AutoOffAB: Toward Automated Offline A/B Testing for Data-Driven Requirement Engineering

Jie JW Wu[*]
University of British Columbia
Kelowna, B.C., Canada
jie.jw.wu@ubc.ca

ABSTRACT

Software companies have widely used online A/B testing to evaluate the impact of a new technology by offering it to groups of users and comparing it against the unmodified product. However, running online A/B testing needs not only efforts in design, implementation, and stakeholders' approval to be served in production but also several weeks to collect the data in iterations. To address these issues, a recently emerging topic, called *offline A/B testing*, is getting increasing attention, intending to conduct the offline evaluation of new technologies by estimating historical logged data. Although this approach is promising due to lower implementation effort, faster turnaround time, and no potential user harm, for it to be effectively prioritized as requirements in practice, several limitations need to be addressed, including its discrepancy with online A/B test results, and lack of systematic updates on varying data and parameters. In response, in this vision paper, I introduce AutoOffAB, an idea to automatically run variants of offline A/B testing against recent logging and update the offline evaluation results, which are used to make decisions on requirements more reliably and systematically.

CCS CONCEPTS

• **Computing methodologies → Learning from implicit feedback**; • **Information systems → Evaluation of retrieval results**.

KEYWORDS

A/B testing, controlled experiments, counterfactual estimation, off-policy evaluation

ACM Reference Format:
Jie JW Wu. 2024. AutoOffAB: Toward Automated Offline A/B Testing for Data-Driven Requirement Engineering. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE Companion '24), July 15–19, 2024, Porto de Galinhas, Brazil.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3663529.3663780

1 INTRODUCTION

Software companies are embracing a data-driven culture and are shifting from traditional requirement-based development to data-driven development and data-driven decision-making [1, 5, 25]. *Online A/B testing* (also known as online controlled experiments, split tests, or randomized experiments) [6, 8, 15, 27] is widely used to collect implicit user behavior and product effect of a given change for online and web-facing products, such as social media [27], search engines [23], social networks [7, 27], and web services [18, 24]. The procedure offers different product variants to different groups of users, then collects data related to the user behavior, and compares the different product variants to the unmodified product [6, 8]. The A/B test allows the gathering of information for a small but significant percentage of users for stakeholders to make decisions on whether to launch a particular variant to 100% of users [15, 27].

However, online A/B test suffers from several limitations. First, it takes significant development efforts to design and implement the change in the code base, with production-level standards. Second, the change will have a real impact on a relatively large group of users in the A/B test to get statistically significant A/B results. So it could affect users negatively if the change in the A/B test includes any bug or safety issue. Thus, domain owners of the products need to sign off for them to be served to a subset of users. Lastly, it typically needs several weeks to run the A/B tests to collect the data with potentially multiple iterations [15]. These limitations dramatically increase the time for the product team to try new ideas.

To address these pain points, a lot of researchers have studied the emerging topic of *offline A/B testing* (or offline policy evaluation, counterfactual evaluation) [9, 10, 14, 19, 20]. The objective of offline A/B testing is to conduct offline evaluation of a new technology by estimating from historical logged data [14]. A number of estimators have been developed such as importance sampling (IS) [16], capped importance sampling (CIS) [2], normalized and capped importance sampling (NCIS) [22] to reach a good balance between bias and variance [3], therefore increasing the correlation between the estimated results in offline A/B tests and the actual results in online A/B tests.

Although offline A/B testing is a promising approach due to much smaller development effort and faster turnaround time, there are still limitations for it to be reliably and effectively used in requirements engineering in practice. The offline A/B testing is a manual process that runs the offline evaluation for manually selected algorithms (or policy [9]) against the one-off historical data. Therefore, there is a lack of systematic updates of offline evaluation on either 1) the updated and chosen historical data or 2) other algorithms that could be more optimal than the manually selected ones.

[*] The author did this work before joining UBC as a postdoc.

