Markov Decision Process modeled with Bandits for Sequential Decision Making in Linear-flow

Wenjun Zeng
zengwenj@amazon.com
Amazon.com
Seattle, Washington, United States

Yi Liu
yiam@amazon.com
Amazon.com
Seattle, Washington, United States

ABSTRACT

In membership/subscriber acquisition and retention, we sometimes need to recommend marketing content for multiple pages in sequence. Different from general sequential decision making process, the use cases have a simpler flow where customers per seeing recommended content on each page can only return feedback as moving forward in the process or dropping from it until a termination state. We refer to this type of problems as sequential decision making in linear–flow. We propose to formulate the problem as an *MDP with Bandits* where Bandits are employed to model the transition probability matrix. At recommendation time, we use Thompson sampling (TS) to sample the transition probabilities and allocate the best series of actions with analytical solution through exact dynamic programming. The way that we formulate the problem allows us to leverage TS's efficiency in balancing exploration and exploitation and Bandit's convenience in modeling actions' incompatibility. In the simulation study, we observe the proposed *MDP with Bandits* algorithm outperforms Q-learning with ϵ-greedy and decreasing ϵ, independent Bandits, and interaction Bandits. We also find the proposed algorithm's performance is the most robust to changes in the across-page interdependence strength.

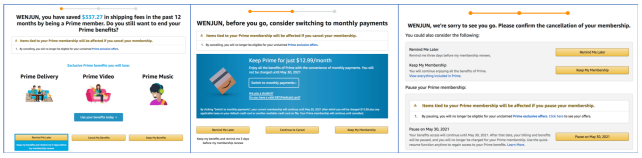
KEYWORDS

Bandit, Reinforcement Learning, posteriror sampling, MDP, recommendation

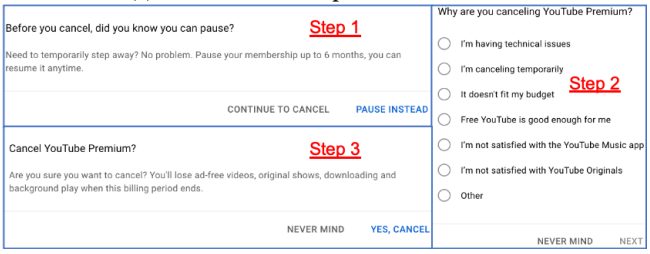
ACM Reference Format:
Wenjun Zeng and Yi Liu. 2021. Markov Decision Process modeled with Bandits for Sequential Decision Making in Linear-flow. In *Proceedings of Marble-KDD 21.* Singapore, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

For membership/subscriber acquisition and retention, we sometimes need to recommend marketing content for multiple pages in sequence. Two examples are shown in Figure 1 for Prime membership and Youtube Premium subscription cancellation flow. After customers trigger the flow, they go through three pages with different marketing content. At each page, they can choose to continue the cancellation or leave the process. Comparing to a one-click cancellation, a multiple-page flow avoids cancellations by accident and offers customers a benefit-informed cancellation process. As a result, while the process is straightforward for cancellation, it helps the retained customers to better leverage the service. Use cases like these require sequential decision making process since each page has a set of content candidates and the algorithm needs to make recommendation considering interdependence between content on successive pages. Different from general sequential decision making process, it is a simpler flow where customers per seeing recommended content on each page can only move forward in the process or drop from it. We define problems like this as sequential decision making in linear–flow. We observe this type of problems in customer engagement beyond content recommendation. For instance, it is common to take a series of nudges, such as promotion email and limited-time discount offer, to increase customer's activity. We can consider customers enter the workflow when their interactions drop below a threshold, and leave the flow with reward as +1 if their interactions increase to a healthy level or −1 if customers become inactive.



