Comparison of Contextual Bandit Approaches for Criteo Ad Placement

Angad Singh, Bharat Jain, Jiya Agrawal, Pratham Arora

Plaksha University

1 Introduction: The Ad Placement Task

Display advertising involves showing relevant ads to users when they visit websites. When a user arrives at a site with an ad slot (an **impression**), a system must choose which ad to display from a set of potential candidates. The primary goal is to select the ad that is most likely to be clicked by the user, thus maximizing the overall **Click-Through Rate (CTR)** across many impressions.

This challenge, based on data provided by **Criteo**, asks participants to develop a **policy** – a decision-making strategy – that takes information about the user/context and the available ad candidates and outputs a score or preference for each candidate. The effectiveness of the policy is measured by how well it performs in terms of generating clicks on a hidden test dataset.

2 Reinforcement Learning Formulation: Contextual Bandits

This ad placement problem can be framed as a Reinforcement Learning (RL) problem, specifically a **Contextual Bandit** problem. Here's why:

- **Context (State):** For each impression, we receive information about the context of the ad slot. This information, along with features describing the candidate ads themselves, forms the "context" or "state". This context is crucial for making an informed decision. This state includes the user context implicitly.
- **Actions:** The available actions are the candidate ads within the pool for that specific impression. The policy must choose one (or rank them).
- **Reward:** The reward is immediate and binary:`1` if the chosen ad is clicked, and `0` otherwise. (The dataset uses `0.001` for click and `0.999` for no-click, but conceptually it's a binary outcome).
- **Episodic Nature:** Each impression is an independent episode. The action taken for one impression DOES NOT influence the context or available actions for the next impression.
- **Goal:** The objective is to learn a policy $\pi(\text{action}|\text{context})$ that maximizes the expected cumulative reward (i.e., the total number of clicks or the overall CTR) over many impressions.

Unlike full RL problems, there's **no long-term sequence of states and actions** within a single episode. The decision is made based only on the current context, making it a contextual bandit setting.

3 Dataset and Preprocessing

The dataset provided for the competition [6] has two parts: **training** and **testing**. The training file is used to learn a policy, which is then evaluated on the test set by submitting predictions to the competition platform. The training dataset contains approximately *10.5 million* impression groups (around 3 GB compressed), each consisting of multiple candidate ads.

Ground truth and propensity scores for the test are not available, and competition submission is not available. To enable evaluation, we created a **80-20 train-validation split** in the training data provided.

Each group of candidates corresponds to a single ad impression and typically looks like the following:

```
17193693 |l 0.999 |p 11.32480 |f 0:300 ...
17193693 |f 0:300 1:250 2:1 10:1 11:1 ...
17193693 |f 0:300 1:250 2:1 12:1 14:1 ...
17193693 |f 0:300 1:250 2:1 12:1 14:1 ...
```

Here:

- `17193693` is the impression ID.
- The first line represents the **logged action** — the ad actually displayed by the logging policy.
- `|l 0.999` is the label indicating the outcome (not clicked). A click is denoted by `0.001`.

- `|p 11.32480` is the inverse propensity score. Since the logging policy was stochastic, the true propensity is $p_{\log} = 1/11.32480$.
- `|f ...` contains the sparse, high-dimensional feature vector in `feature:value` format. Features include both context-only and context-candidate interactions, encoded using one-hot encoding across 74,000 dimensions.

As shown in Figure 1, the number of ad candidates per impression varies significantly, with most impressions having between `10-100` candidates. This distribution insight helped us understand the exploration space for each decision point.

Figure 2 illustrates the distribution of features per ad candidate, showing the dimensionality of the context we're working with in this problem.