FSE Companion '24, July 15–19, 2024, Porto de Galinhas, Brazil

ACM ISBN 979-8-4007-0658-5/24/07...$15.00
https://doi.org/10.1145/3663529.3663780

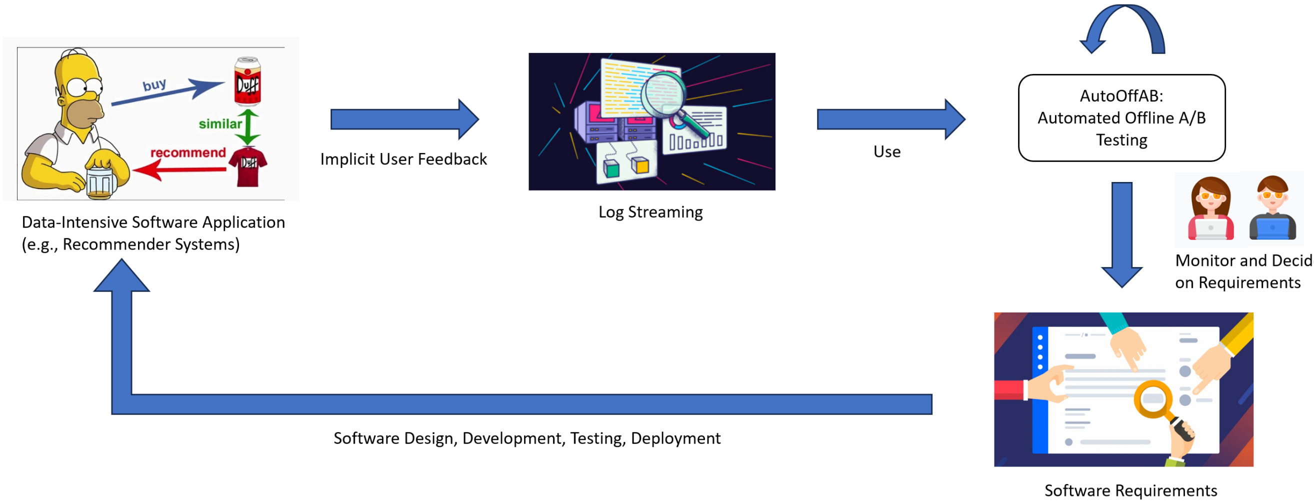

Figure 1: Visual illustration of the proposed AutoOffAB in the context of Data-Driven Requirements Engineering (DDRE) cycle [17]. Without this work, the offline A/B testing needs to be conducted manually by software engineers or ML scientists, which depends heavily on their individual skills. With this work, the offline A/B testing is triggered periodically. Thus, engineers or scientists could focus on monitoring and reviewing the results to be used for decisions on requirements.

This can lead to unreliable offline results and potentially enlarges the discrepancy between offline and online A/B test results.

To address this limitation, in this paper, I introduce AutoOffAB, an idea to automatically generate and periodically update the offline A/B testing evaluation towards more reliable and systematic offline A/B test results for making decisions in requirements engineering. The automation produces offline evaluation as periodic updates from recent historical data rather than one-off historical data. This can prevent the *outdated* evaluation results due to any recent product change. Meanwhile, the periodic automated process also generates results for modified technologies using either randomized genetic algorithms (GA) [12] or potentially more sophisticated methods in the future, rather than a limited set of manually selected technologies in a manual process. I believe that the results from AutoOffAB is more reliable and systematic than the current manual process so that the results and numbers can be more trusted when being prioritized in *Data-Driven Requirements Engineering (DDRE)* [17], as shown in Figure 1. More reliable numbers can also help reduce the gap between the offline A/B testing results and the online A/B testing results, which is a critical criterion for the effectiveness and usefulness of offline A/B testing. In the remaining parts of this paper, I describe the idea of AutoOffAB in detail and discuss possible ways to realize it.

2 OFFLINE A/B TESTING ANALYSIS

2.1 The Current State of Offline A/B Testing

In the current software industry, offline A/B testing is a manual process conducted by software engineers or ML scientists. It has been used in different data-intensive products such as search [14], recommendation [20], ad placement [2, 14], etc. The steps of the offline A/B testing are described as follows. First, the software engineers or ML scientists decide what type of log data to use in the offline evaluation. Second, they select one or a few algorithms to be evaluated. The settings of an algorithm include hyperparameter values, modeling decisions, feature sets, etc. Third, they define the metrics for offline evaluation. Finally, they conduct the evaluation to generate evaluation results for each algorithm against the logged data. Each experimental result corresponds to each algorithm with its setting. Although the offline evaluation significantly reduces the turnaround time of iterating new ideas, it is assumed that the software engineers or ML scientists have full experience with the following questions:

- How many algorithms to evaluate?
- How to determine the settings of these algorithms?
- How to select the historical logs in offline evaluation?
- How to define the cadency of running the evaluation?

