Efficient Ordered Combinatorial Semi-Bandits for Whole-page Recommendation

Yingfei Wang[1], **Hua Ouyang**[2], **Chu Wang**[3], **Jianhui Chen**[4], **Tsvetan Asamov**[5], **Yi Chang**[6]
[1]Department of Computer Science, Princeton University, yingfei@cs.princeton.edu
[2]Apple Inc., hua_ouyang@apple.com
[3]Nokia Bell Labs, chu.wang@nokia-bell-labs.com
[4]Yahoo Research, jianhui@yahoo-inc.com
[5]Department of Operations Research and Financial Engineering, Princeton University, tasamov@princeton.edu
[6]Huawei Research America, yichang@acm.org

Abstract

Multi-Armed Bandit (MAB) framework has been successfully applied in many web applications. However, many complex real-world applications that involve multiple content recommendations cannot fit into the traditional MAB setting. To address this issue, we consider an ordered combinatorial semi-bandit problem where the learner recommends 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 with respect to the best possible subset and positions in hindsight. By the adaptation of a minimum-cost maximum-flow network, a practical algorithm based on Thompson sampling is derived for the (contextual) combinatorial problem, thus resolving the problem of computational intractability. With its potential to work with whole-page recommendation and any probabilistic models, to illustrate the effectiveness of our method, we focus on Gaussian process optimization and a contextual setting where click-through-rate is predicted using 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.

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 (Auer, Cesa-Bianchi, and Fischer 2002; Langford and Zhang 2008). Traditional MAB is a sequential decision making setting defined over a set of K actions. At each time step t, the learner selects a single 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 goal is to maximize the cumulative payoff obtained in a sequence of n allocations over time, or equivalently minimize the *regret*.

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 (Li et al. 2011; Chapelle and Li 2011; Chu et al. 2011; Jie et al. 2013). Despite 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 6 slots to display news articles. These slots differ in positions, container sizes and visual appearance. Several studies (Liu et al. 2015; Lagun and Agichtein 2014) indicate that users do not sequentially scan the webpages. How to make whole-page recommendations, that is, select 6 articles from a larger pool and place them accordingly in the webpage, is a combinatorial problem that is beyond ranking. The goal is to find an optimal layout configuration to maximize the expected total click-through-rate (CTR). A related work on this topic is the optimal page presentation (Wang et al. 2016). Yet even though search engine works in a real time fashion, their models are trained on batch data rather than in an online learning setting, thus neglecting the exploration/exploitation tradeoff.

There are some existing work that addresses the bandits with multi-plays, for example, subset regret problems (Kale, Reyzin, and Schapire 2010; Gopalan, Mannor, and Mansour 2014; Wang et al. 2015; Gai, Krishnamachari, and Liu 2011; Swaminathan et al. 2016), batch-mode bandit optimization with delayed feedbacks (Desautels, Krause, and Burdick 2014) and ranked bandits (Radlinski, Kleinberg, and Joachims 2008). This class of learning problems was also recently formulated as a combinatorial bandit/semi-bandit (Gai, Krishnamachari, and Jain 2012; Audibert, Bubeck, and Lugosi 2013; Wen et al. 2015; Krishnamurthy, Agarwal, and Dudik 2016). However, the complex combinatorial setting in our example is beyond the capacity of existing methods.

To model this scenario, we consider the following ordered combinatorial bandit problem. Given optional 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. Our novelty lies in:

1. Our method does not resort to an oracle to provide approximation solutions. Instead, we formulate the problem via the minimum-cost maximum-flow network, and efficiently provide exact solutions.
2. To the best of our knowledge, our model is the first to

deal with general layout information where the number of positions can be larger than the subset of arms selected, i.e. $S < M$.

3. We use Thompson sampling as the bandit instance. One advantage of Thompson sampling is that no matter how complex the stochastic reward function, it is computationally easy 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, e.g. the Cascade Model and the Examination Hypothesis.