(a) Prime membership cancellation flow



(b) Youtube Premium subscription cancellation flow

Figure 1: Linear–Flow Examples

In this paper, we formulate sequential decision making process in linear-flow as a Markov Decision Process (MDP). We model the transition probability matrix with contextual Bayesian Bandits [3], use Thompson Sampling (TS) as the exploration strategy, and apply exact Dynamic Programming (DP) to solve the MDP. Modeling transition probability matrix with contextual Bandits makes it convenient to specify actions' incompatibility, when certain action pairs at different steps are ineligible. It also makes it straightforward to interpret interdependence strength between actions in two successive situations. Our algorithm is a realization of the Posterior Sampling for Reinforcement Learning (PSRL) framework [4, 5] with one difference. PSRL framework assumes each (state, action) pair is modeled separately while we model the pairs at a step using one Bayesian Bandit model. We present our algorithm in section 2. We

Marble-KDD 21, August 16, 2021, Singapore
2021. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

use simulator to evaluate the proposed algorithm comparing to Q-learning with ϵ-greedy and decreasing ϵ, independent Bandits, and interaction Bandits. The evaluation results are documented in section 3. We then conclude the paper in section 4. In the remainder of this paper, we relate writing to content recommendation applications while the algorithm works for any linear-flow problems.

2 METHODS

2.1 Problem Formulation

We model sequential decision making in linear flow as an MDP. Let D be number of the pages that content recommendation is needed. For each page i, there are N_i content candidates that we can choose from. The selected content can be denoted as a_{i,n_i} where $n_i \in \{1, 2, \cdots, N_i\}$. For simplicity, we will denote the content shown on page i as a_i unless the notation a_{i,n_i} is needed to distinguish between different content for page i. The flow starts with state $S_{1,X}$ where X is customer feature vector. After action a_i is selected to show to customer with features X on page i, there are two possible states: S_{i+1,X,a_i} and S_{Exit}. The only exception is for the last page, page D, the two possible states after action a_D are: S_{End} and S_{Exit}. S_{i+1,X,a_i} means customers move forward in the process to the next page after seeing a_i. S_{Exit} means customers drop from the process without reaching the end and S_{End} means customers reached the end. We denote R_i and G_i as short–term and long–term rewards after action a_i respectively. The goal is to optimize G_1. In the cancellation use cases as in Figure 1, customers are at $S_{1,X}$ after they trigger the cancellation flow. X may include customer features, the upstream channel that customers land from, etc. Customers get to S_{Exit} if they drop from the process at any page leading to $R_i = 1$. Customers are in S_{i+1,X,a_i} if they move on in the process after marketing content a_i. The short-term reward is 0 from entering S_{i+1,X,a_i}. S_{End} is another termination state other than S_{Exit}. Customers reach this state if they cancel their membership after seeing content on the last page. This leads to $R_D = 0$.

2.2 Transition Probability Matrix Modeling

We use contextual Bandit to model and update transition probability matrix. As our rewards are binary and there can be only two outcomes per action, we select BLIP [2] as the Bandit algorithm. The transition probability is formulated through a Probit link function. After showing page i, the probability of customers moving forward in the process and dropping from the process is formulated as in Equations 1 and 2 respectively. By using this formulation, we model only interdependence between two successive pages. For page 1, a_{i-1} is dropped from the subscripts but the rest of the equations remain the same.

$$P(S_{t+1} = S_{i+1,X,a_i} | S_t = S_{i,X,a_{i-1}}, a_t = a_i) = \Phi(-1 * \frac{B^T_{X,a_{i-1},a_i} W_i}{\beta}) \tag{1}$$

$$P(S_{t+1} = S_{Exit} | S_t = S_{i,X,a_{i-1}}, a_t = a_i) = \Phi(\frac{B^T_{X,a_{i-1},a_i} W_i}{\beta}) \tag{2}$$

