Thompson Sampling for Contextual Combinatorial Bandits

Yingfei Wang
Department of Computer Science, Princeton University
Princeton, NJ 08540, USA
yingfei@cs.princeton.edu

Hua Ouyang
Yahoo Labs
701 First Avenue, Sunnyvale, CA 94089, USA
houyang@yahoo-inc.com

Chu Wang
Program in Applied and Computational Mathematics, Princeton University
Princeton, NJ 08540, USA
chuw@math.princeton.edu

Jianhui Chen
Yahoo Labs
701 First Avenue, Sunnyvale, CA 94089, USA
jianhui@yahoo-inc.com

Tsvetan Asamov
Department of Operations Research and Financial Engineering, Princeton University
Princeton, NJ 08540, USA
tasamov@princeton.edu

Yi Chang
Yahoo Labs
701 First Avenue, Sunnyvale, CA 94089, USA
yichang@yahoo-inc.com

ABSTRACT

Multi-Armed Bandit (MAB) framework has been successfully applied in many web applications, where the exploration-exploitation trade-off can be naturally taken care of. However, many complex real-world applications that involve multiple content recommendations cannot fit into the traditional MAB setting. To address this issue, we consider a combinatorial bandit problem where the learner selects S actions from a base set of K actions, and displays the results in S (out of M) different positions. The aim is to maximize the cumulative reward or equivalently minimize the regret with respect to the best possible subset and positions in hindsight. By the adaptation of network flow formulations, a practical algorithm based on Thompson sampling is derived for the (contextual) combinatorial problem, thus resolving the problem of computational intractability. The network flow formulations can also be incorporated in ϵ-greedy and pure exploitation algorithms to provide an exact single-step solution without enumerating on a gigantic number of possible combinatorial subsets. With its potential to work with any probabilistic models, to illustrate the effectiveness of our method, we focus on context-free Gaussian process optimization and a contextual setting where click-through-rate is predicted by logistic regression. We demonstrate the algorithms' performance on synthetic Gaussian process problems and on large-scale news article recommendation datasets from Yahoo! Front Page Today Module.

CCS Concepts

•Information systems → Content ranking; Personalization; Sponsored search advertising; •Computing methodologies → Online learning settings;

Keywords

Thompson sampling; contextual combinatorial bandits; Gaussian processes optimization; recommender systems

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

1. INTRODUCTION

The Multi-Armed Bandit (MAB) problem is a classic and natural framework for many machine learning applications. In this setting, the learner takes an action and only observes partial feedback from the environment. MAB naturally addresses the fundamental trade-off between exploration and exploitation [4, 17]. The learner must balance the exploitation of actions that performed well in the past and the exploration of less played actions that could potentially yield higher payoffs in the future. Traditional MAB is a sequential decision making setting defined over a set of K actions. At each time step t, the learner selects an action I_t and observes some payoff X_{t,I_t}. In stochastic MAB, the reward of each arm is assumed to be drawn from some unknown probability distribution. The learner then updates its knowledge based on this payoff. The goal is to maximize the commutative payoff obtained in a sequence of n allocations over time, or equivalently minimizing the *regret*:

$$R_n \equiv \max_{i=1,\dots,K} \mathbb{E}\left[\sum_{t=1}^{n} X_{t,i} - \sum_{t=1}^{n} X_{t,I_t}\right].$$

MAB has been extensively studied, and many algorithms are proposed and have found applications in various domains. Some successful web applications are news and movie recommendation, web advertising, vertical search and query autocompletion [18, 7, 9, 14]. Dispite of these advances, many real-world applications cannot fit into the traditional multi-armed bandit framework.

We use news article recommendation as a motivating example. Fig. 1 is a snapshot of a portal website. In this view, there are totally 10 slots where 9 news articles and 1 sponsored ad can be displayed. These slots differ in positions, container sizes, image shapes and font colors. To select 9 articles from a large pool and place them to 9 of 10

slots in the webpage is a combinatorial problem. Moreover, a user can also click more than one articles during a view session, and the goal is to find an optimal layout configuration which maximizes the expected *total* click-through-rate (CTR). There are some work that addresses the bandit with multi-plays, for example, subset regret problems [15, 13, 25], batch-mode bandit optimization with delayed feedbacks [12] and ranked bandits [20]. However, the complex combinatorial setting in our example is beyond the capacity of existing methods. In this paper we consider a novel and general contextual combinatorial bandit formulation to account for any layout information or possible position bias and can be accomplished potentially with any probabilistic user click models.

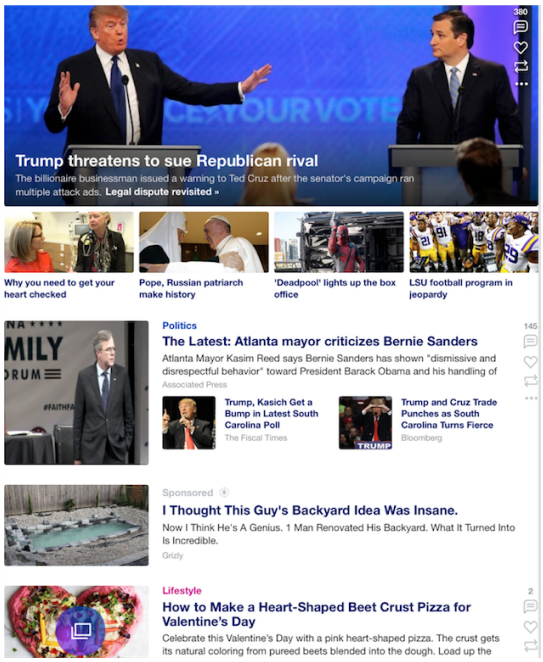


Figure 1: An example of news article recommendation.

To model this complex scenario, we consider the following combinatorial bandit problem. Given the context information, instead of selecting one arm, the learner selects a subset of S actions and displays them on S different positions from M possible positions. If we set $S = 1$, the problem is equivalent to the traditional contextual K-armed bandits. If we set $K = 1$ as a dummy variable and treat N positions as actions, our combinatorial bandit problem degenerates to the unordered subset regrets problems [15, 13]. The bandit ordered slate problem [15] and ranked bandits [20] are also special cases of our setting with $S = M$. Yet our setting is not restricted to learn-to-rank and is general enough to encode any layout information. In our news recommendation example, the slot positions, container shapes, image and font sizes in Fig. 1 can be co-optimized simultaneously.