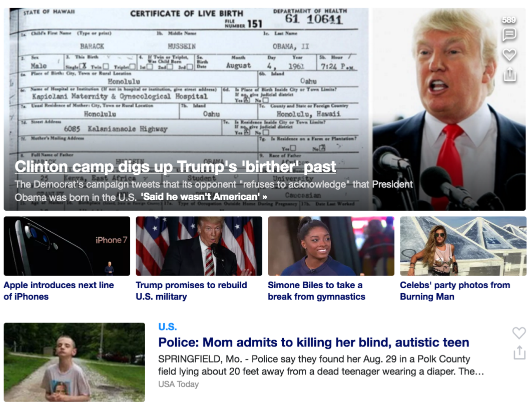

Figure 1: An example of news article recommendation.

Problem Settings

Due to position and layout bias, it is reasonable to assume that for every article and every position, there is a click-through-rate associated with the (content, position) pair, which specifies the probability that a user will click on the content if it is displayed in a certain 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. This feedback model is known as semi-bandits (Audibert, Bubeck, and Lugosi 2013). Since in this work, we explicitly model the positions of the subset of selected arms, we call this setting *ordered combinatorial semi-bandits*.

Ordered (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 $\pi_t : \{1, 2, ..., S\} \mapsto \big(\mathcal{A}, \{1, 2, ..., M\}\big)$. We further refer to each π_t as a super arm. 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 methods can be seen as special cases of our combinatorial bandits setting. For example, $S = 1$ 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 (Kale, Reyzin, and Schapire 2010; Gopalan, Mannor, and Mansour 2014). The bandit ordered slate problem (Kale, Reyzin, and Schapire 2010) and ranked bandits (Radlinski, Kleinberg, and Joachims 2008) 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 optimize whole-page presentation.

Thompson Sampling

In (contextual) K-armed bandit problems, at each round an optional context information x is provided. The learner then chooses an action $a \in \mathcal{A}$ and observes a reward r. Thompson Sampling (Thompson 1933; Chapelle and Li 2011) 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 history $\mathcal{D}^0 = \emptyset$
for $t = 1$ *to* T **do**
  Receive context x_t
  1. Draw θ^t from $P(\theta|\mathcal{D}^{t-1})$
  2. Draw arm $a_t = \arg\max_a \mathbb{E}[r|a, x_t, \hat{\theta}^t]$
   Observe reward r_t
  3. $\mathcal{D}^t = \mathcal{D}^{t-1} \cup \{x_t, a_t, r_t\}$
   Update posterior distribution $P(\theta|\mathcal{D}^t)$
end

Algorithms for the Ordered Combinatorial Semi-Bandits

Our main algorithmic idea is to use Thompson sampling for ordered semi-bandits due to its flexibility for complex reward functions.

On each round t, the ordered combinatorial semi-bandit problem involves choosing S actions from a set $\mathcal{A}$ of K

actions to display in S (out of M) different positions, and receiving 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. This approach may have practical and computational limitations since the number of super arms blows up very quickly.

Suppose the likelihood function of the reward of each context x and (action, position) pair $a_{k,m}$ is modeled in the parametric form $\Pr(r|x, a, \theta)$. The next three sections develop special variants of Thompson sampling which efficiently find the optimal mapping $\pi_t^* : \{1, 2, ..., S\} \mapsto \big(\mathcal{A}, \{1, 2, ..., M\}\big)$ such that

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

Action selection as a constrained optimization

In order to find the best super arm 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 displaying action a_k at position p_m, given context x_t and sampled parameter $\hat{\theta}^t$, as $e_{k,m}$ to simplify notations. We also define indicator variable $f_{k,m}$ to denote whether action a_k is displayed at position p_m, $f_{k,m} \in \{0, 1\}$. 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 $\sum_k \sum_m f_{k,m} = S$. The maximization over the super arms in Eq. (1) can thus be represented as the following integer programming:

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

In general, integer programming problems cannot be solved efficiently. However, as shown in the next section, the given formulation can be interpreted as a network flow that admits polynomial time solutions [and enjoys interesting properties such as the max–flow min–cut duality (Vazirani 2013)].