In the formula, Φ is cumulative distribution function for standard normal distribution. β is a scaling hyperparameter. W_i are the weights quantifying feature contributions to the utility function. B_{X,a_{i-1},a_i} is the final vector that we will learn weights for and formed by combining X, a_{i-1} and a_i (usually with interaction terms). B_{X,a_{i-1},a_i} is the reason why we can easily model content incompatibility. To explain this, let's first assume we have one feature x for X, a_{i-1} has two candidates: $a_{i-1,1}$ and $a_{i-1,2}$, and a_i has three candidates: $a_{i,1}$, $a_{i,2}$ and $a_{i,3}$. We assume $a_{i,3}$ cannot be shown if $a_{i-1,1}$ is shown to customers. We may set B_{a_{i-1},a_i} as in Equation 3 where we have 1 for intercept and contextual feature interacting with action terms. We recognize content incompatibility between $a_{i-1,1}$ and $a_{i,3}$ by simply omitting interaction term $x \cdot a_{i-1,1} \cdot a_{i,3}$ in the equation. In addition, we can interpret interdependence strength between pages by checking the weights for the interaction terms across pages.

$$\begin{aligned} B^T_{X,a_{i-1},a_i} = (&1, x \cdot a_{i,1}, x \cdot a_{i,2}, x \cdot a_{i,3}, \\ &a_{i-1,1} \cdot a_{i,1}, a_{i-1,1} \cdot a_{i,2}, \\ &a_{i-1,2} \cdot a_{i,1}, a_{i-1,2} \cdot a_{i,2}, a_{i-1,2} \cdot a_{i,3}) \end{aligned} \tag{3}$$

We assume the weights W_i follow Normal distribution parameterized by a mean vector $\boldsymbol{\mu}_{W_i}$ and variance vector $\boldsymbol{\nu}_{W_i}$. We update the distribution from prior to posterior by enforcing Normal distribution as the distribution type and obtaining the distribution parameters by minimizing KL-divergence from the approximate normal distribution and the exact distribution.

2.3 RL Policy and Policy Updating

We propose Algorithm 1 *MDP with Bandits* to solve the formulated MDP. $\mathscr{D}_d$ denotes the dataset for updating contextual Bandit for Page d. k denotes the k^{th} impression. $a^*_i(a_{i-1})$ denotes the optimal action for page i given content on page $i-1$. The complexity of the algorithm is $O(DN^2K)$ assuming number of contents on each page is N for discussion simplicity. For each observation, we sample weights from their posterior distribution using TS for calculating transition probability matrix. We then employ exact DP to allocate the optimal series of content to show on each page given the weights. Per reward signal availability, we update the weights distributions and the loop repeats. If we get the feedback signal in batches, the algorithm alters on Lines 11 and 12 where we would have the updates at batch level instead of observation level.

3 SIMULATION

Using simulation, we compare the proposed *MDP with Bandits* algorithm to three other algorithms: independent Bandits, interaction Bandits and Q-learning. Independent Bandits algorithm assumes no interdependence across pages. TS for each Bandit model is conducted separately for each page. Interaction Bandits algorithm models interdependence in two successive pages by using the content shown in a previous page to construct features in the Bandit formulation. Other than the feature construction, TS for each Bandit model in the interaction Bandits algorithm is also conducted separately. As shown in Table 1, in *MDP with Bandits*, we use short-term reward R_i as the feedback signal for Bandits while independent and interaction Bandits use the long-term reward G_i as the feedback

