A Map of Bandits for E-commerce

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

Lihong Li*
llh@amazon.com
Amazon.com
Seattle, WA, United States

ABSTRACT

The rich body of Bandit literature not only offers a diverse toolbox of algorithms, but also makes it hard for a practitioner to find the right solution to solve the problem at hand. Typical textbooks on Bandits focus on designing and analyzing algorithms, and surveys on applications often present a list of individual applications. While these are valuable resources, there exists a gap in mapping applications to appropriate Bandit algorithms. In this paper, we aim to reduce this gap with a structured map of Bandits to help practitioners navigate to find relevant and practical Bandit algorithms. Instead of providing a comprehensive overview, we focus on a small number of key decision points related to reward, action, and features, which often affect how Bandit algorithms are chosen in practice.

KEYWORDS

Bandit, reward, action, E-commerce, recommendation

ACM Reference Format:
Yi Liu and Lihong Li. 2021. A Map of Bandits for E-commerce. In *Proceedings of Marble-KDD 21.* Singapore, 5 pages.

1 INTRODUCTION AND MOTIVATION

Bandit is a framework for sequential decision making, where the decision maker ("agent") sequentially chooses an action (also known as an "arm"), potentially based on the current contextual information, and observes a reward signal. The typical goal of the agent is to learn an optimal action-selection policy to maximize some function of the observed reward signals. This problem is a special case of reinforcement learning (RL) [1], and has been a subject of extensive research in AI.

The main reason for extensive research of Bandit in the literature is its wide applications. The paper focuses on one of the most important domains, E-commerce, including online recommendation, dynamic pricing, supply chain optimization, among others [2]. For instance, Bandit has been used for online recommendation across companies such as Amazon, Google, Netflix, and Yahoo! [3]. Early applications were on optimizing webpage content suggestion such as news articles, advertisement, and marketing messages [4, 5]. Nowadays, its applications have been extended to dynamic pricing [6], revenue management [7], inventory buying [8], as well as recommendation of various contents such as skills through virtual assistants [9].

The rich body of literature not only offers a diverse toolbox of algorithms, but also makes it hard for a practitioner to find the right solution to solve the problem at hand. A main challenge lies in the many choices when formulating a Bandit problem, and the resulting combinatorial explosion of problem space and algorithms. Typical textbooks on Bandits focus on designing and analyzing algorithms [10, 11], and surveys on applications often present a list of individual applications [e.g., 3]. While they are valuable resources, there is a gap in mapping applications to algorithms.

This paper aims to reduce this gap, by presenting a structured map for the world of Bandits. The map consists of a few key decision points, to guide practitioners to navigate in the complex world of Bandits to locate proper algorithms. While we use E-commerce as running examples, the map is useful to other applications. Furthermore, it is beyond the scope of the paper to provide a *comprehensive* map. Instead, our map only focuses on a small number of factors that often affect how Bandit algorithms are chosen in practice.

The map entry is section 2, which assesses whether Bandit is the right formulation. Sections 3 and 4 describe the navigational details of the map, to help locate appropriate algorithms by inspecting several properties of rewards and actions of the application at hand. Section 5 complements the map with a discussion of topics that a practitioner often faces. Section 6 concludes the paper.

2 MAP ENTRY: IS BANDIT THE RIGHT FORMULATION?

In typical sequential decision making modeled by Bandit, the agent repeats the following in every step: observe contextual information x_t, take an action a_t from an action set A_t, and receive reward r_t, where t denotes the step. The reward depends on x_t and a_t. The objective of a Bandit algorithm is to recommend actions for each step to maximize the expected cumulative reward over time: $\sum_{t=1}^{T} E[r_t]$ where T is the total number of steps. Suppose we want to recommend a video (a_t) from recently released ones (A_t) to customers, an application seen with Amazon Prime video, Netflix, HBO, etc. We want to consider customer features to improve the recommendation relevance: film genre (drama, romance, etc.) preference of the customer (x_t). The business metric of interest is total video viewers. The observed reward (r_t) can then be defined as 1 if the customer watches the recommended video and 0 otherwise. Assume stationarity and linear structure, we may model the reward as $E[r_t(x_t, a_t)] = g(w_0 + w_1 \cdot a_t + w_2 \cdot x_t \cdot a_t)$ where w_i's are weights parameters and g is a link function to map a linear predictor to the mean of the reward. Logit and Probit are common link functions in Bandit applications.