However, it appears in some studies that this currently manual process has the following drawbacks:

- It depends solely on the engineers or scientists to decide which setting or parameter values of an algorithm to be used in the offline evaluation. Thus, the choice of variants and their parameter values rely heavily on the skills of engineers, who usually have little assistance or guidance in choosing variants.
- It is often not humanly possible to try all combinations of settings and parameter values to obtain the parameters that lead to the precise optimal result.
- The offline evaluation is a one-off job on certain data from historical logs, but the evaluation results may be inconsistent with the data from different logs (such as most recent data, or shuffled data using certain strategies).

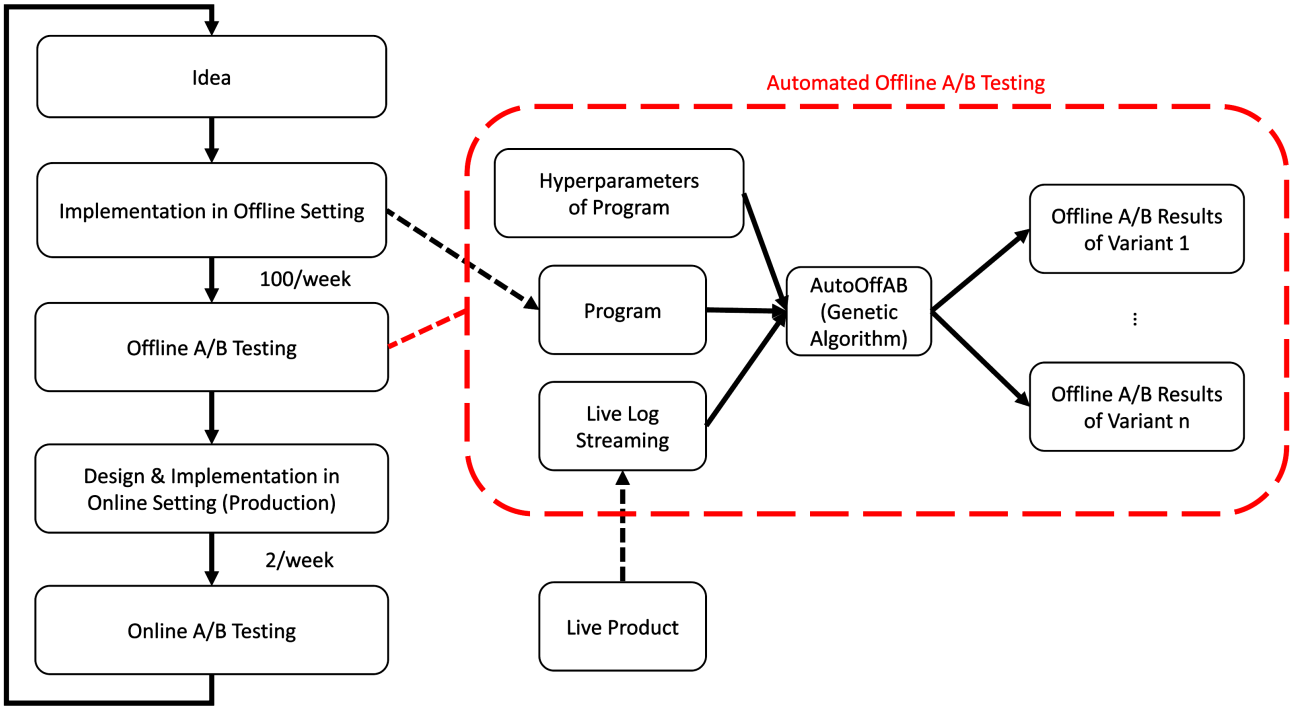
Idea
Implementation in Offline Setting
100/week
Offline A/B Testing
Design & Implementation in Online Setting (Production)
2/week
Online A/B Testing
Automated Offline A/B Testing
Hyperparameters of Program
Program
Live Log Streaming
Live Product
AutoOffAB (Genetic Algorithm)
Offline A/B Results of Variant 1
⋮
Offline A/B Results of Variant n

Figure 2: Visual illustration of AutoOffAB. Overall, AutoOffAB uses the program and log streaming to *periodically* generate a population of variants with *modified settings*, and then evaluates the variants against the *updated chosen logs*.