Algorithm 1 MDP with Bandits

Init $\mathscr{D}_1 = \mathscr{D}_2 = \cdots = \mathscr{D}_D = \emptyset$
for $k = 1, \cdots, K$ **do**
  Receive context x_k and Initiate $\mathbb{E}[G|a_D] = 0$
  Sample $W_1, \cdots, W_D$ from the posterior $P(W_1|\mathscr{D}_1), \cdots, P(W_D|\mathscr{D}_D)$ ▷ Thompson Sampling
  for $i = D, D-1, \cdots, 2$ **do** ▷ Line 5 - 11: Exact Dynamic Programming
    for a_{i-1} in content pool of page $i-1$ **do**:
      $a_i^*(a_{i-1}) = \underset{a_i}{\text{argmax}}\{\mathbb{E}[R|a_i, a_{i-1}, x_k; W_i] + (1 - \mathbb{E}[R|a_i, a_{i-1}, x_k; W_i]) * \mathbb{E}[G|a_i]\}$
      $\mathbb{E}[G|a_{i-1}] = \mathbb{E}[R|a_i^*(a_{i-1}), a_{i-1}, x_k; W_i] + (1 - \mathbb{E}[R|a_i^*(a_{i-1}), a_{i-1}, x_k; W_i]) * \mathbb{E}[G|a_i^*(a_{i-1})]$
    end for
  end for
  $a_1^* = \underset{a_1}{\text{argmax}}\{\mathbb{E}[R|a_1, x_k; W_1] + (1 - E[R|a_1, x_k; W_1])\,\mathbb{E}[G|a_1]\}$
  Display layout $\{a_1^*, a_2^*(a_1^*), \cdots, a_D^*(a_{D-1}^*)\}$ and observe reward $R_{1,k}, \cdots, R_{D,k}$ ▷ Take actions and collect feedbacks
  Update $\mathscr{D}_1 = \mathscr{D}_1 \cup (x_k, a_{1k}, R_{1k})$ and update $\boldsymbol{\mu}_{W_1}, \boldsymbol{\nu}_{W_1}$ ▷ BLIP Updating. Formula (7, 8) in [2]
  Update $\mathscr{D}_i = \mathscr{D}_i \cup (x_k, a_{ik}, a_{i-1,k}, R_{i,k})$ and update $\boldsymbol{\mu}_{W_i}, \boldsymbol{\nu}_{W_i}$ for $i \in \{2, 3, \cdots, D\}$ if a_{ik} was presented
end for

signal. For notation simplicity, we use the tide sign $\sim$ to introduce the linear model form used for Equations 1 and 2 with Probit link function assumed as default. The Bandits forms only involve content features to highlight the action interaction. *MDP with Bandits* and interaction Bandits have the same linear model form. Lift in performance from *MDP with Bandits* (if any) directly quantifies the benefit of modeling the process as an MDP instead of a series of Bandits. For Q-learning, we follow the vanilla setup: we model the content shown in the previous page as the state and update the Q-Table based on Bellman Equation. We set the learning rate to be 0.05 and discount factor to be 1. The policy is based on ϵ-greedy with ϵ linearly decreases from 0.05 (2nd batch) to 0.01 (last batch).

3.1 Simulated Data

We generate simulation data assuming Probit link function between short-term reward R_i and $B^T_{X,a_{i-1},a_i} W_i$ based on equations 1 and 2, with β set as $1 + \alpha_1 + \alpha_c + \alpha_2$ and the linear model forms for $B^T_{X,a_{i-1},a_i} W_i$ specified as in equation 4 for first page and equation 5 for the other pages. Similar to [3], we use the α_1, α_c and α_2 to separately control the importance of content to recommend on the current page, content already shown on the previous page, and the interaction between them. For each simulation run, the ground truth weights W_i are sampled independently from a normal distribution with mean 0 and variance 1 except for w_i^0. Mean for w_i^0 is set as $\Phi^{-1}(x)\beta$ where x reflects the average success rate. With a larger value for α_2, interdependence across pages is set stronger.

$$B^T_{X,a_1} W_1 = w_1^0 + \alpha_1 \sum_{n_1=1}^{N_1} w^1_{n_1} a_{1,n_1} + \alpha_1 \sum_x w^1_x x \tag{4}$$

$$\begin{aligned} B^T_{X,a_{i-1},a_i} W_i = {} & w_i^0 + \alpha_1 \sum_{n_i=1}^{N_i} w^1_{n_i} a_{i,n_i} + \alpha_1 \sum_x w^1_x x + \\ & \alpha_c \sum_{n_{i-1}=1}^{N_{i-1}} w^c_{n_{i-1}} a_{i-1,n_{i-1}} + \\ & \alpha_2 \sum_{n_i=1}^{N_i} \sum_{n_{i-1}=1}^{N_{i-1}} w^2_{n_i,n_{i-1}} a_{i,n_i} a_{i-1,n_{i-1}} + \\ & \alpha_2 \sum_{n_i=1}^{N_i} \sum_x w^2_{n_i,x} a_{i,n_i} x \quad \text{for} \quad i \in 2, \cdots, D \end{aligned} \tag{5}$$