The term "Bandit" refers to the fact that only the reward of the chosen action is observed. If rewards of *all* actions are observed, the setting is called "full-information" [12]. For example, consider typical multi-class classification, where predicting the label for an instance can be viewed as choosing an action. It is full-information, because once we know an instance's correct label, the rewards of

*Both authors contributed equally to this research.

Marble-KDD 21, August 16, 2021, Singapore
2021. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

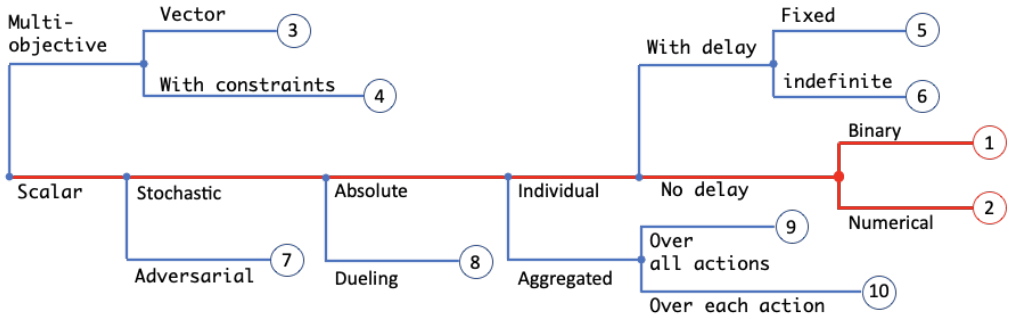

Figure 1: Bandit problems by reward properties

all predictions are known (1 for the correct prediction and 0 otherwise). One should follow a different algorithmic path of supervised learning [13, 14] if the application is in the full-information setting.

On the other hand, in the more general RL setting, the next context ("state") may depend on previous contexts and taken actions, so the agent needs to reason with long-term rewards using more complex algorithms like Q-learning [1]. For example, suppose we want to maximize sales from an online shopping website, which has search, (product) detail, add-to-cart, and checkout pages. The shopper's state on the detail page is affected by results shown earlier on the search page. The revenue at the end of each shopping session depends on information on all pages in the session. When long-term impacts are significant, Bandits may not be the best formulation, but can still be a good baseline or starting point [15].

3 NAVIGATION BY REWARD

The nature of reward signals in an application plays a major role in deciding the right Bandit algorithms. Figure 1 identifies six key properties of rewards that lead to 10 typical use cases in practice. The highlighted two paths are perhaps the most common. The key reward properties are:

Dimension The reward can be one-dimension (scalar) or multi-dimension (vector). In the latter case, the task can be optimizing the vector reward (node 3), or maximizing one reward dimension subject to constraints on remaining dimensions (node 4).

Distributional assumption In stochastic Bandits, the reward is drawn from an unknown distribution. When the reward distribution may change slowly over time, as in many real-world applications, one can still treat it as a stochastic Bandit by constantly retraining the policy with new data. In adversarial Bandits, there is no probabilistic assumption on the reward (node 7).

Relativity While a reward usually measures how good an action is, in some problems like ranking one can also work with relative rewards that compare actions, as in dueling Bandits (node 8).

Granularity When actions are combinatorial (see section 4), the reward can be for the entire action (node 9), or can provide signal for subactions that comprise the action (node 10). The latter setting is also known as semi-bandits.

Delay Practical limitations like software constraints may prevent rewards from being observed before the next actions have to be taken. There are two types of reward delays: bounded (node 5) and indefinite (node 6), depending on whether we have a reasonably small upper bound for the delay.