As an example to illustrate these issues, the offline A/B testing conducted for playlist recommendation [10] used 12 algorithms, with different settings on hyperparameter values, modeling decisions, training data definition, etc. Although the researchers tested 12 different settings extensively, it is still not realistic to try all possible combinations of settings to get the most optimal results. Also, the offline evaluation is a one-off job on the predefined historical logs from systems running in production. However, the product is live and is collecting data every day, with new changes in the product or users as time goes by. So the one-off results of the old historical logs may not hold the same for the recent logs. These issues are not fully explored and addressed.

2.2 Motivation of Automated Offline A/B Testing

My idea is to periodically trigger the offline A/B testing evaluation on the *updated* logs with *modified variants*, instead of manually running it as a one-off job. Intuitively, this can lead to more reliable and more comprehensive offline A/B test results to be prioritized in requirements analysis and specifications. As *data* has become an intrinsic and evolving component in software development, the motivation behind this idea is to replace the manual process with the automated process for offline evaluation based on data of data-intensive applications. This idea addresses the pain points of the current manual process as mentioned above. First, it alleviates the burden of engineers or scientists to choose the settings of the algorithms that may lead to the "best" results based on their educated guesses and skills [21]. In fact, these educated guesses are often inaccurate, as there is evidence pointing out that these intuitions are often wrong and contradict the data from the A/B testing [15]. Second, even if the educated guesses are in the right direction due to the strong skillsets, the machine does a much better job than humans on trying all combinations of settings to find the precise optimal results [11, 26]. Third, reliability can be better ensured by repeating the offline evaluation on logs that are continuously updated and variants with modified settings [13].

3 PROPOSED ARCHITECTURE FOR AUTOMATED OFFLINE A/B TESTING

3.1 Overview

In this paper, I present AutoOffAB to automate the manual procedure of offline A/B testing. Figure 2 provides a visual illustration of AutoOffAB. On the left side, the experiment lifecycle is displayed. As aforementioned, due to the considerable amount of time for collecting data on a large user base and efforts on the design and implementation, only a small number of ideas can be tested in online A/B tests in actual working environments of software companies. Therefore, offline A/B testing can be an area of high Return on Investment (ROI) if the offline A/B results have a strong alignment with online A/B results. Currently, engineers or scientists typically run a one-off offline evaluation on certain historically logged data.

The red dashed box in Figure 2 illustrates how AutoOffAB works. AutoOffAB is based on three components:

- Program,

- Hyperparameters of Program,
- Log Streaming.

The program refers to the implementation of offline A/B testing. Hyperparameters of the program refer to the settings of the program that result in modified technologies such as the model hyperparameters, external and internal parameters, modeling choice, feature set, choice of algorithmic variants, etc. Log streaming is the logs collected by the live product. AutoOffAB uses the program and log streaming to *periodically* generate a population of variants with *modified settings*, and then evaluates the variants against the *updated chosen logs*.

Next, I discuss the following steps of AutoOffAB: 1) Hyperparameter Specification, 2) Algorithm Design of Variants Selection, 3) Evaluation of Variants. 4) Operations and Monitoring of Variants. 5) Decisions on Requirements.

3.2 Hyperparameter Specification

I refer to Hyperparameter Specification as specifying the hyperparameters of the program for the offline A/B evaluation. This includes the specification of all of the hyperparameters that potentially impact the evaluation results, and their valid values. These hyperparameters typically include external and internal parameter values, modeling choices, feature set, training data sources, etc [10]. Formally, the program m of offline A/B testing contains a set of hyperparameters $P = \{p_1, ..., p_n\}$, where each hyperparameter p_i has a corresponding range of valid values R_i. So the goal in this step is to specify $P = \{p_1, ..., p_n\}$ and $R = \{R_1, ..., R_n\}$. Different types of hyperparameters for the program m can be included, such as the one that specifies which data to use from the log streaming.