Network flow

The integer optimization problem (2) can be interpreted as a minimum-cost maximum-flow formulation with edge costs $-e_{k,m}$ as depicted in Figure 2. The decision variables $f_{k,m}$

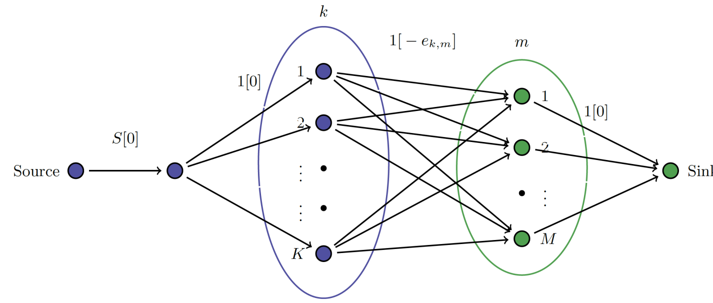

Figure 2: Capacity[cost]. Network flow problem with a maximum capacity of S.

represent the amount of flow to be transferred 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 accommodate 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 with their continuous equivalent $f_{k,m} \in [0, 1]$. The constraint matrices of such problems feature special properties:

Theorem 1 *(Ahuja, Magnanti, and Orlin 1993) 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 (2) can be represented in standard form as $Ax = b$, $x \geq 0$ with a totally unimodular constraint matrix A. Since the incidence matrix of a graph has linearly independent rows and S is an integer, we know that the linear programming relaxation (2) of the super arm selection problem will result in an integer optimal solution $f^* \in \{0, 1\}^{K \times M}$ (Ahuja, Magnanti, and Orlin 1993). Furthermore, linear programming problems can be solved in polynomial time using interior–point methods (Nesterov, Nemirovskii, and Ye 1994), 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 use a linear programming solver in our numerical experiments.

Thompson sampling for the combinatorial semi-bandits with Gaussian processes

As before, we make the assumption that there is an unknown reward function value $g(\cdot)$ for each (price, position) pair. In this section, we consider the cases where observations occur in a continuous domain. At each time, if we choose to measure a point $a_{k,m} \in (\mathcal{A}, M)$, we get to see its function value perturbed by i.i.d. Gaussian noises

$y_t = g(a_{k,m}) + \epsilon_t$, $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$. 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 $a_{k,m}$. Each Gaussian process GP$(\boldsymbol{\mu}, \boldsymbol{K})$ can be specified by its mean function $\boldsymbol{\mu}$ and covariance (or kernel) function $\boldsymbol{K} = [k(a_{k,m}, a_{k',m'})]_{a_{k,m}, a_{k',m'} \in (\mathcal{A}, M)}$. 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 $\mathcal{F}_t$, we have $g \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{K}_t)$.

At time t, for any order of the chosen arm $\{a_{i_1}, ..., a_{i_S}\} = \{\pi_t(1), ..., \pi_t(S)\}$, we loop over S arms and recursively update the statistics, 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.

At each round, Thompson sampling first draws a 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 (2) to select the super arm π_t with $e_{k,m} = \hat{\theta}^t_{k,m}$.

Thompson sampling for the combinatorial bandits with logistic regression

In web applications, observations are usually binary values which cannot be properly modeled by a continuous Gaussian process. To this end, we instead use Bayesian logistic regression to model the likelihood (CTR) depending on both the position and the action itself. Exact Bayesian inference is intractable since the evaluation of the posterior comprises a product of logistic link functions. In our model, the posterior on the weights is approximated by a Gaussian distribution with a diagonal covariance matrix. With a Gaussian prior on the weights $w_j \sim \mathcal{N}(m_j, q_j^{-1})$, the Laplace approximation can be obtained by first finding the mode of the posterior distribution and then fitting a Gaussian distribution centered at that mode (see Chapter 4.5 of (Bishop 2006)).