For each simulation, we sample the actual weights in equations 4 and 5 once and we calculate the long-term reward of each combination $(a_1, \cdots, a_D)$ as:

$$\begin{aligned} & \mathbb{E}[G|W, a_1, \cdots, a_D] = \\ & \qquad \mathbb{E}[R_1 + (1 - R_1)R_2 + \cdots + \Pi_{i=1}^{D-1}(1 - R_i)R_D] \end{aligned} \tag{6}$$

where $R_i = \Phi(\frac{B^T_{X,a_{i-1},a_i} W_i}{\beta})$ for $i \in 2, \cdots, D$ and $R_1 = \Phi(\frac{B^T_{X,a_1} W_1}{\beta})$.

The series of actions for each algorithm are selected by itself. Given sampled weights and selected actions, empirical rewards (R) are sampled from the data generation model. Actual reward G_k is 0 if any R_k is 0. Each simulation was run for $K = 14k$ time steps with updating batch size $1k$ to mimic a 2-week experiment. We ran 100 simulations and report the averaged performance of each algorithm using batch size normalized cumulative regret:

$$\frac{1}{\#runs} \sum_{j=1}^{\#runs} \frac{1}{batchSize} \sum_{k=1}^{K} (\mathbb{E}[G|W, a^*_{1,j}, \cdots, a^*_{D,j}] - G_{j,k}) \tag{7}$$

Where $\mathbb{E}[G|W, a^*_{1,j}, \cdots, a^*_{D,j}]$ is the best expected long-term reward for j^{th} run and $G_{j,k}$ is the long-term reward for j^{th} run and k^{th} time step.

Table 1: Algorithms in Comparison

Algorithm	Linear model form	# parameters
MDP with Bandits	Page 1: $R_1 \sim a_1$; Page i: $R_i \sim a_i + a_{i-1} + a_{i-1} : a_i$ for $i \in 2, \cdots, D$	$O(DN^2)$
Interaction Bandits	Page 1: $G_1 \sim a_1$; Page i: $G_i \sim a_i + a_{i-1} + a_{i-1} : a_i$	$O(DN^2)$
Independent Bandit	Page 1: $G_1 \sim a_1$; Page i: $G_i \sim a_i$	$O(DN)$

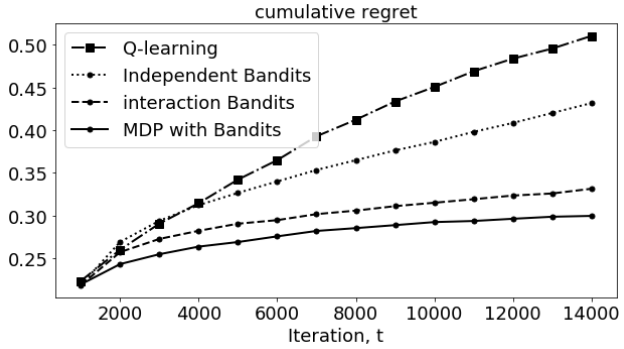


Figure 2: Example run of algorithms on simulated data with $\alpha_1 = 1, \alpha_c = 1, \alpha_2 = 2$