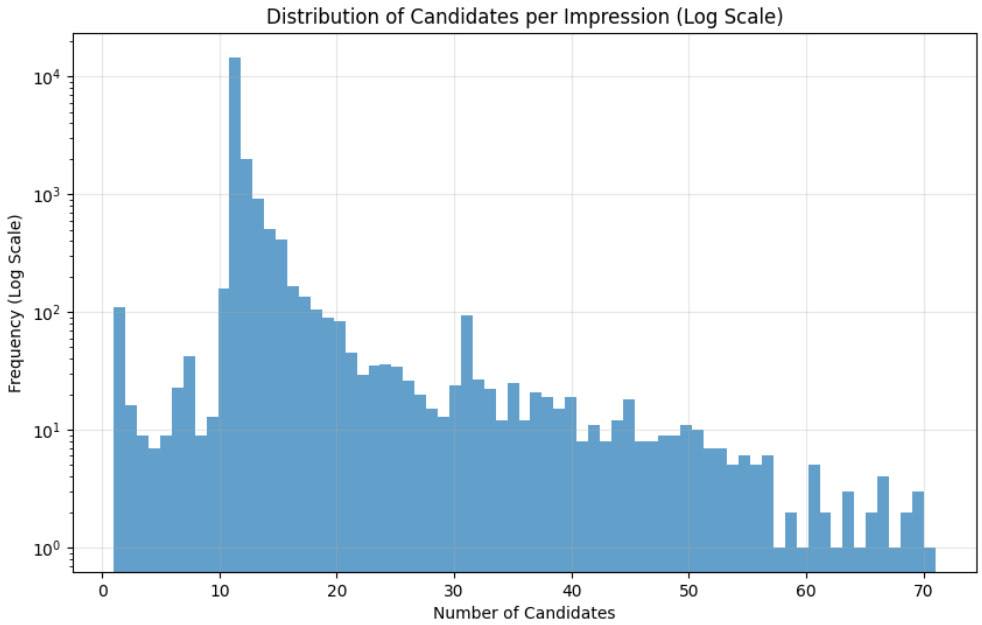

Figure 1: Distribution of candidates per impression (log scale)

4 Evaluation Metrics and Scoring

Since we cannot simply deploy every submitted policy to test it live, we need a way to evaluate how well a new policy (π_{eval}) would perform using only the historical log data collected by a different policy ($\pi_{\log}$). This is called **Off-Policy Evaluation**. The primary metric used is **Inverse Propensity Score (IPS).**

4.1 Inverse Propensity Score (IPS)

The core idea of IPS is to re-weight the observed rewards from the log data to account for the differences between the logging policy and the policy being evaluated. If the evaluated policy was more likely to choose the action that was actually logged (compared to the logging policy), we give the observed reward more weight. If it was less likely, we give it less weight.

The formula for the IPS estimator of the value (expected reward) of the evaluation policy π_{eval} is:

$$V_{\text{IPS}}(\pi_{\text{eval}}) = \frac{1}{N} \sum_{i=1}^{N} \frac{\pi_{\text{eval}}(a_i|x_i)}{\pi_{\log}(a_i|x_i)} R_i$$

where:

- N is the total number of impressions in the dataset.
- x_i is the context for impression i.
- a_i is the action (ad) actually taken/logged for impression i.

