the case that $V^{\pi}_{\gamma}(C(h_0)) \le \mathbb{E}^{\pi}_{gmv}$, where V^{π}_{γ} is state value function defined in Equation (4), $C(h_0)$ is the initial nonterminal state of a search session, $\mathbb{E}^{\pi}_{gmv}$ is the expected gross merchandise volume (GMV) of π defined in Equation (8). Only when $\gamma = 1$, we have $V^{\pi}_{\gamma}(C(h_0)) = \mathbb{E}^{\pi}_{gmv}$.*

Proof. The proof is trivial since the difference between $\mathbb{E}^{\pi}_{gmv}$ and $V^{\pi}_{\gamma}(C(h_0))$, namely $\sum^{T}_{k=2}(1-\gamma^{k-1})\big(\Pi^{k-1}_{j=1} c^{\pi}_j\big) b^{\pi}_k m^{\pi}_k$, is always positive when $\gamma < 1$. □

Now we can give the answer to the question proposed in the beginning of this section: considering future rewards in a search session MDP is necessary since maximizing the undiscounted expected accumulative rewards can optimize the performance of the search engine in the aspect of GMV. The sequential nature of our multi-step ranking problem requires the ranking decisions at different steps to be optimized integratedly rather than independently.

5 ALGORITHM

In this section, we propose a policy gradient algorithm for learning an optimal ranking policy in a search session MDP (SSMDP). We resort to the policy gradient method since directly optimizing a parameterized policy function addresses both the policy representation issue and the large-scale action space issue of an SSMDP. Now we briefly review the policy gradient method in the context of SSMDP. Let $\mathcal{M} = \langle T, \mathcal{H}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}\rangle$ be an SSMDP, π_θ be the policy function with the parameter θ. The objective of the agent is to find an optimal parameter which maximizes the expectation of the T-step returns along all possible trajectories

$$J(\theta) = \mathbb{E}_{\tau\sim\rho_\theta}\big\{R(\tau)\big\} = \mathbb{E}_{\tau\sim\rho_\theta}\Big\{\sum_{t=0}^{T-1} r_t\Big\}, \tag{9}$$

where τ is a trajectory like $s_0, a_0, r_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T$ and follows the trajectory distribution ρ_θ under the policy parameter θ, $R(\tau) = \sum^{T-1}_{t=0} r_t$ is the T-step return of the trajectory τ. Note that if the terminal state of a trajectory is reached in less than T steps, the sum of the rewards will be truncated in that state. The gradient of the target $J(\theta)$ with respect to θ is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\rho_\theta}\Big\{\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s_t, a_t) R^T_t(\tau)\Big\}, \tag{10}$$

where $R^T_t(\tau) = \sum^{T-1}_{t'=t} r_{t'}$ is the sum of rewards from step t to the terminal step T in the trajectory τ. This gradient leads to the well-known REINFORCE algorithm [31]. The policy gradient theorem proposed by Sutton et al. [29] provides a framework which generalizes the REINFORCE algorithm. In general, the gradient of $J(\theta)$ can be written as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\rho_\theta}\Big\{\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s_t, a_t) Q^{\pi_\theta}(s_t, a_t)\Big\},$$

where Q^{π_θ} is the state-action value function under the policy π_θ. If π_θ is deterministic, the gradient of $J(\theta)$ can be rewritten as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\rho_\theta}\Big\{\sum_{t=0}^{T-1} \nabla_\theta \pi_\theta(s_t) \nabla_a Q^{\pi_\theta}(s_t, a)\Big|_{a=\pi_\theta(s_t)}\Big\}.$$

Silver et al. [26] show that the deterministic policy gradient is the limiting case of the stochastic policy gradient as policy variance tends to zero. The value function Q^{π_θ} can be estimated by temporal-difference learning (e.g., actor-critic methods [28]) aided by a function approximator Q^w with the parameter w which minimizes the mean squared error $\text{MSE}(w) = ||Q^w - Q^{\pi_\theta}||^2$.