Various algorithms have been proposed to solve MAB problems [4, 16, 22]. In this work we focus on the Thompson sampling [23] method. The idea of Thompson sampling is to draw each arm according to its probability of being optimal. One advantage of Thompson sampling is that no matter how complex the stochastic reward function (of the action, context and the unknown parameter θ), it is computationally easy and algorithmically straightforward to sample from the posterior distribution and select the action with the highest reward under the sampled parameter. Thus it has the potential to work with any probabilistic user click models. In contrast, the popular UCB policies can not extend easily beyond (generalized) linear models. Another particular advantage of Thompson sampling is that it requires no tuning parameters.

The primary contributions of this paper are the definition of the (contextual) combinatorial bandit problem. With the potential to work with any probabilistic (user click) models, we mainly focus on context-free Gaussian process optimization in the bandit setting and a contextual setting where the features of the content, the user information and positions are used to predict CTR by logistic regression. Due to the Thompson sampling's flexibility on the stochastic reward function and by adaptation of network flow formulations, a practical algorithm is derived to provide exact solution for the (contextual) combinatorial problem without enumerating on a gigantic number of possible combinatorial subsets. The network flow formulations can also be incorporated in ϵ-greedy and pure exploitation algorithms to provide an exact single-step solution in the (contextual) combinatorial bandit problem. We demonstrate the algorithms' performance on synthetic Gaussian process problems and on large-scale news article recommendation dataset from Yahoo! Front Page Today Module.

2. PROBLEM SETTINGS

Due to position and layout bias, it is reasonable to assume that for every article and every position that it can be shown in, there is a click-through-rate associated with the (content, position) pair, which specifies the probability that a user will click on the vertical if it is displayed in that position. In a sequence of rounds $t = 1, 2, ..., T$, the learner is required to choose S actions from a base set $\mathcal{A}$ of K actions to display in S (out of M) different positions, and receives a reward that is the sum of the rewards of (action, position) pair in the chosen subset. All the payoff for each displayed (content position) pair is revealed.

(Contextual) Combinatorial Bandits. In each round t, the learner is presented with a (optional) context vector x_t. In order to take layout information into consideration, a feature vector $a_{k,m}$ is constructed for each (action, position) pair (k, m), such that $a_k \in \mathcal{A}$, and $m \in \{1, 2, ..., M\}$. The learner chooses S actions from $\mathcal{A}$ to display in S (out of M) different positions. Thus a valid combinatorial subset is a mapping from S different actions to S different positions; or more concisely, it is a one-to-one mapping $\pi_t : \{1, 2, ..., S\} \mapsto \big(\mathcal{A}, \{1, 2, ..., M\}\big)$. The learner then receives reward $r_{\pi_t(s)}(t)$ for each chosen (action, position) pair. The total reward of round t is the sum of the rewards of each position $\sum_{s=1}^{S} r_{\pi_t(s)}(t)$. The goal is to maximize the expected cumulative rewards over time $\mathbb{E}\left[\sum_{t=1}^{T}\sum_{s=1}^{S} r_{\pi_t(s)}(t)\right]$.

An important special case of the contextual combinatorial bandits is the context-free setting in which the context x_t remains constant for all t. By setting S, K to special values, many existing literature, including (contextual) K-armed bandits, bandit slate problems [15] and subset regrets problems [13], can be seen as special cases of our combinatorial bandits setting.

In the example of sponsored search, we may view ads in the pool as actions. For the t-th user visit, the context vector x_t summarizes information including the user and the page view. The action feature vector $a_{k,n}$ includes features of the candidate ad and identifiers of the positions. S out of K ads are chosen to be displayed on K out of M positions. The When each of the displayed ads is clicked on, a reward of 1 is acquired; otherwise, the reward is 0. The goal is to maximize the expected number of clicks from users.

3. THOMPSON SAMPLING

In (contextual) K-armed bandit problems, at each round a (optional) context information x is provided for the learner. The learner then chooses an action $a \in \mathcal{A}$ and observes a reward r. The goal is to find a policy that maximizes the expected cumulative reward of the context sequence.

Thompson Sampling [23, 7] for the contextual bandit problems is best understood in a Bayesian setting as follows. Each past observation consists of a triplet (x_i, a_i, r_i) and the likelihood function of the reward is modeled in the parametric form $\Pr(r|a, x, \theta)$ over some parameter set Θ. Given some known prior distribution over Θ, the posterior distribution of these parameters is given by the Bayes rule based on the past observations. At each time step t, the learner draws $\hat{\theta}^t$ from the posterior and chooses the action that has the largest expected reward based on the sampled $\hat{\theta}^t$, as described in Algorithm 1.

Algorithm 1: Thompson sampling

input : Prior Distribution $P(\theta)$ and observation history $\mathcal{D}^0 = \emptyset$

for $t = 1$ *to* T **do**

Receive context x_t

1. Sample from posterior:
Draw θ^t from $P(\theta|\mathcal{D}^{t-1})$

2. Select action:
Draw arm $a_t = \arg\max_a \mathbb{E}[r|a, x_t, \hat{\theta}^t]$
Observe reward r_t

3. Update posterior:
$\mathcal{D}^t = \mathcal{D}^{t-1} \cup \{x_t, a_t, r_t\}$
Update posterior distribution $P(\theta|\mathcal{D}^t)$ according to Bayes rule

end

It can be seen from the algorithm that in Step 1, the posterior sampling has nothing to do with the format of the stochastic reward function r and the function is only evaluated at sampled $\hat{\theta}^t$ in Step 2. This makes Thompson sampling computationally applicable for complex reward function as long as a probabilistic modeling for the unknown rewards can be obtained.

4. ALGORITHMS FOR THE CONTEXTUAL COMBINATORIAL BANDIT PROBLEMS

Our main algorithmic idea is to use the Thompson sampling algorithm for contextual bandits due to its flexibility for complex reward functions.

On each round t, the contextual combinatorial bandit problem involves choosing S actions from a set $\mathcal{A}$ of K actions to display in S (out of M) different positions, and receives a reward that is the sum of the chosen subset. A naive approach is to treat each complex combination as a super arm and apply a traditional bandit algorithm which involves enumerating the values on all the super arms at each time step. This approach encounters practical and computational issues since the number of super arms is $\binom{M}{S}\binom{K}{S}S!$ which is gigantic (e.g., $\approx 10^8$ even for $K = 100, M = 10$ and $S = 3$).

Suppose the likelihood function of the reward of each context x and (action, position) pair $a_{k,m}$ is modeled in the parametric form $\Pr(r|x, a, \theta)$ over some parameter set Θ. Therefore we need a special variant of Thompson sampling which efficiently finds the optimal mapping $\pi_t^* : \{1, 2, ..., S\} \mapsto \big(\mathcal{A}, \{1, 2, ..., M\}\big)$ so that

$$\pi_t^* \in \arg\max_{\pi_t} \sum_{s=1}^{S} \mathbb{E}[r|a_{\pi_t(s)}, x_t, \hat{\theta}^t]. \tag{1}$$

The rest of this section is organized as follows. In Section 4.1, we formally consider the optimization problem (1). In order to efficiently solve (1), we translate it into a minimum-cost maximum-flow problem in Section 4.2. Based on the network flow formulation, we derive the practical Thompson sampling methods for the combinatorial bandits. With the potential of woking with any probabilistic models, we demonstrate our approach with Gaussian processes and Bayesian logistic regression in Section 4.3 and Section 4.4.