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 (Chapelle and Li 2011; Wang, Wang, and Powell 2016), where the Laplace approximated posterior serves as a prior on the weights to update the model when a new batch of training data becomes available. To be more specific, given a new batch of training data $(\boldsymbol{x}_i, y_i)$, based on the current prior distribution $w_j \sim \mathcal{N}(m_j, q_j^{-1})$, we first find the MAP (maximum a posteriori) estimation

$$\hat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}} -\frac{1}{2} \sum_{j=1}^{d} q_i (w_i - m_i)^2 - \sum_{i=1}^{n} \log(1 + \exp(-y_i \boldsymbol{w}^T \boldsymbol{x}_i)). \tag{3}$$

Then, by finding the Hessian evaluated at $\hat{\boldsymbol{w}}$, we can compute the inverse variance of each weight w_j as:

$$q_j = q_j + \sum_{i=1}^{n} \sigma(\hat{\boldsymbol{w}}^T \boldsymbol{x}_i)(1 - \sigma(\hat{\boldsymbol{w}}^T \boldsymbol{x}_i) x^2_{ij}. \tag{4}$$

Hence the approximated posterior is $w_j \sim N(\hat{w}_j, q_j^{-1})$.

The past training example is made of (x, a, p, r) with x as the context, a as the action, p as the position and r as a binary reward. Suppose each context x and action a is represented by feature vectors $\boldsymbol{\phi_x}$, and $\boldsymbol{\psi_a}$, respectively. To reflect the effect of different physical positions on the page, the click-through probability is modeled as $\Pr(r = 1|x, a, p) = \sigma\big(F(x, a, p)\big)$, where

$$F(x, a, p) = \mu + \boldsymbol{\alpha}^T \boldsymbol{\phi_x} + \boldsymbol{\beta}^T \boldsymbol{\psi_a} + \sum_{m=1}^{M} \gamma_m \mathbb{I}(p, p_m), \tag{5}$$

$\sigma(z) = \frac{e^z}{1+e^z}$ is the logistic link function and $\mathbb{I}(x, y)$ is the indicator function that is one if $x = y$ and zero otherwise.

We use $\boldsymbol{w}$ to denote the unknown parameter set $\boldsymbol{w} = [\mu; \boldsymbol{\alpha}; \boldsymbol{\beta}; \boldsymbol{\gamma}]$. At each round, we first draw a random parameter $\hat{\boldsymbol{w}}^t$ from the approximated posterior $\mathcal{N}(m_i, q_i^{-1})$. Since reward r is Bernoulli distributed with $\Pr(r = 1|x, a, p) = \sigma\big(F(x, a, p)\big)$, we have $\mathbb{E}[r|a_{k,m}, x_t, \hat{\boldsymbol{w}}^t] = \sigma(\Phi^T \hat{\boldsymbol{w}}^t)$, where $\Phi = [1; \boldsymbol{\phi}_{x_t}; \boldsymbol{\psi}_{a_k}; \boldsymbol{e}_m]$ with e_m as a column vector with only the mth element one and zeros otherwise. We then use the linear programming to select the super arm π that maximizes the reward function (1) with $\mathbb{E}[r|a_{k,m}, x_t, \hat{\boldsymbol{w}}^t] = \sigma(\Phi^T \hat{\boldsymbol{w}}^t)$. 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 each user visit changes over time. The algorithm of Thompson sampling for the combinatorial semi-bandits with Bayesian logistic regression is summarized in Algorithm 2.

Algorithm 2: 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. Draw $\hat{w}^t_j$ from $\mathcal{N}(m_j, q_j^{-1})$
  2. Comput $e_{k,m} = \mathbb{E}[r|a_{k,m}, x_t, \hat{\boldsymbol{w}}^t]$, $\forall k, m$
    Solve the optimization problem (2) and get $[\hat{f}^t_{k,m}]$
    Display the super arm according to $[\hat{f}^t_{k,m}]$
    Observe rewards $r(t)$
  3. Update m_j, q_j according to Algorithm Eq. (3)(4)
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) (Becker, Meek, and Chickering 2007):

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

Intuitively, if the action of interest is placed below a good action, the click-through rate will be lower. Therefore the quality feature functions Q_i depends on the values of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. A boosting-style algorithm can be used to learn the parameters. In Bayesian settings, after we get a new batch of training data, we first set each $Q_i = 0$ and use Eq. (3)(4) to find m_j and q_j. Next, we update the Q_i value using the updated m_j and q_j. We then use updated Q_i values to calculate Eq. (3)(4) 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 (2) 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. Hence any probabilistic modeling of user clicks, e.g. the Cascade Model (Craswell et al. 2008) and the Examination Hypothesis (Richardson, Dominowska, and Ragno 2007), has the potential to be incorporated in our ordered combinatorial bandit settings.