5.1 The DPG-FBE Algorithm

Instead of using stochastic policy gradient algorithms, we rely on the deterministic policy gradient (DPG) algorithm [26] to learn an optimal ranking policy in an SSMDP since from a practical viewpoint, computing the stochastic policy gradient may require more samples, especially if the action space has many dimensions. However, we have to overcome the difficulty in estimating the value function Q^{π_θ}, which is caused by the high variance and unbalanced distribution of the immediate rewards in each state. As indicated by Equation (3), the immediate reward of any state-action pair (s, a) is zero or the expected deal price $m(h)$ of the item history page h resulted by (s, a). Firstly, the reward variance is high because the deal price $m(h)$ normally varies over a wide range. Secondly, the immediate reward distribution of (s, a) is unbalanced because the conversion events lead by (s, a) occur much less frequently than the two other cases (i.e., abandon and continuation events) which produce zero rewards. Note that the same problem also exists for the T-step returns of the trajectories in an SSMDP since in any possible trajectory, only the reward of the last step may be nonzero. Therefore, estimating Q^{π_θ} by Monte Carlo evaluation or temporal-difference learning may cause inaccurate update of the value function parameters and further influence the optimization of the policy parameter.

Our way for solving the above problem is similar to the model-based reinforcement learning approaches [3, 11], which maintain an approximate model of the environment to help with performing reliable updates of value functions. According to the Bellman Equation [28], the state-action value of any state-action pair (s, a)

under any policy π is

$$Q^{\pi_\theta}(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s')\Big(\mathcal{R}(s,a,s') + \max_{a'} Q^{\pi_\theta}(s',a')\Big), \quad (11)$$

The right-hand side of Equation (11) can be denoted by $\mathcal{T}Q^{\pi_\theta}(s,a)$, where $\mathcal{T}$ is the Bellman operator with respect to the policy π_θ. Let h' be the next item page history resulted by (s,a). Only the states $C(h')$, $B(h')$, and $L(h')$ can be transferred to from (s,a) with nonzero probability. Among these three states, only $B(h')$ involves a nonzero immediate reward and $C(h')$ involves a nonzero Q-value. So the above equation can be simplified to

$$Q^{\pi_\theta}(s,a) = b(h')m(h') + c(h') \max_{a'} Q^{\pi_\theta}\left(C(h'), a'\right), \quad (12)$$

where $b(h')$, $c(h')$, and $m(h')$ are the conversion probability, continuing probability and expected deal price of h', respectively. Normally, the value function Q^{π_θ} can be approximated by a parameterized function Q^w with an objective of minimizing the mean squared error (MSE)

$$\text{MSE}(w) = ||Q^w - Q^{\pi_\theta}||^2 = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \Big(Q^w(s,a) - Q^{\pi_\theta}(s,a)\Big)^2.$$

The derivative of $\text{MSE}(w)$ with respect to the parameter w is

$$\nabla_w \text{MSE}(w) = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \Big(Q^{\pi_\theta}(s,a) - Q^w(s,a)\Big)\nabla_w Q^w(s,a).$$

However, since $Q^{\pi_\theta}(s,a)$ is unknown, we cannot get the accurate value of $\nabla_w \text{MSE}(w)$. One way for solving this problem is to replace Q^{π_θ} with $\mathcal{T}Q^w$ and approximately compute $\nabla_w \text{MSE}(w)$ by

$$\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \Big(b(h')m(h') + c(h') \max_{a'} Q^w(s',a') - Q^w(s,a)\Big)\nabla_w Q^w(s,a),$$

where $s' = C(h')$ is the state of continuation event of h'. Every time a state-action pair (s,a) as well as its next item page history h' is observed, w can be updated in a full backup manner:

$$\Delta w \leftarrow \alpha_w \nabla_w Q^w(s,a)\Big(b(h')m(h') + c(h')Q^w(s',a') - Q^w(s,a)\Big),$$

where α_w is a learning rate and $a' = \pi_\theta(s')$. With this full backup updating method, the sampling errors caused by immediate rewards or returns can be avoided. Furthermore, the computational cost of full backups in our problem is almost equal to that of one-step sample backups (e.g., Q-learning [30]).