4.1 Action selection as a constrained optimization problem

In order to apply the approach described above, we need a way to compactly represent the maximization over the super arms π_t as in Eq. (1) without enumeration. we first denote the expected reward of each (action, position) pair $\mathbb{E}[r|a_{k,m}, x_t, \hat{\theta}^t]$ for display action a_k at position p_m given context x_t and sampled parameter $\hat{\boldsymbol{w}}^t$ as $e_{k,m}$ to simplify the notation. We also define indicator variable $f_{k,m}$ to stand for whether to display action a_k at position p_m or not, $f_{k,m} \in \{0, 1\}$. We next translate a valid super arm into mathematical constraints. First, since each action can be displayed at most once, it corresponds to the constraint $\sum_m f_{k,m} \leq 1, \forall k$. Second, no two actions can be placed in the same position and thus we have $\sum_k f_{k,m} \leq 1$, $\forall m = 1, ..., M$. Finally, there should be exactly S actions chosen, which is equivalent to $\sum_k \sum_m f_{k,m} = S$. Since the expected reward of each super arm is the sum of the reward of each position, it can be written as $\sum_k \sum_m f_{k,m} e_{k,m}$. Now we are ready to represent the maximization over the super arms as the following integer programming:

$$\begin{aligned}
\max_{f} \quad & \sum_{k=1}^{K} \sum_{m=1}^{M} f_{k,m} e_{k,m} \\
\text{subject to} \quad & \\
& \sum_{m=1}^{M} f_{k,m} \leq 1, \ \forall k = 1, \ldots, K \\
& \sum_{k=1}^{K} f_{k,m} \leq 1, \ \forall m = 1, \ldots, M \\
& \sum_{k=1}^{K} \sum_{m=1}^{M} f_{k,m} = S \\
& f_{k,m} \in \{0, 1\}, \ \forall k = 1, \ldots, K, \ m = 1, \ldots, M
\end{aligned} \tag{2}$$

In general, integer programming problems cannot be solved efficiently. However, the given formulation can be interpreted as a network flow problem, a class of problems that adopt polynomial time solutions (and enjoy interesting properties such as the max–flow min–cut duality [24]). We elaborate on our solution approach in section 4.2.

4.2 Network Flow

This section deals with the exact solution for the optimization problem (2). The integer optimization problem (2) can be interpreted as a minimum-cost maximum-flow formulation with edge costs $-e_{k,m}$ as has been depicted in Figure 2. The decision variables $f_{k,m}$ represent the amount of flow to be transfered along the edges of a bipartite graph with expected rewards $e_{k,m}$. In addition, S represents the total size of the network flow. Moreover, the flow capacity of the edges adjacent to the bipartite graph is 1, which implies that those edges can accomodate a flow of at most 1 unit. Furthermore, we can change integer programming formulation of (2) to a linear programming by relaxing the last set of constraints in (3) with their continuous equivalent. Thus we arrive at the following formulation:

$$\begin{aligned}
\min_{f} \quad & \sum_{k=1}^{K}\sum_{m=1}^{M} -e_{k,m} f_{k,m} \\
\text{subject to} \quad & \\
& \sum_{m=1}^{M} f_{k,m} \leq 1,\ \forall k = 1, \ldots, K \\
& \sum_{k=1}^{K} f_{k,m} \leq 1,\ \forall m = 1, \ldots, M \\
& \sum_{k=1}^{K}\sum_{m=1}^{M} f_{k,m} = S \\
& f_{k,m} \in [0,1],\ \forall k = 1, \ldots, K,\ m = 1, \ldots, M
\end{aligned} \tag{3}$$

However, the constraint matrices of such problems feature special properties.

THEOREM 1 ([3]). *The node-arc incidence matrix of a directed network is totally unimodular.*

Hence, we know that the set of constraints in the linear programming relaxation of problem (3) can be represented in standard form as $Ax = b,\ x \geq 0$ with a totally unimodular constraint matrix A. In addition, we also know that the following equivalence relation holds:

THEOREM 2 ([3]). *Let A be an integer matrix with linearly independent rows. Then the following conditions are equivalent:*

- *A is unimodular.*
- *Every basic feasible solution defined by the constraints $Ax = b,\ x \geq 0$ is integer for any integer vector b.*

Since the incidence matrix of a graph has linearly independent rows and S is an integer, we know that the linear programming relaxation (3) of the super arm selection problem will result in an integer optimal solution $f^* \in \{0,1\}^{K \times M}$. Furthermore, linear programming problems can be solved in polynomial time using interior–point methods [19], and therefore we can solve the super arm selection problem efficiently. Please note that in general, specialized algorithms for min–cost network flow problems can have better running times than the linear programming approach. However, such specialized methods usually do not allow for the introduction of addition constraints which can arise in practice, and the development and testing of such methods is beyond the scope of the current paper. For these reasons, we used a linear programming solver in our numerical experiments.

It is worth noting that we do not assume any specific function form of the likelihood function of the reward $\Pr(r|x, a, \theta)$ as long as effective Bayesian inference can be obtained for the unknown quantities. Hence any probabilistic modeling of user clicks, e.g. the Cascade Model [11] and the Examination Hypothesis [21], has the potential to be incorporated in our combinatorial bandit settings. Besides, even though we are motivated by web user click applications, our approach is not restricted to predicting CTRs with 0/1 reward. It is generally applicable to any probabilistic modeling of any reward functions. In the next two sections, we demonstrate our approach in both context-free Gaussian processes optimization and a contextual setting where the CTR is predicted by Bayesian logistic regression.

4.3 Thompson sampling for the combinatorial bandits with Gaussian processes

Other than the web clicking problems, many real-world problems fit easily into our combinatorial bandits formulation. For example, a nice area of application of this logic is dynamic pricing. It can apply to small business that has a lot of room to change its pricing decisions, but it is not magnificent enough to change the underlying unknown revenue curve resulting from the global supply and demand. An interesting example of such business is recharging stations for mobile phones in a develop country. Cell phone use in rural Africa is currently popular since it is harder to stretch a land line out to every home in rural area. An entrepreneur with small power generator such as a car battery or a solar panel can serve many uses. The question is the pricing decision of the African entrepreneur running cell phone recharging stations. For example, the entrepreneur can purchase car batteries or solar panels, change it in town and then travel to different villages to charge phone for a fee that can be any value between \$0.1 and \$0.3. Suppose the entrepreneur has more than one charging devices and can go to different places day by day, the dynamic pricing application can be modeled as the combinatorial bandits. Since not all the combinations of places make sense, probably due to travel distances, the entrepreneur can easily add in other constraints in the linear programming relaxation.