Value type Reward can be binary (node 1) or numerical (node 2). These two leaf nodes are unique as value type must be defined for the other seven nodes to complete the reward formulation. We take this into consideration for algorithm recommendation in Table 1.

We design the structure in Figure 1 so that it covers common use cases in E-commerce. Leaf nodes 3–10 are not exhaustive as the splits are not mutually exclusive. For instance, adversarial Bandits can also be a dueling one [30] and there can be delay in reward [22]. In practice, however, such combinations appear uncommon.

In Table 1, for each leaf note we list example business problems with suggested algorithms. Given this paper's focus, we recommend algorithms that are empirically validated, especially those that find wide applications in practice. For leaf nodes 3–10, we may have two suggested benchmark papers when binary and numerical reward are both common.

To find Bandit solutions for a given business problem, one can use Figure 1 as a guide to land in the most relevant node, then refer to Table 1 for similar applications and algorithmic suggestions. We emphasize that there are no universally best algorithms, but expect the suggested references offer a good starting point for algorithm development and experimentation.

4 NAVIGATION BY ACTION

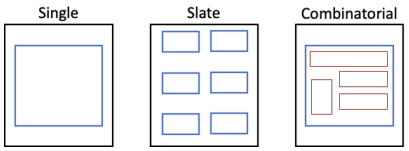

Figure 2: Illustration of three action types

The action set A also plays an important role in determining the right bandit algorithm. Here, we identify two properties: action type and action set size. The cases based on these properties are not exclusive to those identified in the previous section, but are orthogonal in many scenarios.

Figure 2 shows three common action types: single, slate and combinatorial. In the first, the action set is a set of items. The set is often small and finite, but can also be large or even infinite. In the second, the action is a slate consisting of a ranked list of items

Table 1: Example business applications and references for cases in Figure 1

No.	Business problems	References
1	Estimate click-through rate in online ad selection; Recommend answers to questions from users to virtual assistant	[4, 16]
2	Recommend products on webpage to maximize revenue; Optimize inventory buying to maximize cash flow	[17, 18]
3	Recommend video content to maximize streaming customers and also paid service subscriber size	[19]
4	Maximize advertisement click-through rate with cost budget for bidding; Inventory selection to maximize revenue considering inventory capacity	[20, 21]
5	N-day free-trial marketing (Robinhood Gold, Netflix Subscription, Spotify Premium) for acquiring paid member. We do not know if users will become paid members or not until day N	[22]
6	Users may not watch a recommended video until later; Users may not buy a product advertised until later. Delay in a potential positive action is not time bounded.	[23, 24]
7	Malicious activities (fake reviews, click fraud, etc.) modeling in recommender system; adaptive shortest path routing	[25–27]
8	Interleaved search evaluation, with relative feedback for two search results extracted from user clicks	[28]
9	Recommendation of a combination of contents where reward cannot be easily attributed to individual content	[5]
10	Page layout selection where reward can be traced back to the component on the page	[29]

Table 2: Example business applications and references for Single, Slate and Combinatorial Bandits

Action type	Business problems	References
Single	Estimate click-through rate in online ad selection; recommend products on webpage to maximize revenue	[18, 24, 31]
Slate	Return a ranked result list for user's search query; show multiple Fashion products on the landing page	Position effect: [32, 33]; Diversity effect: [34, 35]
Combinatorial	Select components (image, title, action button, etc.) to form optimal marketing messages; whole page optimization	[5]