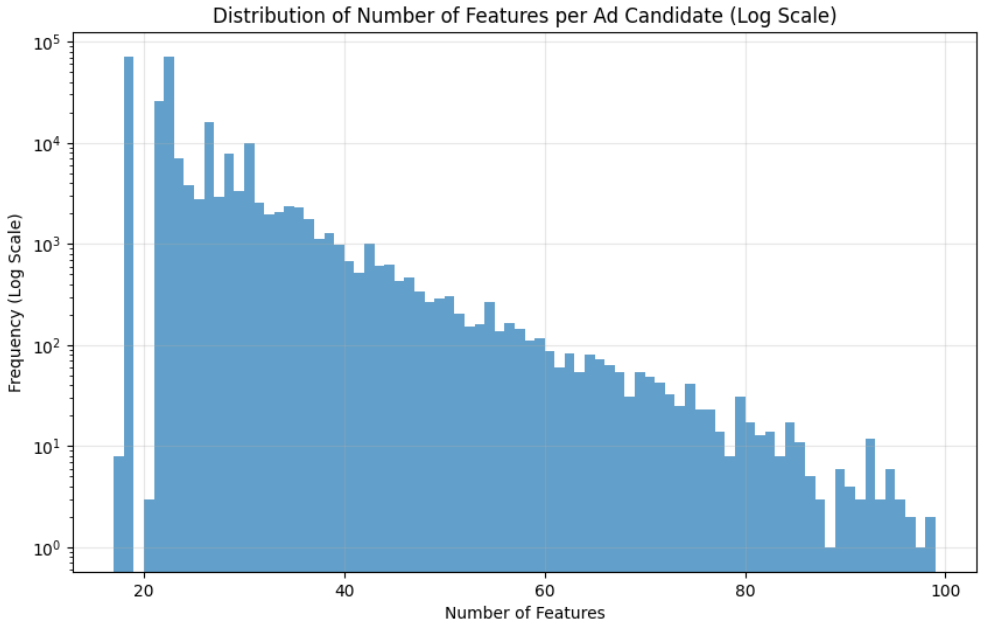

Figure 2: Distribution of features per ad candidate (log scale)

- R_i is the reward (1 for click, 0 for no-click) observed for impression i.
- $\pi_{\text{log}}(a_i|x_i)$ is the probability the logging policy chose action a_i given context x_i (this is the propensity score from the dataset).
- $\pi_{\text{eval}}(a_i|x_i)$ is the probability the policy we are evaluating would have chosen action a_i given context x_i.
- The ratio $\frac{\pi_{\text{eval}}(a_i|x_i)}{\pi_{\text{log}}(a_i|x_i)}$ is the importance weight.

While theoretically unbiased, IPS can suffer from high variance, especially if the propensity scores π_{log} are very small for some actions that π_{eval} might favor.

5 Methodology: Approaches Explored

5.1 Epsilon-Greedy

The Epsilon-Greedy algorithm[7] is a simple yet effective strategy for **balancing exploration and exploitation** in bandit problems. It operates based on a single parameter, ϵ **(epsilon)**, which represents the probability of exploring.

5.1.1 Core Strategy

For each impression (context x), let $A(x)$ be the set of available candidate ads (actions), and let $Q(x, a)$ be the estimated value (e.g., predicted click probability) of choosing action a in context x. Let $a^* = \arg\max_{a \in A(x)} Q(x, a)$ be the action with the highest estimated value (the greedy action).

The Epsilon-Greedy policy π_ϵ selects an action a according to the following probability distribution:

$$\pi_\epsilon(a|x) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(x)|} & \text{if } a = a^* \\ \frac{\epsilon}{|A(x)|} & \text{if } a \neq a^* \end{cases}$$

where $|A(x)|$ is the number of available candidates for the impression.

This means:

- **Exploitation (with probability $1 - \epsilon$):** The policy chooses the greedy action a^*.
- **Exploration (with probability ϵ):** The policy chooses an action uniformly at random from all available candidates in $A(x)$.

Note that the greedy action a^* is also chosen with some probability during the exploration phase ($\frac{\epsilon}{|A(x)|}$), hence its total probability is $1 - \epsilon + \frac{\epsilon}{|A(x)|}$.

5.1.2 Implementation for the dataset

In our implementation for the Criteo dataset, we combined the Epsilon-Greedy strategy with a **Logistic Regression model**:

- **Underlying Model:** A Logistic Regression model was trained to predict the probability of a click given the high-dimensional sparse feature vector of an ad candidate. This model was trained using only the data from the first candidate (the logged action) in the training split, using its features and the observed click outcome (`1` for click, `0` for no-click). Class weighting was used to handle the inherent imbalance between clicks and non-clicks.
- **Exploitation Step:** When exploiting (probability $1-\epsilon$), the trained Logistic Regression model predicts the click probability for all candidates in the current impression. The candidate with the highest predicted probability is selected (or ranked highest).
- **Exploration Step:** When exploring (probability ϵ), one candidate is chosen uniformly at random from the available pool for that impression, regardless of the model's predictions.
- **Prediction Output:** For evaluation using IPS/SNIPS, the policy needs to output scores or probabilities. In the exploitation phase, the raw probabilities from the Logistic Regression model are used. In the exploration phase, to simulate the random choice, the randomly selected candidate is assigned a high score (e.g., 1.0) and all others a low score (e.g., 0.0), ensuring it gets selected by the evaluation mechanism.
- **Hyperparameter Tuning:** The crucial parameter ϵ was tuned by running the entire training and evaluation process for a range of epsilon values (from 0.0 to 0.9). The performance (IPS and SNIPS scores) was measured for each value on a held-out test set. The epsilon value yielding the highest SNIPS score (0.850 in our experiments) was selected as the optimal value for this approach.