3.3 Algorithm Design of Variants Selection and Evaluation

After the hyperparameters P and their valid value ranges R are specified, the next step is to generate and select variants $V = \{v_1, ..., v_m\}$ for evaluation. Each variant v_j is an implementation of the program m with its assigned values for hyperparameter $P = \{p_1, ..., p_n\}$. In the manual process, engineers or scientists need to use their experience and skillset to assign the hyperparameter values. In the proposed automated architecture, a genetic algorithm is used to generate a population of variants in each offline evaluation e_k. These variants are then selected and evaluated against the historical data D_{e_k}, based on the pre-defined measurement c for offline evaluation. The measurement is used as a fitness function for the genetic algorithm. So, the objective of the genetic algorithm is to find a variant v that maximizes $c(v, D_{e_k})$, the evaluation measurement of the given variant v and data D_{e_k} from the evaluation e_k. The genetic algorithm is chosen because of its simplicity and wide usage, but any search and optimization technique in Search Based Software Engineering (SBSE) [11] or more sophisticated methods could be used in future work. Note that if the measurement function c has multiple objectives rather than one objective, multi-objective optimization such as Multi-Objective Evolutionary Algorithm (MOEA) [4] may be used.

3.4 Results Monitoring and Decisions on Requirements

The last step of AutoOffAB is related to how to analyze and monitor the periodic, automated offline evaluation results, and how to use the results to make decisions and prioritizations on requirements. First, the continuous and automated evaluation results need to be monitored and analyzed to find out if there are any non-trivial findings that are worth discussing and will potentially impact requirements engineering, such as any change in evaluation results on most recent data, any abnormal results on certain models, etc. The monitoring and analysis are expected to be lightweight, without heavy efforts in data analytics. The stakeholders need to review the evaluation results and make decisions on requirements for which work items need to be prioritized based on the results. With the automated offline A/B testing results, a continuous cycle is formed, known as the Data-Driven Requirement Engineering cycle. As shown in Figure 1, the cycle starts with implicit feedback from users via logging. Then, the automated offline A/B testing is run to generate evaluation results. The results are monitored and analyzed by engineers or scientists to make decisions on requirements. Finally, the derived requirements are executed, developed, and launched in the development process. After the new launches, The cycle starts again with implicit user feedback from updated chosen data.

4 FUTURE PLANS

Public Benchmark and Baseline Methods. As for the future plan, firstly, a public benchmark together with baseline methods and their evaluations need to be created for the task of automating offline A/B testing. The benchmark will enable the comparison between different algorithms and will facilitate the development and evaluation of offline A/B testing. Secondly, I have formulated this problem as an optimization problem, so different optimization methods or search-based methods can be used as baselines for this problem, such as the GA-based methods mentioned in this work.

LLM-based Automation for Offline A/B Testing. Finally, given the recent advances in Natural Language Processing (NLP) and Large Language Models (LLM) on code generation and software engineering tasks, LLM-based approaches should be explored to create more intelligent non-trivial variants for offline A/B testing so that the capability of automated offline evaluation will go beyond changing hyperparameter values with search-based methods only.

5 CONCLUSION

The great potential of offline A/B testing has attracted much interest from both academia and the software industry recently. The beauty of offline A/B testing lies in its support of much faster iteration of trying ideas in practice for many data-intensive ML-enabled applications such as search, recommender systems, and advertising. In this paper, I present AutoOffAB, an idea toward automated offline A/B testing. AutoOffAB automatically runs and periodically updates the offline A/B testing results, which are used to make decisions on requirements for further development. Given the importance of offline A/B testing, I argue that there should be a better presence for Software Engineering research to enable more reliable and systematic offline A/B test results via solutions like AutoOffAB.

REFERENCES

[1] Florian Auer, Rasmus Ros, Lukas Kaltenbrunner, Per Runeson, and Michael Felderer. 2021. Controlled experimentation in continuous experimentation: Knowledge and challenges. *Information and Software Technology* 134 (2021), 106551.

[2] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual Reasoning and Learning Systems: The Example of Computational Advertising. *Journal of Machine Learning Research* 14, 11 (2013).

[3] Xiaocong Chen, Siyu Wang, Julian McAuley, Dietmar Jannach, and Lina Yao. 2023. On the opportunities and challenges of offline reinforcement learning for recommender systems. *ACM Transactions on Information Systems* (2023).

[4] Kalyanmoy Deb. 2011. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*. Springer, 3–34.