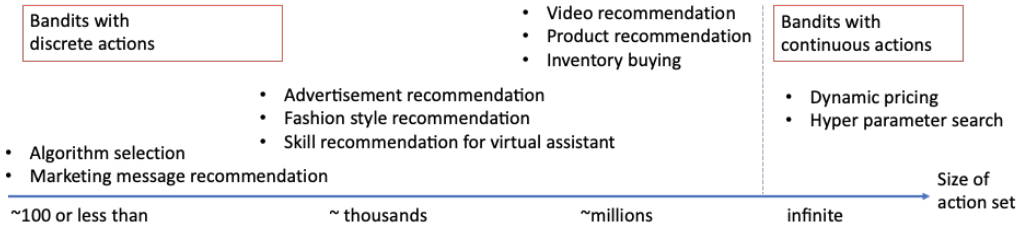

Figure 3: Example business applications by action set sizes

chosen from a pool, such as ranked results for a given search query. The challenge is the exponentially many possible permutations that require more efficient algorithms. Furthermore, we need to consider two effects: position bias for actions shown at different slots and item diversity in the overall slate. In the third, the action is a combinatorial object (such as content layout on a web page), consisting of sub-actions coming from different sets. The algorithmic challenge is often in dealing with combinatorial explosions of actions, and with interaction effects between sub-actions. In the literature, slate bandits are sometimes referred to as combinatorial. Table 2 lists business problems and recommended work by these two types.

Another action property is the size of the action set. In Figure 3, we list representative business applications with increasing size. Many Bandit use cases have discrete actions. When the action space is small, actions ID can be considered as a categorical value and encoded as a set of binary variables in the reward formulation. When the action space becomes large, we use features to represent actions. Such action featurization not only reduces the dimension of variables but also enables generalization across actions which mitigates action cold-start problem. While action and contextual features contain different meanings, they can be handled in a similar manner. More discussions are found in section 5. It is also common to see continuous actions in practice. In this case, the continuous action set often has a natural distance metric, so that one can use tools like Gaussian process to solve the Bandit problem [36, 37].

5 OTHER TOPICS

This section discusses feature engineering, offline policy evaluation, and best-arm identification, three important topics in Bandit applications to complement the previous sections.

5.1 Feature Engineering

In many applications, we use features to deal with large context or action sets more efficiently. The expected reward r can be written as a function of action features $\boldsymbol{\phi_a}$ and contextual features $\boldsymbol{\phi_x}$: $E[r] =$

$f(\boldsymbol{\phi_a}, \boldsymbol{\phi_x})$. Feature engineering in Bandit involves selection/preprocessing of $\boldsymbol{\phi_a}$ and $\boldsymbol{\phi_x}$ and their interaction terms. Linear Bandits are the most studied in the literature where $E[r]$ is assumed to be linear in the features. To model non-linearity especially when the size(s) of $\boldsymbol{\phi_a}$ or/and $\boldsymbol{\phi_x}$ is large, we can learn lower-dimension embeddings from the raw features ($\boldsymbol{\phi_a}$ or/and $\boldsymbol{\phi_x}$) and put the embeddings in the reward function instead [38, 39]. Embedding generation techniques for supervised learning generally apply to Bandits [40]. Another way to relax the linear-reward assumption is to use non-linear Bandits where reward function becomes non-linear in feature vectors [e.g., 31].

5.2 Offline Policy Evaluation

Testing a Bandit policy in real user traffic is often expensive, and poses risks on user experiences. It is common to evaluate a new policy offline before deploying it. A key challenge in offline policy evaluation is that we do not know how users would have reacted to actions different from the one in the log data, since the data only have rewards for selected actions. This *counterfactual* nature makes Bandit offline evaluation similar to causal inference where we want to infer the average reward $E_\pi[r]$ (the causal effect) if policy π is used to choose actions. There exist effective approaches to evaluating a stationary policy, including simulation [41], inverse propensity scoring [42, 43], doubly robust evaluation [44], and self-normalized inverse propensity estimators [45]. Typically, offline evaluation becomes more challenging with a larger action set. For a slate Bandit, pseudoinverse estimator is available to account for position bias [46]. Offline evaluation for non-stationary Bandits remains challenging [47, 48], with opportunities for further research.

5.3 Best-arm Identification

In some bandit applications, our goal is not to maximize reward during an experiment, but to identify the best action (e.g., best marketing campaign strategy) at the end of the experiment. This problem, known as best-arm identification [49–51], shares the same goal as conventional A/B/N testing [52], but can be statistically more efficient by adaptively selecting actions during an experiment.

6 CONCLUSIONS

We presented a structured map for the world of Bandits, with the hope to guide practitioners to navigate to practical Bandit algorithms. The work is not attempted to provide a comprehensive map. Instead, it focuses on a few key decision points that often affect how Bandit algorithms are chosen. We hope it reduces the gap in connecting applications to appropriate algorithms.