This method leverages a standard machine learning model for exploitation while ensuring continued exploration through the simple epsilon mechanism.

5.2 Upper Confidence Bound (UCB)

The Upper Confidence Bound (UCB) [3] algorithm offers a deterministic approach to the exploration-exploitation dilemma, guided by the principle of "optimism in the face of uncertainty". It contrasts with Epsilon-Greedy by employing a **more targeted exploration strategy** rather than relying on purely random choices.

5.2.1 Core Strategy: Optimism and Confidence Intervals

In the context of the Criteo challenge, we adapt UCB by treating each candidate *slot position* (index $j = 0, 1, \dots$) as an independent "arm". The objective for this policy variant is to learn the average click-through rate (CTR) associated with placing an ad at each position, based on historical observations for that position.

To achieve this, UCB maintains two key statistics for each arm j:

- N_j: The number of times arm j has been selected (pulled).
- $\hat{\mu}_j$: The empirical mean reward (average CTR) observed from arm j after N_j pulls. This represents the current best estimate of the arm's value (the exploitation component).

At each time step T (total impressions processed), the algorithm considers both the estimated value $\hat{\mu}_j$ and an exploration bonus for each arm. This bonus term quantifies the uncertainty surrounding the estimate $\hat{\mu}_j$. It is designed to be larger for arms that have been pulled less frequently (N_j is small), reflecting higher uncertainty about their true potential. Mathematically, this bonus typically depends on $\log(T)$ and N_j, ensuring it decreases as confidence grows (N_j increases) but also guarantees that all arms are eventually revisited ($\log(T)$ increases).

The UCB policy then selects the arm j that offers the most promising combination of high estimated reward ($\hat{\mu}_j$) and high uncertainty (large exploration bonus). This optimistic selection ensures that the algorithm explores arms that might be underestimated due to limited sampling, while still favoring arms that consistently perform well. The hyperparameter $c \geq 0$ tunes the balance, controlling how strongly the uncertainty influences the decision towards exploration.

5.2.2 Application and Tuning

For implementation, the training phase involves processing the logged data to update the reward tallies and pull counts (N_j) for each position index, along with the total impression count T. During prediction, these learned statistics are used to apply the UCB selection principle, determining which position is favored based on its estimated value and exploration potential. The

exploration tendency is governed by the hyperparameter c, which was tuned by evaluating performance (using SNIPS) on a validation set across different c values. While this positional UCB simplifies the contextual nature of the problem, it establishes a robust baseline founded on principled exploration mechanics.

5.3 Thompson Sampling

5.3.1 Core Algorithm and Arms

Thompson Sampling[1] is a Bayesian approach to the exploration-exploitation dilemma in contextual bandits. In the Criteo ad placement problem, the "arms" of the bandit are the candidate ads available for display in a given impression. Each arm has contextual features and an unknown click probability.

The algorithm maintains a posterior distribution over model parameters and makes decisions by:

- Sampling parameters from the posterior distribution
- Using these parameters to score each arm
- Selecting the arm with the highest score
- Updating the posterior after observing the outcome

This sampling-based approach naturally balances exploration and exploitation as the posterior becomes more concentrated around true parameters over time.

5.3.2 Implementation

For the Criteo ad placement task, we implemented Thompson Sampling with Bayesian Logistic Regression. The key components include:

- Model Structure: The click probability is modeled as $P(\text{click}|\mathbf{x}, \boldsymbol{\theta}) = \sigma(\mathbf{x}^T\boldsymbol{\theta})$, where σ is the sigmoid function, $\mathbf{x}$ is the feature vector, and $\boldsymbol{\theta}$ is the parameter vector.
- Posterior Approximation: We maintain a diagonal Gaussian posterior $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$ for computational efficiency.
- Decision Process: For each impression, we: - Sample parameters $\hat{\boldsymbol{\theta}}$ from the current posterior - Score each arm using $\sigma(\mathbf{x}^T\hat{\boldsymbol{\theta}})$ - Select the arm with the highest score
- Training: The model is trained offline on historical impressions, updating the posterior parameters incrementally with each observed impression and reward.