Experiments

We provide experimental results on both synthesis datasets and Yahoo! Front Page Webscope datasets. We demonstrate the ability of our approach to solve real-world problems that can be modeled as ordered contextual combinatorial semi-bandits and that have not been throughly studied before.

Baseline algorithms

As baseline algorithms to compare against our proposed algorithms, we consider the following approaches.

Random. Randomly select S actions and S positions.

Unordered ϵ-Greedy (U-ϵ-greedy). 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 whole-page optimization which can not translate to a ranked list, this approach does not apply since it is not obvious on how to place the actions.

Exploitation. This is an extension of pure exploitation algorithm based on our network flow techniques. It maintains an estimate of the function value of each (action, position) pair. At each step, instead of enumerating the values on all 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. (2).

ϵ-Greedy. Use Exploitation with probability $1 - \epsilon$ and use Random with probability ϵ.

Ranked bandits (Radlinski, Kleinberg, and Joachims 2008). 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. MAB_m is responsible for select which action is displayed at rank m. If this action is already chosen at higher ranks, it randomly chooses a different action. The MAB instance is chosen as the UCB1-Normal algorithm (Auer, Cesa-Bianchi, and Fischer 2002) for context-free Gaussian processes optimization.

GP-UCB (Srinivas et al. 2009). This approach only works for Gaussian process optimization. A linear programing approach can not be used to select the best super arm with the highest upper confident bound. Hence 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.

Experiments on context-free Gaussian processes

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 (Richardson, Dominowska, and Ragno 2007) 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, d_k is randomly generated from $[0.3, 0.8]$ for different action 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. For Thompson sampling, we also consider the impact of posterior reshaping $\boldsymbol{K}_t \rightarrow \alpha^2 \boldsymbol{K}_t$ in the posterior sampling step. In particular, decreasing the variance would have the effect of increasing exploitation over exploration.

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: 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. The good performance of Thompson sampling is consistent with other empirical evaluations in existing literature on K-armed bandits. 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. One possible explanation is that even though we use RBA algorithm for multiple clicks per time, it is designed on one click per 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. The price to pay is higher variance. As for the impact of noise level, the variance of the algorithm grows when the noise level increases. U-ϵ-greedy is the biggest victim and Thompson sampling is the least affected.

News Article Recommendation

We consider applying Thompson sampling for contextual combinatorial bandits to personalization of news article recommendation on Yahoo! front page (Agarwal et al. 2009; Li et al. 2011; Chapelle and Li 2011). 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 optimize whole-page presentation. The candidate article pool is dynamic over time with an average size around 20. Suppose the user can click on more than one article during each view session. The goal is to maximize the total number of clicks.