[5] Aleksander Fabijan, Pavel Dmitriev, Helena Holmstrom Olsson, and Jan Bosch. 2018. The online controlled experiment lifecycle. *IEEE Software* 37, 2 (2018), 60–67.

[6] Fabian Fagerholm, Alejandro Sanchez Guinea, Hanna Mäenpää, and Jürgen Münch. 2017. The RIGHT model for continuous experimentation. *Journal of Systems and Software* 123 (2017), 292–305.

[7] Dror G Feitelson, Eitan Frachtenberg, and Kent L Beck. 2013. Development and deployment at facebook. *IEEE Internet Computing* 17, 4 (2013), 8–17.

[8] Brian Fitzgerald and Klaas-Jan Stol. 2017. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software* 123 (2017), 176–189.

[9] Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. 2018. Offline a/b testing for recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 198–206.

[10] Alois Gruson, Praveen Chandar, Christophe Charbuillet, James McInerney, Samantha Hansen, Damien Tardieu, and Ben Carterette. 2019. Offline evaluation to make decisions about playlistrecommendation algorithms. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 420–428.

[11] Mark Harman, S Afshin Mansouri, and Yuanyuan Zhang. 2012. Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys (CSUR)* 45, 1 (2012), 1–61.

[12] John H Holland. 1992. Genetic algorithms. *Scientific american* 267, 1 (1992), 66–73.

[13] Yue Jia and Mark Harman. 2010. An analysis and survey of the development of mutation testing. *IEEE transactions on software engineering* 37, 5 (2010), 649–678.

[14] Thorsten Joachims and Adith Swaminathan. 2016. Counterfactual evaluation and learning for search, recommendation and ad placement. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 1199–1201.

[15] Ron Kohavi, Diane Tang, and Ya Xu. 2020. *Trustworthy online controlled experiments: A practical guide to a/b testing*. Cambridge University Press.

[16] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. 2011. 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*. 297–306.

[17] Walid Maalej, Maleknaz Nayebi, Timo Johann, and Guenther Ruhe. 2015. Toward data-driven requirements engineering. *IEEE software* 33, 1 (2015), 48–54.

[18] Niko Pajkovic. 2022. Algorithms and taste-making: Exposing the Netflix Recommender System's operational logics. *Convergence* 28, 1 (2022), 214–235.

[19] Agnė Reklaitė and Jevgenij Gamper. 2022. Offline assessment of interference effects in a series of AB tests. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*. 262–263.

[20] Prabhat Kumar Saraswat, Samuel William, and Eswar Reddy. 2021. A Hybrid Approach for Offline A/B Evaluation for Item Ranking Algorithms in Recommendation Systems. In *Proceedings of the First International Conference on AI-ML Systems*. 1–6.

[21] Sebastian Simon, Nikolay Kolyada, Christopher Akiki, Martin Potthast, Benno Stein, and Norbert Siegmund. 2023. Exploring Hyperparameter Usage and Tuning in Machine Learning Research. In *2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN)*. IEEE, 68–79.

[22] Adith Swaminathan and Thorsten Joachims. 2015. The self-normalized estimator for counterfactual learning. *advances in neural information processing systems* 28 (2015).

[23] Diane Tang, Ashish Agarwal, Deirdre O'Brien, and Mike Meyer. 2010. Overlapping experiment infrastructure: More, better, faster experimentation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 17–26.

[24] Bradley C Turnbull. 2019. Learning Intent to Book Metrics for Airbnb Search. In *The World Wide Web Conference*. 3265–3271.

[25] Jie JW Wu, Thomas A Mazzuchi, and Shahram Sarkani. 2023. Comparison of multi-criteria decision-making methods for online controlled experiments in a launch decision-making framework. *Information and Software Technology* 155 (2023), 107115.

[26] Jie JW Wu, Thomas A Mazzuchi, and Shahram Sarkani. 2023. A multi-objective evolutionary approach towards automated online controlled experiments. *Journal of Systems and Software* 203 (2023), 111703.

[27] Ya Xu, Nanyu Chen, Addrian Fernandez, Omar Sinno, and Anmol Bhasin. 2015. From infrastructure to culture: A/B testing challenges in large scale social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2227–2236.

Received 27-JAN-2024; accepted 2024-04-09