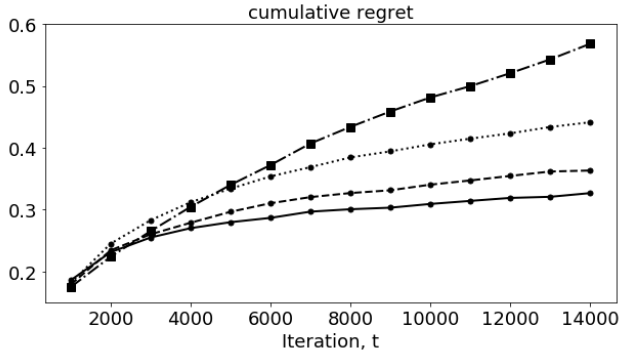


Figure 3: Example run of algorithms on simulated data with $\alpha_1 = 1, \alpha_c = 1, \alpha_2 = 2$ **and 1 categorical feature with 3 categories**

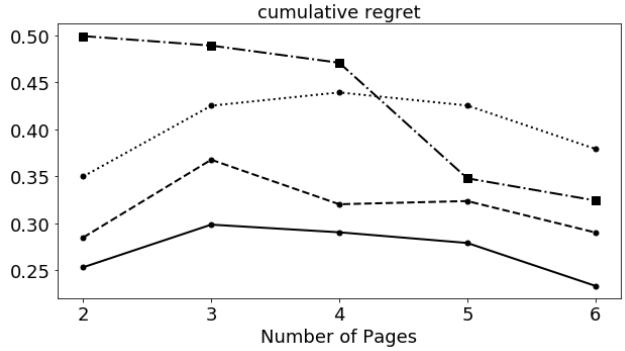


Figure 4: Algorithm performance as number of pages vary. $\alpha_1 = 1, \alpha_c = 1, \alpha_2 = 2$

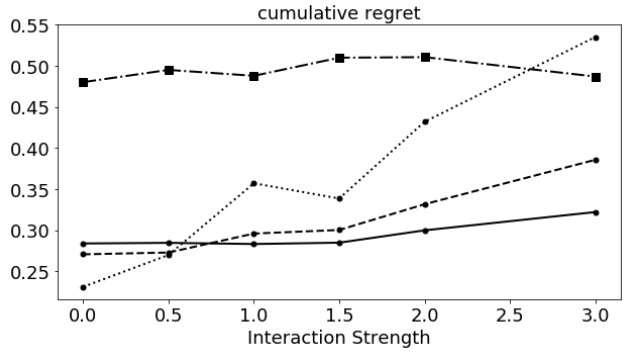


Figure 5: Algorithm performance as α_2 **is varied.** $\alpha_1 = 1, \alpha_c = 1$

3.2 Simulation Results

We first compare the performance with $\alpha_1 = 1, \alpha_c = 1, \alpha_2 = 2$ in both non-contextual and contextual use case. In the contextual case, we include 1 categorical feature with 3 equally-likely categories. We assume 3 pages and each page has 3 content candidates. Results are shown in figures 2 and 3. The proposed algorithm converges faster than all the other three. Also, it has the lowest cumulative regret followed by interaction Bandits. This means when there is considerable interdependence across pages, modeling interactions in the transition probability reduces regrets and modeling the process as an MDP instead of separate Bandits further lifts the performance. On the other hand, bandit algorithms with TS generally outperform Q-learning with ϵ-greedy, given its efficiency in exploration.

We then take a look at the impact of number of pages on performance, shown in figure 4. The number of pages varies from 2 to 6 and that of content combinations N^D varies from 9 to 729. *MDP with Bandits* consistently outperforms the other three algorithms. The performance advantage over Q-learning shrinks with increasing page number.

To test the model's sensitivity to interdependence strength, we conduct a set of simulation runs with α_2 increasing from 0 to 3 while fixing α_1 and α_c at 1. Results are shown in figure 5. When α_2 is 0, independent bandits algorithm outperforms the other two bandit algorithms as both interaction Bandits and *MDP with Bandits* require traffic to learn the weights for interaction terms should actually be set as 0. As α_2 increases, *MDP with Bandits* starts to outperform. As Q-learning directly learns the Q function by Bellman equation, interaction strength has minor influence to its performance. Overall,

the performance for *MDP with Bandits* is consistently better than all other algorithms when interaction strength is greater than 1.