REFERENCES

[1] R. S. Sutton and A. G. Barto., *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press, 2018.

[2] D. Bouneffouf and I. Rish, "A survey on practical applications of multi-armed and contextual bandits," 2019. arXiv:1904.10040.

[3] E. Gangan, M. Kudus, and E. Ilyushin, "Survey of multi-armed bandit algorithms applied to recommendation systems," *International Journal of Open Information Technologies*, vol. 9, no. 4, pp. 12–27, 2021.

[4] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *WWW*, 2010.

[5] D. N. Hill, H. Nassif, Y. Liu, A. Iyer, and S. Vishwanathan, "An efficient bandit algorithm for realtime multivariate optimization," in *KDD*, pp. 1813–1821, 2017.

[6] K. Misra, E. M. Schwartz, and J. Abernethy, "Dynamic online pricing with incomplete information using multiarmed bandit experiments," *Marketing Science*, vol. 38, no. 2, pp. 226–252, 2019.

[7] K. J. Ferreira, D. Simchi-Levi, and H. Wang, "Online network revenue management using Thompson sampling," *Operations Research*, vol. 66, no. 6, pp. 1586–1602, 2018.

[8] H. Yuan, Q. Luo, and C. Shi, "Marrying stochastic gradient descent with bandits: Learning algorithms for inventory systems with fixed costs," *Management Science*, pp. 1–27, 2021.

[9] S. Upadhyay, M. Agarwal, D. Bounneffouf, and Y. Khazaeni, "A bandit approach to posterior dialog orchestration under a budget," 2019. arXiv:1906.09384.

[10] A. Slivkins, *Introduction to Multi-Armed Bandits.* Foundations and Trends in Machine Learning, NOW Publishers, 2019.

[11] T. Lattimore and C. Szepesvári, *Bandit Algorithms.* Cambridge University Press, 2020.

[12] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games.* USA: Cambridge University Press, 2006.

[13] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer, 2009.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016.

[15] W. Zeng and Y. Liu, "Markov decision process modeled with bandits for sequential decision making in linear–flow," *2021 KDD Multi–Armed Bandits and Reinforcement Learning Workshop*, 2021.

[16] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich, "Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine," in *ICML*, Omnipress, 2010.

[17] C. Riquelme, G. Tucker, and J. Snoek, "Deep Bayesian bandits showdown: An empirical comparison of Bayesian deep networks for Thompson sampling," in *ICLR*, 2018.

[18] S. Agrawal and N. Goyal, "Thompson sampling for contextual bandits with linear payoffs," in *ICML*, 2013.

[19] M. M. Drugan and A. Nowe, "Designing multi-objective multi-armed bandits algorithms: A study," *IJCNN*, vol. 3, pp. 1–8, 2013.

[20] A. Badanidiyuru, R. Kleinberg, and A. Slivkins, "Bandits with knapsacks," *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 2013.

[21] W. Ding, T. Qin, X. Zhang, and T. Liu, "Multi-armed bandit with budget constraint and variable costs," in *AAAI*, 2013.

[22] P. Joulani, A. György, and C. Szepesvári, "Online learning under delayed feedback," in *ICML*, 2013.

[23] C. Vernade, A. Carpentier, T. Lattimore, G. Zappella, B. Ermis, and M. Brueckner, "Linear bandits with stochastic delayed feedback," in *ICML*, 2020.

[24] O. Chapelle and L. Li, "An empirical evaluation of Thompson sampling," in *NIPS*, 2011.

[25] L. Yang, M. Hajiesmaili, M. S. Talebi, J. C. S. Lui, and W. S. Wong, "Adversarial bandits with corruptions: Regret lower bound and no-regret algorithm," in *NeurIPS*, 2020.

[26] A. Beygelzimer, J. Langford, L. Li, L. Reyzin, and R. E. Schapire, "Contextual bandit algorithms with supervised learning guarantees," in *AISTATS*, 2011.