Our policy gradient algorithm is based on the deterministic policy gradient theorem [26] and the full backup estimation of the Q-value functions. Unlike previous works which entirely model the reward and state transition functions [3, 11], we only need to build the conversion probability model $b(\cdot)$, the continuing probability model $c(\cdot)$, and the expected deal price model $m(\cdot)$ of the item page histories in an SSMDP. These models can be trained using online or offline data by any possible statistical learning method. We call our algorithm Deterministic Policy Gradient with Full Backup Estimation (DPG-FBE) and show its details in Algorithm 1.

As shown in this table, the parameters θ and w will be updated after any search session between the search engine agent and users. Exploration (at line 3) can be done by, but not limited to, ϵ-greedy (in discrete action case) or adding random noise to the output of π_θ (in continuous action case). Although we have no assumptions on the specific models used for learning the actor π_θ and the critic Q^w in Algorithm 1, nonlinear models such as neural networks are preferred due to the large state/action space of an SSMDP. To solve the convergence problem and ensure a stable learning process, a replay buffer and target updates are also suggested [18, 21].

Algorithm 1: Deterministic Policy Gradient with Full Backup Estimation (DPG-FBE)

Input: Learning rate α_θ and α_w, pretrained conversion probability model b, continuing probability model c, and expected deal price model m of item page histories

1 Initialize the actor π_θ and the critic Q^w with parameter θ and w;
2 **foreach** *search session* **do**
3 Use π_θ to sample a ranking action at each step with exploration;
4 Get the trajectory τ of the session with its final step index t;
5 $\Delta w \leftarrow 0$, $\Delta\theta \leftarrow 0$;
6 **for** $k = 0, 1, 2, \ldots, t-1$ **do**
7 $(s_k, a_k, r_k, s_{k+1}) \leftarrow$ the sample tuple at step k;
8 $h_{k+1} \leftarrow$ the item page history of s_k;
9 **if** $s_{k+1} = B(h_{k+1})$ **then**
10 Update the models b, c, and m with the samples $(h_{k+1}, 1)$, $(h_{k+1}, 0)$, and (h_{k+1}, r_k), respectively;
11 **else**
12 Update the models b and c with the samples $(h_{k+1}, 0)$ and $(h_{k+1}, 1)$, respectively;
13 $s' \leftarrow C(h_{k+1})$, $a' \leftarrow \pi_\theta(s')$;
14 $p_{k+1} \leftarrow b(h_{k+1})m(h_{k+1})$;
15 $\delta_k \leftarrow p_{k+1} + c(h_{k+1})Q^w(s',a') - Q^w(s_k,a_k)$;
16 $\Delta w \leftarrow \Delta w + \alpha_w \delta_k \nabla_w Q^w(s_k,a_k)$;
17 $\Delta\theta \leftarrow \Delta\theta + \alpha_\theta \nabla_\theta \pi_\theta(s_k) \nabla_a Q^w(s_k,a_k)$;
18 $w \leftarrow w + \Delta w/t$, $\theta \leftarrow \theta + \Delta\theta/t$;

6 EXPERIMENTS

In this section, we conduct two groups of experiments: a simulated experiment in which we construct an online shopping simulator and test our algorithm DPG-FBE as well as some state-of-the-art online learning to rank (LTR) algorithms, and a real application in which we apply our algorithm in *TaoBao*, one of the largest E-commerce platforms in the world.