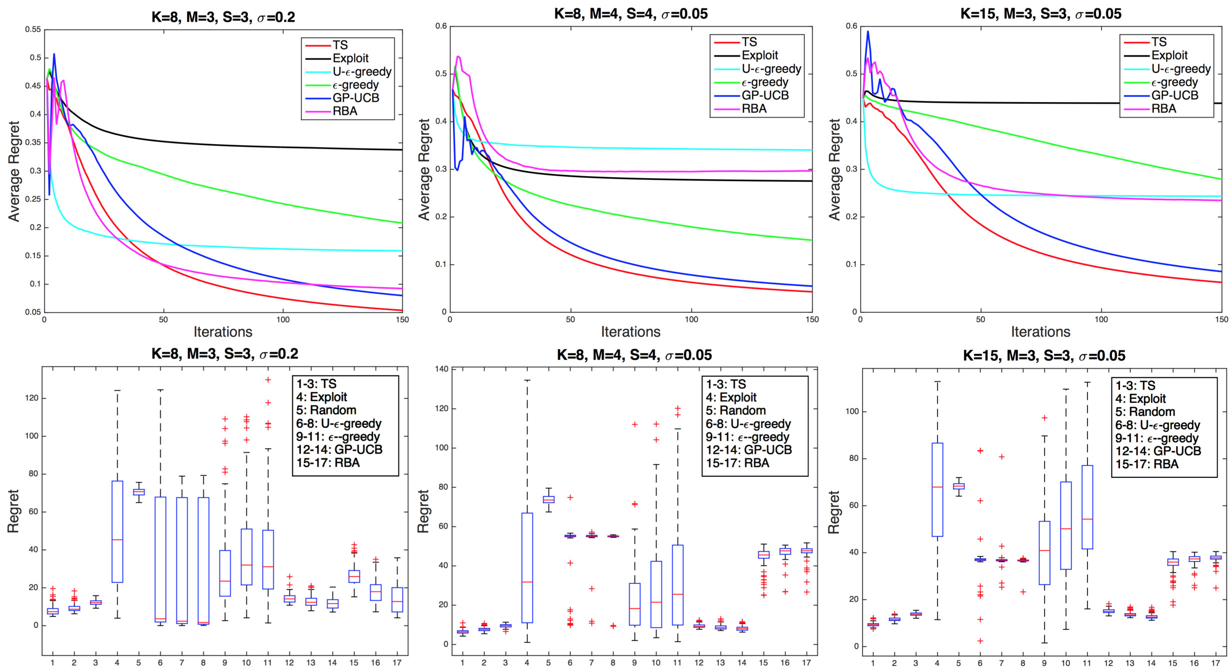

Figure 3: Comparison of performance: Thompson sampling and various heuristics. Top: average regret as a function of T over 100 repetitions. Bottom: distribution of the regret at T = 150.

In Yahoo! Webscope datasets (Yahoo! Webscope 2009), for each visit, the user and each of the candidate articles are associated with a feature vector (Chu et al. 2009). We treat each user vector as the context information x and the news articles as actions a. We set the number of possible positions $M = 5$. We use recursive Bayesian logistic regression (Eq. (3)(4)) based on the user click model (5) to predict article CTRs.

We can not use the unbiased offline evaluation (Li et al. 2011) 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 it more realistic, we first use all the user click data on May 1, 2009 to train a weight vector using logistic regression and then construct $\boldsymbol{w}^*$ by perturbation. We experiment with different choices of S. For the reasons explained in baseline algorithms, we compare our approach with Random, Exploitation and ϵ-greedy.

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 after 590,747 user visits that is normalized with respect to the number of selected actions in Table 1.

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 (Chapelle and Li 2011).

In order to demonstrate the practicability and scalability of our algorithm, we report run-time numbers as follows. The experiments are ran on a Linux server [Intel(R) Xeon(R) CPU X5650 2.67GHz, 8G memory]. The average running time of each LP is below 0.01s (per impression). In order to further test the scalability, we set the number of positions to M=20, which is enough in web applications. The average running time is below 0.05s. The delay for model refresh under Thompson sampling after every 10 minutes is around 9s. We emphasize that the goal of the experiments is not to claim the superiority of Thompson sampling, but to demonstrate our ability to optimize whole-page representation beyond ranking.

Method	TS (0.5)	TS(1)	Exploit
Reward ($S = 3$)	0.0862	0.0861	0.0846
Reward ($S = 5$)	0.0249	0.0246	0.0244
Method	Random	ϵ-greedy (0.02)	ϵ-greedy (0.01)
Reward ($S = 3$)	0.0538	0.0849	0.0845
Reward ($S = 5$)	0.0144	0.0243	0.0244

Table 1: Average rewards on Yahoo! Webscope datasets.

Conclusion

In this paper, we extend the traditional multi-armed bandit problems to a more general ordered combinatorial setting. This is motivated by many web applications with whole-page recommendation. By the adaptation of a min-cost max-flow network, a practical algorithm based on Thompson sampling is derived for (contextual) combinatorial semi-bandits, which does not resort to an oracle to provide approximation solutions. Our method has the ability to work with general layout information where the number of positions can be larger than the subset of arms selected and thus can optimize whole-page representation. 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.