As before, we make the assumption that there is an unknown reward function value $g(\cdot)$ for each (price, position) pair $\mathcal{X}$. At each time, if we choose to measure a point $x \in \mathcal{X}$, we get to see its function value perturbed by i.i.d. Gaussian noise $y_t = g(x) + \epsilon_t,\ \epsilon_t \sim \mathcal{N}(0, \sigma^2)$. Since in dynamic pricing, the revenue of placing the recharging station at nearby positions or the revenue of similar prices are highly correlated, we can enforce implicit properties like smoothness by modeling the unknown function g as a sample of a *Gaussian process* (GP) whose realizations consist of random variables associated with each $x \in \mathcal{X}$. Every finite collection of those random variables has a multivariate normal distribution. Each Gaussian process $\mathrm{GP}(\boldsymbol{\mu}, \boldsymbol{K})$ can be

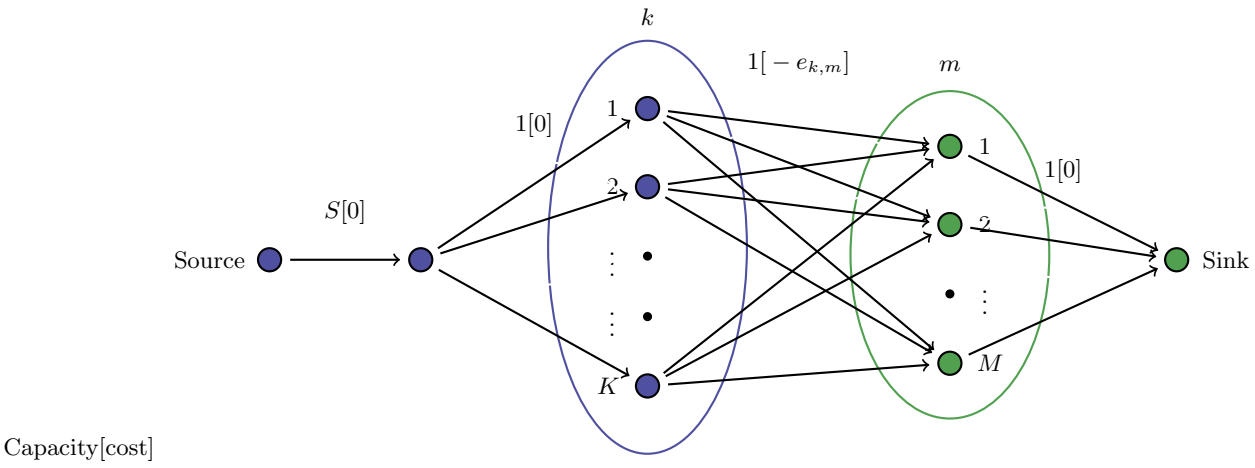


Figure 2: Network flow problem with a maximum capacity of S. The decision variables $f_{k,m}$ represent the amount of flow to be transfered along the edges of a bipartite graph with expected rewards $e_{k,m}$.

specified by its mean function $\boldsymbol{\mu}$ and covariance (or kernel) function $\boldsymbol{K} = [k(x, x')]_{x,x' \in \mathcal{X}}$. We denote the normal distribution $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{K}_0)$ as the prior distribution over g. We introduce the σ-algebra $\mathcal{F}_t$ formed by previous t observations of super arms $\boldsymbol{Y}_t = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_t\}$ at points $X_t = \{\pi_1, ..., \pi_t\}$, where $\boldsymbol{y}_t = [y_{t,1}, ..., y_{t,S}]^T$. We define $\boldsymbol{\mu}_t = \mathbb{E}[g|\mathcal{F}_t]$ and $\boldsymbol{K}_t = \text{Cov}[g|\mathcal{F}_t]$. By the property of Gaussian Processes, conditioning on $\mathcal{F}_t$, $g \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{K}_t)$.

At time t, for any order of the chosen arm $\{x_{i_1}, ..., x_{i_S}\} = \{\pi_t(1), ..., \pi_t(S)\}$, loop over S arms and update the statistics recursively, starting from $\boldsymbol{\mu}_{t+1} \leftarrow \boldsymbol{\mu}_t$, $\boldsymbol{K}_{t+1} \leftarrow \boldsymbol{K}_t$:

$$\begin{aligned} \boldsymbol{\mu}_{t+1} &\leftarrow \boldsymbol{\mu}_{t+1} + \frac{y_{t+1,j} - \mu_{t+1,x_{i_j}}}{\sigma^2 + \sigma^2_{t+1}(x_{i_j})} \boldsymbol{K}_{t+1} \boldsymbol{e}_{x_{i_j}}, \\ \boldsymbol{K}_{t+1} &\leftarrow \boldsymbol{K}_{t+1} - \frac{\boldsymbol{K}_{t+1} \boldsymbol{e}_{x_{i_j}} (\boldsymbol{e}_{x_{i_j}})^T \boldsymbol{K}_{t+1}}{\sigma^2 + \sigma^2_{t+1}(x_{i_j})}, \end{aligned}$$

where $\sigma^2_{t+1}(x) = k_{t+1}(x, x)$ and $\boldsymbol{e}_x$ is a column vector with only the element x one and others zero.

The Thompson sampling algorithm then works as follows. At each round, it first draws random parameter $\hat{\theta}^t_{k,m}$ from the posterior $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{K}_t)$ and then uses the linear programming to solve optimization problem (3) to select the super arm π_t with $e_{k,m} = \hat{\theta}^t_{k,m}$.

4.4 Thompson sampling for the combinatorial bandits with logistic regression

In this section, we consider using Bayesian logistic regression to model the likelihood (CTR) of each (action, position) pair. Exact Bayesian inference for logistic regression is intractable since the evaluation of the posterior distribution comprises a product of logistic sigmoid functions and the integral in the normalization constant is intractable as well. In our model, the posterior distribution on the weights is approximated by a Gaussian distribution with a diagonal covariance matrix. With a Gaussian prior on the weight vector, the Laplace approximation can be obtained by finding the mode of the posterior distribution and then fitting a Gaussian distribution centered at that mode (see Chapter 4.5 of [6]).