6.1 Simulation

The online shopping simulator is constructed based on the statistical information of items and user behaviors in *TaoBao*. An item is represented by a n-dim ($n > 0$) feature vector $\boldsymbol{x} = (x_1, \ldots, x_n)^\top$ and a ranking action of the search engine is a n-dim weight vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)^\top$. The ranking score of the item $\boldsymbol{x}$ under the ranking action $\boldsymbol{\mu}$ is the inner product $\boldsymbol{x}^\top\boldsymbol{\mu}$ of the two vectors. We choose 20 important features related to the item category of *dress* (e.g., price and quality) and generate an item set $\mathcal{D}$ by sampling 1000 items from a distribution approximated with all the items of the dress category. Each page contains 10 items so that there are at most 100 ranking rounds in a search session. In each ranking round, the user operates on the current item page (such as clicks, abandonment, and purchase) are simulated by a user behavior model, which is constructed from the user behavior data of the dress items in *TaoBao*. The simulator outputs the probability of each possible user operation given the recent item pages examined

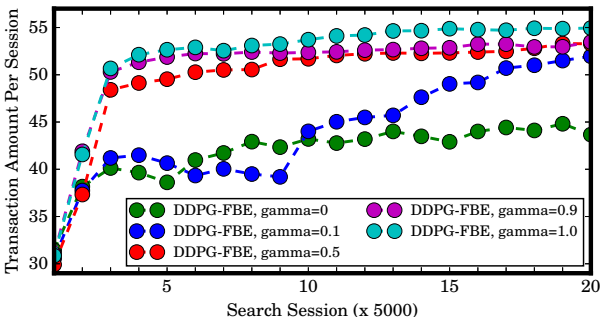

Figure 3: The learning performance of the DDPG-FBE algorithm in the simulation experiment

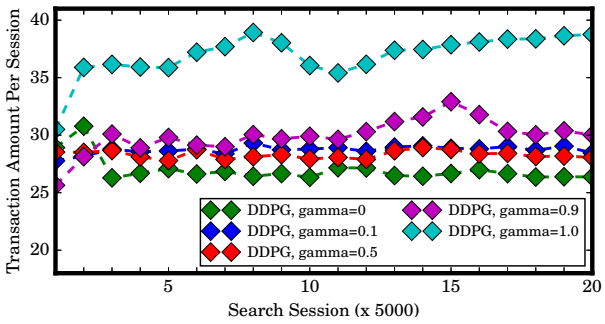

Figure 4: The learning performance of the DDPG algorithm in the simulation experiment

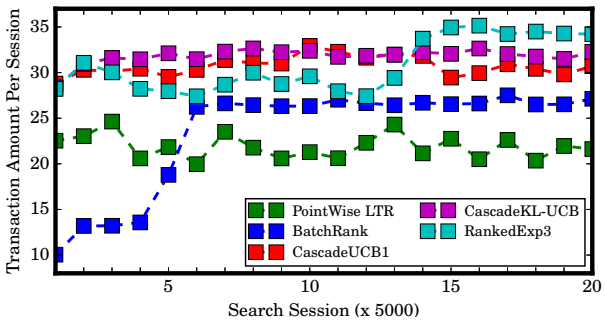

Figure 5: The learning performance of five online LTR algorithms in the simulation experiment

by the user. A search session will end when the user purchases one item or leaves.

Our implementation of the DPG-FBE algorithm is a deep RL version (DDPG-FBE) which adopts deep neural networks (DNN) as the policy and value function approximators (i.e., actor and critic). We also implement the deep DPG algorithm (DDPG) [18]. The state of environment is represented by a 180-dim feature vector extracted from the last 4 item pages of the current search session. The actor and critic networks of the two algorithms have two full connected hidden layers with 200 and 100 units, respectively. We adopt *relu* and *tanh* as the activation functions for the hidden layers and the output layers of all networks. The network parameters are optimized by Adam with a learning rate of 10^{-5} for the actor and 10^{-4} for the critic. The parameter τ for the soft target updates [18] is set to 10^{-3}. We test the performance of the two algorithms under different settings of the discount rate γ. Five online LTR algorithms, point-wise LTR, BatchRank [33], CascadeUCB1 [12], CascadeKL-UCB [12], and RankedExp3 [23] are implemented for comparison. Like the two RL algorithms, the point-wise LTR method implemented in our simulation also learns a parameterized function which outputs a ranking weight vector in each state of a search session. We choose DNN as the parameterized function and use the logistic regression algorithm to train the model, with an objective function that approximates the goal of maximizing GMV. The four other online LTR algorithms are regret minimization algorithms which are based on variants of the bandit problem model. The test of each algorithm contains 100,000 search sessions and the transaction amount of each session is recorded. Results are averaged over 50 runs and are shown in Figures 3, 4, and 5.