References

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

Ahuja, R. K.; Magnanti, T. L.; and Orlin, J. B. 1993. Network flows: theory, algorithms, and applications.

Audibert, J.-Y.; Bubeck, S.; and Lugosi, G. 2013. Regret in online combinatorial optimization. *Mathematics of Operations Research* 39(1):31–45.

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

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

Bishop, C. M. 2006. *Pattern recognition and machine learning*, volume 4. springer New York.

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

Chu, W.; Park, S.-T.; Beaupre, T.; Motgi, N.; Phadke, A.; Chakraborty, S.; and Zachariah, J. 2009. A case study of behavior-driven conjoint analysis on yahoo!: Front page today module. In *15th ACM SIGKDD*, 1097–1104. ACM.

Chu, W.; Li, L.; Reyzin, L.; and Schapire, R. E. 2011. Contextual bandits with linear payoff functions. In *AISTATS*, 208–214.

Craswell, N.; Zoeter, O.; Taylor, M.; and Ramsey, B. 2008. An experimental comparison of click position-bias models. In *Proceedings of the 2008 WSDM*, 87–94. ACM.

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

Gai, Y.; Krishnamachari, B.; and Jain, R. 2012. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking (TON)* 20(5):1466–1478.

Gai, Y.; Krishnamachari, B.; and Liu, M. 2011. On the combinatorial multi-armed bandit problem with markovian rewards. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, 1–6. IEEE.

Gopalan, A.; Mannor, S.; and Mansour, Y. 2014. Thompson sampling for complex online problems. In *Proceedings of ICML*, 100–108.

Jie, L.; Lamkhede, S.; Sapra, R.; Hsu, E.; Song, H.; and Chang, Y. 2013. A unified search federation system based on online user feedback. In *Proceedings of the 19th ACM SIGKDD*, 1195–1203. ACM.

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

Krishnamurthy, A.; Agarwal, A.; and Dudik, M. 2016. Contextual semibandits via supervised learning oracles. In *Advances In Neural Information Processing Systems*, 2388–2396.

Lagun, D., and Agichtein, E. 2014. Effects of task and domain on searcher attention. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 1087–1090. ACM.

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

Li, L.; Chu, W.; Langford, J.; and Wang, X. 2011. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth WSDM*, 297–306. ACM.

Liu, Z.; Liu, Y.; Zhou, K.; Zhang, M.; and Ma, S. 2015. Influence of vertical result in web search examination. In *Proceedings of SIGIR*, 193–202. ACM.

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

Radlinski, F.; Kleinberg, R.; and Joachims, T. 2008. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th ICML*, 784–791. ACM.

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

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

Swaminathan, A.; Krishnamurthy, A.; Agarwal, A.; Dudík, M.; Langford, J.; Jose, D.; and Zitouni, I. 2016. Off-policy evaluation for slate recommendation. *arXiv preprint arXiv:1605.04812*.

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

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

Wang, Y.; Reyes, K. G.; Brown, K. A.; Mirkin, C. A.; and Powell, W. B. 2015. 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.

Wang, Y.; Yin, D.; Jie, L.; Wang, P.; Yamada, M.; Chang, Y.; and Mei, Q. 2016. Beyond ranking: Optimizing whole-page presentation. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 103–112. ACM.

Wang, Y.; Wang, C.; and Powell, W. 2016. The knowledge gradient for sequential decision making with stochastic binary feedbacks. In *Proceedings of The 33rd International Conference on Machine Learning*, 1138–1147.

Wen, Z.; Ashkan, A.; Eydgahi, H.; and Kveton, B. 2015. Efficient learning in large-scale combinatorial semi-bandits. In *Proceedings of the 32nd ICML*.

2009. *Yahoo! Webscope dataset ydata-frontpage-todaymodule-clicks-v1-0*. http://webscope.sandbox.yahoo.com/.