In order to continuously refresh the system with new data, the Bayesian logistic regression has been extended to leverage for model updates after each batch of training data [7, 8, 26]. To be more specific, the Laplace approximated posterior serves as a prior on the weights to update the model when a new batch of training data becomes available as described in Algorithm 2. Note that by setting $q_j = \lambda$, an l_2 regularized Logistic regression can be interpreted as a Bayesian model with a Gaussian prior on the weights with standard deviation $1/\sqrt{\lambda}$.

Algorithm 2: Recursive Bayesian Logistic Regression with Batch Updates

input : Regularization parameter $\lambda > 0$
$m_j = 0$, $q_j = \lambda$. (Each weight w_j has an independent prior $\mathcal{N}(m_j, q_j^{-1})$)
for $t = 1$ *to* T **do**
  Get a new batch of training data $(\boldsymbol{x}_i, y_i)$, $i = 1, ..., n$.
  Find $\hat{\boldsymbol{w}}$ as the maximizer of: $-\frac{1}{2}\sum_{j=1}^{d} q_i(w_i - m_i)^2 - \sum_{i=1}^{n} \log(1 + \exp(-y_i \boldsymbol{w}^T \boldsymbol{x}_i))$.
  $m_j = \hat{w}_j$
  $q_j = q_j + \sum_{i=1}^{n} \sigma(\hat{\boldsymbol{w}}^T \boldsymbol{x}_i)(1 - \sigma(\hat{\boldsymbol{w}}^T \boldsymbol{x}_i)) x_{ij}^2$.
end

The training example is made of triplets (x, a, p). Specifically, suppose each context x and action k is represented by feature vectors $\boldsymbol{\phi}_{\boldsymbol{x}}$, and $\boldsymbol{\psi}_{\boldsymbol{a}}$, respectively, to reflect the effect of different physical positions on the page, the extended logistic regression model is $\sigma\big(F(x, a, p)\big)$, where

$$F(x, a, p) = \mu + \boldsymbol{\alpha}^T \boldsymbol{\phi}_{\boldsymbol{x}} + \boldsymbol{\beta}^T \boldsymbol{\psi}_{\boldsymbol{a}} + \sum_{m=1}^{M} \gamma_m \mathbb{I}(p, p_m). \quad (4)$$

At each round, after drawing a random parameter $\hat{\boldsymbol{w}}^t$ from the approximated posterior $\mathcal{N}(m_i, q_i^{-1})$, we use the linear programming developed in Section 4.2 to select the super arm π with $\mathbb{E}[r|a_{k,m}, x_t, \hat{\boldsymbol{w}}^t] = \sigma(\Phi^T \hat{\boldsymbol{w}}^t)$, where $\Phi = [\boldsymbol{\phi}_{x_t}; \boldsymbol{\psi}_{a_k}; \boldsymbol{e}_m]$. It is worth noting here that our model does not require the action set $\mathcal{A}$ to be fixed. This offers great benefit for web applications, in which, for example, the pool

of available news articles for recommendation for each user visit changes over time. The algorithm of Thompson sampling for the combinatorial bandits with Bayesian logistic regression (TSLR) is summarized in Algorithm 3.

Algorithm 3: Thompson sampling for the combinatorial bandits with logistic regression

input : Regularization parameter $\lambda > 0$
$m_j = 0$, $q_j = \lambda$.
for $t = 1$ *to* T **do**
  Receive context x_t
  1. Sample from posterior:
  Draw $\hat{w}_j^t$ from $\mathcal{N}(m_j, q_j^{-1})$
  2. Select action:
  Comput $e_{k,m} = \mathbb{E}[r|a_{k,m}, x_t, \hat{\boldsymbol{w}}^t]$, $\forall k, m$
  Solve the optimization problem (3) and get $[\hat{f}_{k,m}^t]$
  Display the super arm according to $[\hat{f}_{k,m}^t]$
  Observe rewards $r(t)$
  3. Update posterior:
  Update m_j, q_j according to Algorithm 2
end

We close this section by briefly illustrating the flexibility on different choices of user click models. As an example, we consider the extended user click models with content quality features $Q_i(a, p)$ (whether the quality of link above/below is better and the number of links above/below which are better) [5]:

$$F(x, a, p) = \mu + \boldsymbol{\alpha}^T \boldsymbol{\phi}_{\boldsymbol{x}} + \boldsymbol{\beta}^T \boldsymbol{\psi}_{\boldsymbol{a}} + \sum_{m=1}^{M} \gamma_m \mathbb{I}(p, p_m) + \sum_{i=1}^{|Q|} \eta_i Q_i(a, p).$$

As studied in [5], a boosting-style algorithm can be used to learn the parameters of the extended model. In Bayesian settings, after we get the new batch of training data, first we set each $Q_k = 0$ and run one-step recursive Bayesian logistic regression to find m_j and q_j. Next, we update the Q_k value using the updated m_j and q_j. We then re-run recursive Bayesian logistic regression to get new values of m_j and q_j. We iterate this process until the first iteration in which the log-likelihood of the data decreases. We use the final m_j and q_j as the posterior parameter value from which we get the sampled $\hat{\boldsymbol{w}}$ and then solve the optimization problem (3) to select the super arms. That is to say, due to the unique properties of Thompson sampling, the user click model is encapsulated in its own probabilistic updates, resulting in the wide applicability of the proposed algorithm.

5. EXPERIMENTS

In this section, we provide experimental results on both synthesis datasets and Yahoo! Front Page Webscope datasets to demonstrate the ability of our approach to solve the real world problems that can be modeled as our contextual combinatorial bandit problems and that have not been throughly studied before.

5.1 Baseline algorithms

As baseline algorithms to compare against our proposed algorithm in Section 4, we consider the following approaches.

Random. Select S actions and S positions uniformly at random.

Unordered ϵ-Greedy. This approach is a common heuristic which does not take the position bias into consideration. It maintains an estimate of the function value of each action (regardless of positions). For the case of learning-to-rank with $S = M$, the exploitation part selects top-M actions base on current estimations and places them in order. For other cases where $S < M$ or for general layout information which can not translate to a ranked list, this approach does not apply since it is not obvious on how and where to place the actions.

Exploitation. This is an extension of pure exploitation algorithm for traditional bandit problems based on our network flow techniques. It maintains an estimate of the function value of each (action, position) pair. At each time step, it needs to select the super arm with the highest estimation in value. Instead of applying the naive approach of enumerating the values on all the super arms or using any approximation heuristics, it can in fact benefit from the network flow formulation and use linear programming to find the exact solution of Eq. (3).

ϵ-Greedy. Use Exploitation with probability $1 - \epsilon$ and use Random with probability ϵ. The Exploitation part is done by our network flow formulation.