Now let us first examine the figure of DDPG-FBE. It can be found that the performance of DDPG-FBE is improved as the discount rate γ increases. The learning curve corresponding to the setting $\gamma = 0$ (the green one) is far below other curves in Fig. 3, which indicates the importance of delay rewards. The theoretical result in Section 4 is empirically verified since the DDPG-FBE algorithm achieves the best performance when $\gamma = 1$, with 2% growth of transaction amount per session compared to the second best performance. Note that in E-commerce scenarios, even 1% growth is considerable. The DDPG algorithm also performs the best when $\gamma = 1$, but it fails to learn as well as the DDPG-FBE algorithm. As shown in Fig. 4, all the learning curves of DDPG are under the value 40. The point-wise LTR method also outputs a ranking weight vector while the other four online LTR algorithms can directly output a ranked item list according to their own ranking mechanisms. However, as we can observe in Fig. 5, the transaction amount lead by each of the algorithms is much smaller than that lead by DDPG-FBE and DDPG. This is not surprising since none of these algorithms are designed for the multi-step ranking problem where the ranking decisions at different steps should be optimized integratedly.

6.2 Application

We apply our algorithm in *TaoBao* search engine for providing online realtime ranking service. The searching task in *TaoBao* is characterized by high concurrency and large data volume. In each second, the *TaoBao* search engine should respond to hundreds of thousands of users' requests in concurrent search sessions and simultaneously deal with the data produced from user behaviours. On sale promotion days such as the *TMall Double* 11 *Global Shopping Festival*[3], both the volume and producing rate of the data would be multiple times larger than the daily values.

In order to satisfy the requirement of high concurrency and the ability of processing massive data in *TaoBao*, we design a data stream-driven RL ranking system for implementing our algorithm DPG-FBE. As shown in Figure 6, this system contains five major components: a query planner, a ranker, a log center, a reinforcement learning component, and an online KV system. The work

[3]This refers to November the 11-th of each year. On that day, most sellers in *TaoBao* and *TMall* carry out sale promotion and billions of people in the world join in the online shopping festival

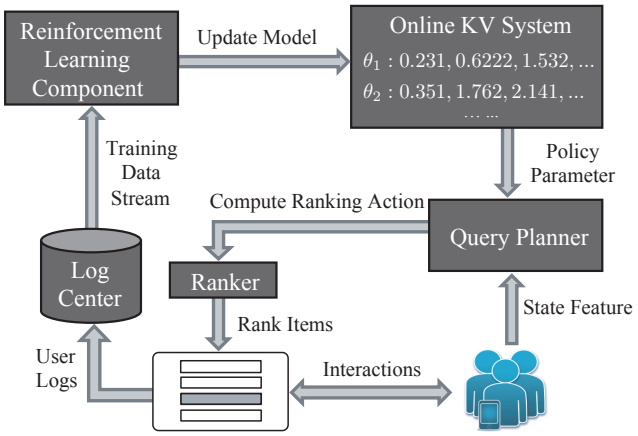

Figure 6: RL ranking system of *TaoBao* search engine

flow of our system mainly consists of two loops. The first one is an online acting loop (in the right bottom of Figure 6), in which the interactions between the search engine and *TaoBao* users take place. The second one is a learning loop (on the left of the online acting loop in Figure 6) where the training process happens. The two working loops are connected through the log center and the online KV system, which are used for collecting user logs and storing the ranking policy model, respectively. In the first loop, every time a user requests an item page, the query planner will extract the state feature, get the parameters of the ranking policy model from the online KV system, and compute a ranking action for the current state (with exploration). The ranker will apply the computed action to the unranked items and display the top K items (e.g., $K = 10$) in an item page, where the user will give feedback. In the meanwhile, the log data produced in the online acting loop is injected into the learning loop for constructing training data source. In the log center, the user logs collected from different search sessions are transformed to training samples like (s, a, r, s'), which are output continuously in the form of data stream and utilized by our algorithm to update the policy parameters in the learning component. Whenever the policy model is updated, it will be rewritten to the online KV system. Note that the two working loops in our system work in parallel but asynchronously, because the user log data generated in any search session cannot be utilized immediately.