When scanning through figures 2, 3, 4, 5, we find interaction Bandits is the 2nd best when *MDP with Bandits* is the best and its performance is comparable to that of *MDP with Bandits*. When interaction strength is small, it can perform better than *MDP with Bandits*, as at the left bottom of figure 5. In addition, similar to *MDP with Bandits*, the cumulative regrets of interaction Bandits always level out as in figures 2, 3. These mean interactive Bandit, where separate contextual Bandits with long-term reward and action shown on the previous page as feature are built to model each page, is a good surrogate modeling strategy to a multi-step RL and always converges to the optimal actions in the case of stationary reward. This is because that interactive Bandit samples the best action for each page under the expected rewards of all the trajectories after the action. With improving performance estimation regarding each trajectory along training, the algorithm is able to learn the trajectory with the highest long-term reward. We do not though consistently observe convergence for independent Bandits and Q-learning. We observe independent Bandits got stuck in local optimal actions that maximize short-term but not long-term reward for some simulation runs, even if it is using the long-term reward. For Q-learning, theoretically it is guaranteed to converge, but with the conditions that the learning rate needs to be diverged and each state-action pair must be visited infinitely often [7]. In practice, it's not trivial to tune the parameters and the convergence may not be obtained.

4 CONCLUSIONS

We proposed a RL algorithm with posterior sampling for recommendation in multi-step linear–flow. We set the problem up as an *MDP with Bandits* to model transition probability matrix. At policy inference time, we use TS to sample the transition probabilities and allocate the best series of actions with exact dynamic programming. The algorithm is able to leverage TS's efficiency in balancing exploration and exploitation and Bandit's convenience in modeling actions' incompatibility. In the simulation study, we observe the proposed algorithm outperforms Q-learning with ϵ-greedy, independent Bandits, and interaction Bandits, under different scenarios as long as there is reasonable interdependence strength across pages. We also see the proposed algorithm is the most robust to changes in the interdependence strength. We often face dynamic environments where actions and interdependence between sequential actions are subject to change. Robustness to changes is thus a good to have. On the other hand, we find interactive Bandit, where separate contextual Bandits with long-term reward and action shown on the previous page as feature are built to model each page, is a good surrogate modeling strategy to a multi-step. Just like the proposed algorithm, it always converges to the optimal actions in the case of stationary reward. This is evidence that when the long-term impact of actions is significant, Bandits if set up properly may not be the best formulation, but can still be a good baseline.

Our work is a start in using Bandit (1-step RL) to solve stateful RL (multi-step RL). More efforts are in need to generalize the application. For instance, one desired algorithm extension is to use Bandit to model a multi-stage flow where customers can move forward with more than two options after seeing an action. In terms of algorithm evaluation, we seek to leverage logged data for counterfactual evaluation in addition to simulation analysis [1, 6].

ACKNOWLEDGMENTS

We would like to thank Lihong Li and Shipra Agrawal for their valuable comments and helpful discussions.

REFERENCES

[1] Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601* (2011).
[2] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. Omnipress.
[3] Daniel N Hill, Houssam Nassif, Yi Liu, Anand Iyer, and SVN Vishwanathan. 2017. An efficient bandit algorithm for realtime multivariate optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1813–1821.
[4] Ian Osband, Daniel Russo, and Benjamin Van Roy. 2013. (More) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*. 3003–3011.
[5] Ian Osband and Benjamin Van Roy. 2017. Why is posterior sampling better than optimism for reinforcement learning?. In *International Conference on Machine Learning*. PMLR, 2701–2710.
[6] Alex Strehl, John Langford, Sham Kakade, and Lihong Li. 2010. Learning from logged implicit exploration data. *arXiv preprint arXiv:1003.0120* (2010).
[7] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.