Ranked bandits [20]. This approach works only for learn-to-rank with $S = M$. It runs an multi-armed bandit (MAB) instance MAB_m for each rank m. Each of the M copies of the multi-armed bandit algorithm maintains a value (or index) for every action. When deleting the ranking to display to users, MAB_1 is responsible for select which action is displayed at rank 1. MAB_2 then recommends the action to display at rank 2. If this action is already chosen at higher ranks, the select action is then picked arbitrarily. This procedure is repeated until all M actions are selected. The MAB instance is chosen as the UCB1-Normal algorithm [4] for context-free Gaussian processes optimization. More precisely, UCB1-Normal selects the action for which $\bar{x}_j + \sqrt{\alpha \cdot \frac{v_j - t_j \bar{x}_j^2}{t_j - 1} \cdot \frac{\ln(t-1)}{t_j}}$ is maximized, where $\bar{x}_j$ is the average reward for action j, v_j is the sum of squared rewards and t_j is the number of time action j is played. For other cases where $S < M$ or for general layout information, this approach does not apply due to the same reason.

GP-UCB [22]. This approach only works for Gaussian process optimization. In order to obtain a valid upper confident bound, since the standard deviation of the sum of Gaussian random variables is not equal to the sum of individual standard deviations, a linear programing approach can not be used to select the best super arm with the highest upper confident bound. As a result, we treat each complex combination as a super arm and apply the traditional GP-UCB algorithm which involves enumerating the values on all the super arms at each time step. Since the number of super arms explodes quickly, this approach does not scale beyond toy examples. Besides, it is not applicable with logistic regression since due to the sigmoid function, there is no easy way to compute an upper confident bound even with enumeration. It is also worth mentioning that the frequentist's UCB algorithm is not considered here since it requires play each super arm at once which is computationally intractable.

5.2 Experiments on context-free Gaussian processes optimization

We first consider learning-to-rank experimental settings

with $S = M$. In terms of the true function values for each (action, position) pair, similar to the Examination Hypothesis [21] that lower-ranked places has lower probability to be examined, we assume that the value for each action k is discounted by e^{-d_k} at lowered ranks. Specifically, we use the arithmetic progression $\mu_k = 0.5 - 0.025k, k = 1, ..., K$, as the true function value for each arm k at rank 1. The value of (a_k, p_m) is then $c_{km} = \mu_k e^{-(m-1)d_k}$. To make the learning settings more interesting, we allow d_k to be different for different action k. In the experiments, d_k is randomly generated from $[0.3, 0.8]$ for each k. Different sampling noise levels σ and different choices of S, M, K are used in the experiments. For Gaussian processes, we start with a mean vector of zeros and choose the Squared Exponential Kernel $k(x, x') = \alpha^2 \exp(-\beta_1(k - k')^2 - \beta_2(m - m')^2)$ with $\alpha = 100, \beta_1 = 0.2, \beta_2 = 0.1$ in the experiments.

The experimental results are reported on 100 repetitions of each algorithm. In order to make a fair comparison, in each repetition, all the observations are pre-generated and shared across all the competing algorithms. The only difference is the way each strategy select the actions; all the rest, including the models update, is identical as described in Section 4.3.

For Thompson sampling, we also consider the impact of posterior reshaping. In particular, for normally distributed posterior, decreasing the variance would have the effect of increasing exploitation over exploration. In our simulations, we have tried to reshape the covariance matrix $\boldsymbol{K}_t$ to $\alpha^2 \boldsymbol{K}_t$. This only affects the posterior sampling step and does not change the model updates.

We conduct our experiments with different values of tuning parameters. The first row in Fig. 3 shows the mean average regret of different algorithms at the best value of its tuning parameter across time. We also report the distribution of the regret at $T = 150$ of different algorithms with different parameter values in the second row. On each box, the central red line is the median, the edges of the box are the 25th and 75th percentiles, and outliers are plotted individually. The correspondence between algorithms and boxes is the following:

- Box 1-3: Thompson sampling (TS) with posterior reshaping parameter $\alpha = 0.25, 0.5, 1$.
- Box 4: Exploitation.
- Box 5: Random.
- Box 6-8: Unordered ϵ-greedy (U-ϵ-greedy) with $\epsilon = 0.02, 0.01, 0.005$.
- Box 9-11: ϵ-greedy with $\epsilon = 0.02, 0.01, 0.005$.
- Box 12-14: GP-UCB with $\alpha = 2, 1, 0.5$.
- Box 15-17: RBA with $\alpha = 4, 2, 1$.

It can be seen from the figure that Thompson sampling clearly outperforms others. It not only achieves the lowest regret, but also has small variance. Since in our approach, a linear program is used to provide single-step exact solutions for selecting the super arms, the good performance of Thompson sampling is consistent with other empirical evaluations in existing literature on K-armed bandits in which each super arm can be theoretically treated as one arm. Without explicitly considering the position bias, the unordered ϵ-greedy does not yield good performances. RBA performs well on some datasets while poorly on others. On explanation is that even though we use RBA algorithm for multiple clicks per time, it is designed on only allowing one click on the ranked list.

In terms of posterior reshaping, value of smaller α in general yields lower regret since it is in favor of exploitation over exploration. But the price to pay is higher variance. As for the impact of noise level, the variance of the algorithm grows when increasing the noise level. The unordered ϵ-greedy is the biggest victim and Thompson sampling is the least affected.

In order to understand the behavior of different algorithms when $S < M$, we conduct another set of experiments. The rankings or the Examination Hypothesis is no longer suitable for the case $S < M$. We instead use the 2-dimensional Camelback test function (see the bottom right figure in Fig. 4), to generate true function values for each (action, position) pair. We flip the function values since the standard Camelback function is for minimization. We also experiment with other standard text functions, including Griewank and Branin functions. We get similar results and thus do not include them in the paper. The noise level is set to 0.01.

We experiment with different choices of K and S. Since we are dealing with general layout information that is not a ranked list, RBA and U-ϵ-greedy can not be applied. As for GP-UCB, when $K = 50, 100$ and $M = 8$, enumeration on the super arms are computationally unaffordable. We compare our approach with Random, Exploitation and ϵ-greedy with the results shown in Fig. 4.

The results are consistent with previous findings. The average regret is normalized with respect to the number S of selected actions, i.e.

$$\Big(\max_{\pi} \mathbb{E}\big[\sum_{s=1}^{S} r_{\pi(s)}\big] - \mathbb{E}\big[\sum_{t=1}^{T}\sum_{s=1}^{S} r_{\pi_t(s)}(t)\big]/T \Big)/S.$$

It should be noted that the average regret in case $S = 4$ decreases slightly faster than the $S = 2$ case . At first glance, it is counter-intuitive because $S = 4$ is a more difficult problem. However, since there are twice as many examples in the $S = 4$ case for model updates, the parameter at each time step are estimated much better than the $S = 2$ case. This benefits the arm selection in each algorithm.