The linear ranking mode used in our simulation is also adopted in this *TaoBao* application. The ranking action of the search engine is a 27-dim weight vector. The state of the environment is represented by a 90-dim feature vector, which contains the item page features, user features and query features of the current search session. We add user and query information to the state feature since the ranking service in *TaoBao* is for any type of users and there is no limitation on the input queries. We still adopt neural networks as the policy and value function approximators. However, to guarantee the online realtime performance and quick processing of the training data, the actor and critic networks have much smaller scale than those used in our simulation, with only 80 and 64 units in each of their two fully connected hidden layers, respectively. We implement DDPG and DDPG-FBE algorithms in our system and conduct one-week A/B test to compare the two algorithms. In each day of the test, the DDPG-FBE algorithm can lead to 2.7% $\sim$ 4.3% more transaction amount than the DDPG algorithm [4]. The DDPG-FBE algorithm was also used for online ranking service on the TMall Double 11 Global Shopping Festival of 2016. Compared with the baseline algorithm (an LTR algorithm trained offline), our algorithm achieved more than 30% growth in GMV at the end of that day.

7 CONCLUSIONS

In this paper, we propose to use reinforcement learning (RL) for ranking control in E-commerce searching scenarios. Our contributions are as follows. Firstly, we formally define the concept of search session Markov decision process (SSMDP) to formulate the multi-step ranking problem in E-commerce searching scenarios. Secondly, we analyze the property of SSMDP and theoretically prove the necessity of maximizing accumulative rewards. Lastly, we propose a novel policy gradient algorithm for learning an optimal ranking policy in an SSMDP. Experimental results in simulation and *TaoBao* search engine show that our algorithm perform much better than the state-of-the-art LTR methods in the multi-step ranking problem, with more than 40% and 30% growth in gross merchandise volume, respectively. In the future, we will also try to handle more challenges, such as the dynamic environment [7], arised in the recommendation system applications.

ACKNOWLEDGMENTS

We would like to thank our colleague – Yusen Zhan for useful discussions and supports of this work. We would also like to thank the anonymous referees for their valuable comments and helpful suggestions. Yang Yu is supported by Jiangsu SF (BK20160066).

REFERENCES

[1] Alizila. 2017. Joe Tsai Looks Beyond Alibaba's RMB 3 Trillion Milestone. http://www.alizila.com/joe-tsai-beyond-alibabas-3-trillion-milestone/.
[2] Peter Auer. 2002. Using Confidence Bounds for Exploitation-Exploration Tradeoffs. *Journal of Machine Learning Research* 3, Nov (2002), 397–422.
[3] Ronen I. Brafman and Moshe Tennenholtz. 2002. R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *Journal of Machine Learning Research* 3 (2002), 213–231.
[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank Using Gradient Descent. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*. 89–96.
[5] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. 2006. Adapting Ranking SVM to Document Retrieval. In *Proceedings of the 29th Annual International Conference on Research and Development in Information Retrieval (SIGIR'06)*. 186–193.
[6] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th International Conference on Machine Learning (ICML'07)*. ACM, 129–136.
[7] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. 2018. Stablizing reinforcement learning in dynamic environment with application to online recommendation. In *Proceedings of the 24th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'18)*.
[8] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2013. Balancing Exploration and Exploitation in Listwise and Pairwise Online Learning to Rank for Information Retrieval. *Information Retrieval* 16, 1 (2013), 63–90.
[9] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*. ACM, 133–142.