[27] S. Kale, L. Reyzin, and R. E. Schapire, "Non-stochastic bandit slate problems," in *NIPS*, 2010.

[28] Y. Yue and T. Joachims, "Beat the mean bandit," in *ICML*, 2011.

[29] Z. Wen, B. Kveton, and A. Ashkan, "Efficient learning in large-scale combinatorial semi-bandits," in *ICML*, 2015.

[30] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims, "The k-armed dueling bandits problem," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1538–1556, 2012.

[31] D. Zhou, L. Li, and Q. Gu, "Neural contextual bandits with UCB-based exploration," in *ICML*, 2020.

[32] B. Ermis, P. Ernst, Y. Stein, and G. Zappella, "Learning to rank in the position based model with bandit feedback," in *CIKM*, 2020.

[33] Y. Wang, H. Ouyang, C. Wang, J. Chen, T. Asamov, and Y. Chang, "Efficient ordered combinatorial semi-bandits for whole-page recommendation," in *AAAI*, 2017.

[34] L. Qin, S. Chen, and X. Zhu, "Contextual combinatorial bandit and its application on diversified online recommendation," in *ICDM*, 2014.

[35] Y. Yue and C. Guestrin, "Linear submodular bandits and their application to diversified retrieval," in *NIPS*, 2011.

[36] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Information-theoretic regret bounds for Gaussian process optimization in the bandit setting," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3250–3265, 2012.

[37] A. Krishnamurthy, J. Langford, A. Slivkins, and C. Zhang, "Contextual bandits with continuous actions: Smoothing, zooming, and adapting," in *COLT*, 2020.

[38] B. Lin, D. Bouneffouf, G. Cecchi, and I. Rish, "Contextual bandit with adaptive feature extraction," *2018 IEEE International Conference on Data Mining Workshops*, 2020.

[39] T. Zahavy and S. Mannor, "Deep neural linear bandits: Overcoming catastrophic forgetting through likelihood matching," 2019. arXiv:1901.08612.

[40] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Q. Candela, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pp. 1–9, 2014.

[41] M. A. Bayir, M. Xu, Y. Zhu, and Y. Shi, "Genie: An open box counterfactual policy estimator for optimizing sponsored search marketplace," in *WSDM*, 2019.

[42] L. Bottou, J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson, "Counterfactual reasoning and learning systems: The example of computational advertising," *Journal of Machine Learning Research*, vol. 14, no. 65, pp. 3207–3260, 2013.

[43] A. Strehl, J. Langford, L. Li, and S. M. Kakade, "Learning from logged implicit exploration data," in *NIPS* (J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, eds.), 2010.

[44] M. Dudik, J. Langford, and L. Li, "Doubly robust policy evaluation and learning," in *ICML*, 2011.

[45] A. Swaminathan and T. Joachims, "The self-normalized estimator for counterfactual learning," in *NIPS*, 2015.

[46] A. Swaminathan, A. Krishnamurthy, A. Agarwal, M. Dudik, J. Langford, D. Jose, and I. Zitouni, "Off-policy evaluation for slate recommendation," in *NIPS*, 2017.

[47] L. Li, W. Chu, J. Langford, and X. Wang, "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms," *WSDM*, 2011.

[48] M. Dudik, D. Erhan, J. Langford, and L. Li, "Sample-efficient nonstationary policy evaluation for contextual bandits," in *UAI*, 2012.

[49] A. Kazerouni and L. M. Wein, "Best arm identification in generalized linear bandits," *Operations Research Letters*, vol. 49, no. 3, pp. 365–371, 2021.

[50] M. Soare, A. Lazaric, and R. Munos, "Best-arm identification in linear bandits," in *NIPS*, 2014.

[51] S. Bubeck, R. Munos, and G. Stoltz, "Pure exploration in finitely-armed and continuous-armed bandits," *Theoretical Computer Science*, vol. 412, no. 19, pp. 1832–1852, 2010.

[52] R. Kohavi, D. Tang, and Y. Xu, *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge University Press, 2020.