5.3 News Article Recommendation

In this section, we consider applying Thompson sampling for contextual combinatorial bandits to personalization of news article recommendation on Yahoo! front page [2, 18, 7]. Our work differs from previous literature in that rather than only recommending one article (e.g. only at the story position) at each user visit (see Fig. 1 for an illustration), we have the ability to recommend more than one articles at different positions. The candidate article pool is dynamic over time with an average size around 20. Suppose the user can click on more than one articles during each view session. The goal is to choose the most relevant articles to users or equivalently, maximize the total number of clicks of the displayed articles. There is a fundamental exploration v.s. exploitation tradeoff: in order to learn the true CTR of each article, it needs to be displayed for long time benefit, leading to a potential drop in the shot-term performance.

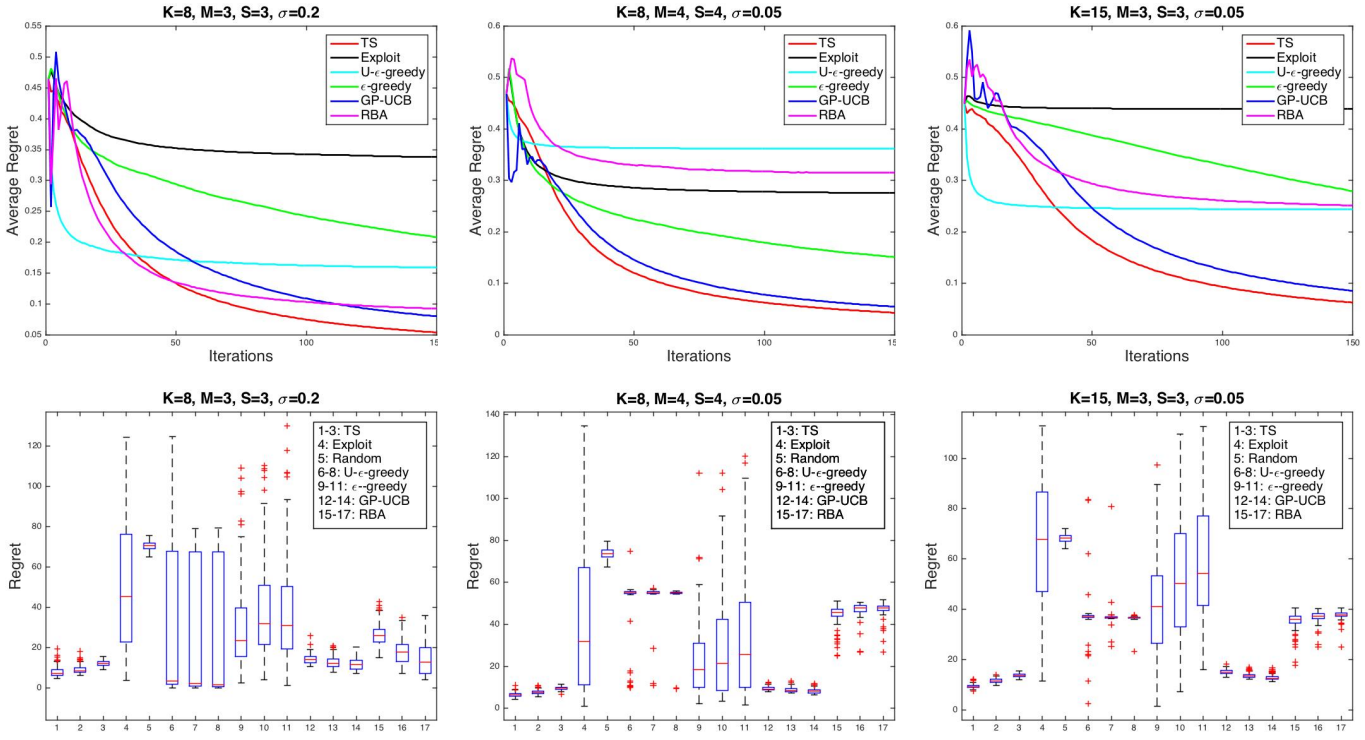


Figure 3: Comparison of performance: Thompson sampling and various heuristics. Top: average regret as a function of T over 100 repetitions. Bottom: distribution of the regret at T = 150.

In Yahoo! Webscope datasets [1], for each visit, both the user and each of the candidate articles are associated with a feature vector of dimension 6 (including a constant feature), constructed by conjoint analysis with a bilinear model [10]. We treat each user feature vector as the context information x and the news articles as actions a, with the feature vectors denoted by $\boldsymbol{\phi}_x$ and $\boldsymbol{\psi}_a$, respectively. We set the number of possible positions $M = 5$. We use the recursive Bayesian logistic regression as in Algorithm 2 based on the user click model (4) to predict article CTRs. Note here due to the combinatorial problems considered in this paper, this logistic regression model is different from that in [7]. Since only one article is displayed upon user visit in [7], it is only affordable in [7] to have a different weight vector for each article.

Evaluating an exploration/exploitation policy is difficult since we do not know the reward of an action that was not chosen at each user visit in the log data. We can not use the unbiased offline evaluation [18] in our case. Since in the combinatorial problems, the number of super arms is gigantic (e.g. $\binom{20}{5} = 15504$), it is rare to have a logged data point that matches the selected super arm. This reduces the effective data size substantially.

Based on the real world context and article features in the Yahoo! Webscope datasets, we instead simulate the true clicks using a weight vector $\boldsymbol{w}^*$. To make the settings more realistic, we first use all the user click data on May 1, 2009 to get a weight vector using logistic regression and then construct $\boldsymbol{w}^*$ by perturbation. Since in our simulated environment, we know the actual CTR of each (action, pair), we use the expected number of clicks as the reward for each user visit. We experiment with different choices of S. For the reasons explained in Section 5.1, we compare our approach with Random, Exploitation and ϵ-greedy.