[4] We cannot report the accurate transaction amount due to the information protection rule of Alibaba. Here we provide a reference index: the GMV achieved by Alibaba's China retail marketplace platforms surpassed 476 billion U.S. dollars in the fiscal year of 2016 [1].

[10] Sumeet Katariya, Branislav Kveton, Csaba Szepesvari, Claire Vernade, and Zheng Wen. 2017. Stochastic Rank-1 Bandits. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017*. 392–401.

[11] Michael Kearns and Satinder Singh. 2002. Near-Optimal Reinforcement Learning in Polynomial Time. *Machine Learning* 49, 2-3 (2002), 209–232.

[12] Branislav Kveton, Csaba Szepesvari, Zheng Wen, and Azin Ashkan. 2015. Cascading Bandits: Learning to Rank in the Cascade Model. In *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*. 767–776.

[13] Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvari. 2015. Combinatorial Cascading Bandits. In *Advances in Neural Information Processing Systems 28 (NIPS'15)*. 1450–1458.

[14] Paul Lagrée, Claire Vernade, and Olivier Cappe. 2016. Multiple-Play Bandits in the Position-based Model. In *Advances in Neural Information Processing Systems 29 (NIPS'16)*. 1597–1605.

[15] John Langford and Tong Zhang. 2008. The Epoch-greedy Algorithm for Multi-Armed Bandits with Side Information. In *Advances in Neural Information Processing Systems 21 (NIPS'08)*. 817–824.

[16] Ping Li, Qiang Wu, and Christopher J Burges. 2008. Mcrank: Learning to Rank Using Multiple Classification and Gradient Boosting. In *Advances in Neural Information Processing Systems 21 (NIPS'08)*. 897–904.

[17] Shuai Li, Baoxiang Wang, Shengyu Zhang, and Wei Chen. 2016. Contextual Combinatorial Cascading Bandits. In *Proceedings of the 31st International Conference on Machine Learning (ICML'16)*. 1245–1253.

[18] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous Control with Deep Reinforcement Learning. *arXiv preprint arXiv:1509.02971* (2015).

[19] Tie-Yan Liu et al. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[20] Hamid R. Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S. Sutton. 2010. Toward Off-Policy Learning Control with Function Approximation. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*. 719–726.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-Level Control Through Deep Reinforcement Learning. *Nature* 518, 7540 (2015), 529–533.

[22] Ramesh Nallapati. 2004. Discriminative Models for Information Retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'04)*. ACM, 64–71.

[23] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning Diverse Rankings with Multi-Armed Bandits. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*. ACM, 784–791.

[24] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*. 1889–1897.

[25] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529, 7587 (2016), 484–489.

[26] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic Policy Gradient Algorithms. In *Proceedings of the 31st International Conference on Machine Learning (ICML'14)*. 387–395.

[27] Aleksandrs Slivkins, Filip Radlinski, and Sreenivas Gollapudi. 2013. Ranked Bandits in Metric Spaces: Learning Diverse Rankings Over Large Document Collections. *Journal of Machine Learning Research* 14, Feb (2013), 399–436.

[28] R.S. Sutton and A.G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

[29] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems 13 (NIPS'00)*. 1057–1063.

[30] C.J.C.H. Watkins. 1989. *Learning From Delayed Rewards*. Ph.D. Dissertation. King's College, Cambridge.

[31] Ronald J Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning* 8, 3-4 (1992), 229–256.

[32] Yisong Yue and Thorsten Joachims. 2009. Interactively Optimizing Information Retrieval Systems as a Dueling Bandits Problem. In *Proceedings of the 26th International Conference on Machine Learning (ICML'09)*. ACM, 1201–1208.

[33] Masrour Zoghi, Tomas Tunys, Mohammad Ghavamzadeh, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. 2017. Online Learning to Rank in Stochastic Click Models. In *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*. 4199–4208.

[34] Shi Zong, Hao Ni, Kenny Sung, Nan Rosemary Ke, Zheng Wen, and Branislav Kveton. 2016. Cascading Bandits for Large-Scale Recommendation Problems. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence (UAI'16)*. 835–844.