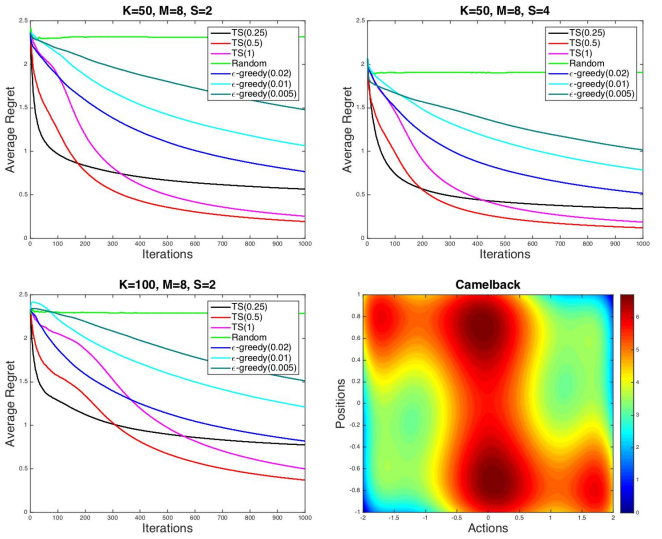
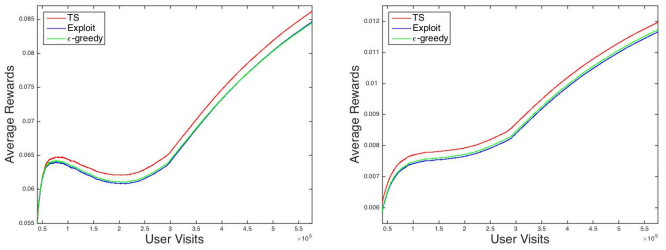


Figure 4: Comparison of performance on Camelback function: Thompson sampling and various heuristics.

Since in a real world system, it is infeasible to update the model after each user feedback, we model this behavior by refreshing the system after every 10 minutes. We report the average reward that is normalized with respect to the

number of selected actions in Fig. 5 and Table 5.3.



(a) $S = 3$ (b) $S = 5$

Figure 5: Average rewards of each competing algorithms on Yahoo! Webscope datasets.

In Fig. 5, we drop the first 40000 user visits, since with the 10 minutes update delay, the system has no knowledge about the CTRs at the very beginning, leading to poor performance of every algorithm.

Thompson sampling yields the best result. Exploit and ϵ-greedy have similar performance. It is consistent with the previous findings that the change in context induces some exploration [7]. Thompson sampling with $\alpha = 0.5$ achieves slightly better performance than the one without posterior reshaping.

When comparing $S = 3$ and $S = 5$, we find that there is some slowdown in Fig. 5(b) or even a decrease in reward in Fig. 5(a). One explanation is that with only a small number of system refresh, the estimation on the parameters $\boldsymbol{w}$ is not stable. Yet since $S = 5$ has more training data in each update than $S = 3$, it yields a smoother reward curve. Due to the same reason, $S = 5$ achieves a higher average CTR at the end of the experiments.

Method	TS (0.5)	TS(1)	Exploit
Reward ($S = 3$)	0.0862	0.0861	0.0846
Reward ($S = 5$)	0.0120	0.0118	0.0117
Method	Random	ϵ-greedy (0.02)	ϵ-greedy (0.01)
Reward ($S = 3$)	0.0538	0.0849	0.0845
Reward ($S = 5$)	0.0072	0.0117	0.0117

Table 1: Average rewards on Yahoo! Webscope datasets.

We emphasize here that the main goal of the experiments is not to claim the superiority of Thompson sampling, but to demonstrate the ability of our approach to solve the real world problems that can be modeled as our contextual combinatorial bandit problems.

6. CONCLUSION

In this paper, we extend the traditional multi-armed bandit problems to a more general contextual combinatorial setting. This is motivated by many web applications where instead of choosing only one action at a time, one can choose S actions and place them on S out of M positions. Naive enumeration on possible combinations is computationally unaffordable. By the adaptation of network flow formulations, a practical algorithm based on Thompson sampling is derived for the (contextual) combinatorial problem. The network flow formulations can also be incorporated in ϵ-greedy and pure exploitation algorithms to provide an exact single-step solution. Due to the unique properties of Thompson sampling, the system update is encapsulated in the chosen probabilistic models. This provides easy incorporation of any probabilistic (user click) models in our proposed framework. We demonstrate the algorithms' performance on synthetic Gaussian process problems and on news article recommendation dataset from Yahoo! Front Page Today Module to illustrate the ability of our approach to solve the real world problems that can be modeled as our contextual combinatorial bandit problems and that have not been throughly studied before.

7. REFERENCES

[1] *Yahoo! Webscope dataset ydata-frontpage-todaymodule-clicks-v1-0*. http://labs.yahoo.com/Academic_Relations.

[2] D. Agarwal, B.-C. Chen, P. Elango, N. Motgi, S.-T. Park, R. Ramakrishnan, S. Roy, and J. Zachariah. Online models for content optimization. In *Advances in Neural Information Processing Systems*, pages 17–24, 2009.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows*. 1993.

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

[5] H. Becker, C. Meek, and D. M. Chickering. Modeling contextual factors of click rates. In *AAAI*, volume 7, pages 1310–1315, 2007.

[6] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[7] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

[8] O. Chapelle, E. Manavoglu, and R. Rosales. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):61, 2014.

[9] W. Chu, L. Li, L. Reyzin, and R. E. Schapire. Contextual bandits with linear payoff functions. In *International Conference on Artificial Intelligence and Statistics*, pages 208–214, 2011.

[10] W. Chu, S.-T. Park, T. Beaupre, N. Motgi, A. Phadke, S. Chakraborty, and J. Zachariah. A case study of behavior-driven conjoint analysis on yahoo!: Front page today module. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1097–1104. ACM, 2009.

[11] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 87–94. ACM, 2008.

[12] T. Desautels, A. Krause, and J. W. Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *The Journal of Machine Learning Research*, 15(1):3873–3923, 2014.

[13] A. Gopalan, S. Mannor, and Y. Mansour. Thompson sampling for complex online problems. In *Proceedings*

of The 31st International Conference on Machine Learning, pages 100–108, 2014.

[14] L. Jie, S. Lamkhede, R. Sapra, E. Hsu, H. Song, and Y. Chang. A unified search federation system based on online user feedback. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1195–1203. ACM, 2013.

[15] S. Kale, L. Reyzin, and R. E. Schapire. Non-stochastic bandit slate problems. In *Advances in Neural Information Processing Systems*, pages 1054–1062, 2010.

[16] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[17] J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pages 817–824, 2008.

[18] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 297–306. ACM, 2011.

[19] Y. Nesterov, A. Nemirovskii, and Y. Ye. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994.

[20] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*, pages 784–791. ACM, 2008.

[21] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*, pages 521–530. ACM, 2007.

[22] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[23] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

[24] V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[25] Y. Wang, K. G. Reyes, K. A. Brown, C. A. Mirkin, and W. B. Powell. Nested-batch-mode learning and stochastic optimization with an application to sequential multistage testing in materials science. *SIAM Journal on Scientific Computing*, 37(3):B361–B381, 2015.

[26] Y. Wang, C. Wang, and W. Powell. The knowledge gradient with logistic belief models for binary classification. *arXiv preprint arXiv:1510.02354*, 2015.