This approach provides an effective balance between exploring new ads and exploiting known high-performers, making it well-suited for maximizing click-through rates in diverse contextual settings.

5.4 Actor-Critic Methods

Actor-Critic[5] methods combine policy-based and value-based reinforcement learning approaches. For the Criteo ad placement challenge, we implemented a neural network architecture with:

- **Shared Feature Processing:** Multiple fully connected layers transform sparse Criteo features into dense representations
- **Actor Network:** Outputs a probability distribution $\pi(a|x;\theta)$ over candidate ads given context x
- **Critic Network:** Estimates the expected reward $V(x;w)$ for a given context

This dual-network approach allows simultaneous policy improvement and value estimation, making it well-suited for the ad placement optimization problem where balancing exploration with exploitation is critical.

5.4.1 Implementation

- **Off-Policy Correction:** Inverse Propensity Scoring (IPS) adjusts rewards ($R'_i = R_i/p_i$) to account for the discrepancy between logging and current policies
- **Advantage Calculation:**
$$A(x_i, a_i) \approx R'_i - V(x_i; w)$$
measures how much better actions performed than expected
- **Training Process:** - Critic updates minimize the error between predicted values and observed corrected rewards - Actor updates follow policy gradients scaled by advantage estimates - Entropy regularization prevents premature convergence to deterministic policies
- Combined Loss Function:
$$L = L_{\text{actor}} + c_1 L_{\text{critic}} - c_2 L_{\text{entropy}}$$
optimized via gradient descent

5.4.2 Evaluation

Once trained, the model processes new impression contexts to compute probabilities across all candidate ads. These probabilities serve as selection scores and are evaluated using off-policy metrics (IPS/SNIPS). This approach effectively addresses the counterfactual nature of the Criteo challenge by learning from historical data while accounting for the exploration-exploitation tradeoff inherent in ad placement optimization. The Actor-Critic framework provides a principled way to maximize expected click-through rates in this complex contextual decision problem.

5.5 SoftMax Exploration for Ad Placement

5.5.1 Model Overview

The SoftMax[2] policy addresses the exploration-exploitation trade-off in ad placement by modeling each candidate position as an arm in a multi-armed bandit framework. The algorithm maintains empirical estimates of expected rewards (click-through rates) and employs the softmax transformation:

$$\pi_t(a_i) = \frac{\exp(Q_t(a_i)/\tau)}{\sum_{j=1}^{n} \exp(Q_t(a_j)/\tau)} \tag{1}$$

where $\pi_t(a_i)$ is the selection probability for arm i at time t, $Q_t(a_i)$ represents the empirical mean reward, τ is the temperature parameter, and n is the number of arms.

5.5.2 Learning Methodology

The implementation learns from logged data by:

1. Identifying the logged action in each impression
2. Recording the observed reward (1 for click, 0 for no-click)
3. Updating arm statistics: $n_{\text{pulls}}[a] \leftarrow n_{\text{pulls}}[a] + 1$ and $\text{sum\_rewards}[a] \leftarrow \text{sum\_rewards}[a] + r$

These updates enable the calculation of $Q_t(a_i) = \frac{\text{sum\_rewards}[i]}{n_{\text{pulls}}[i]}$ for each arm.

5.5.3 Implementation

Several technical aspects were addressed:

Temperature Parameter: A value of $\tau = 0.1$ provided an effective exploration-exploitation balance for the Criteo dataset based on reward sparsity and dataset characteristics.

Cold Start: For arms with insufficient historical data, we apply a default reward value calculated as the global CTR across the training dataset.

Numerical Stability: To prevent overflow/underflow, we normalize Q-values by subtracting the maximum before exponential transformation:

$$\text{scores} = \exp\left(\frac{Q_t(a_i) - \max_j Q_t(a_j)}{\tau}\right) \tag{2}$$

6 Results and Analysis

6.1 Evaluation Results

In this section, we present the **performance evaluation** of our various contextual bandit approaches on the Criteo Ad Placement dataset. Since we do not have access to the ground truth of the official test set, the following results are computed using our train-validation split of the original training dataset.

6.1.1 Performance Comparison

Table 1 shows the Inverse Propensity Score (IPS) achieved by each of our methods, alongside the results of the top performers from the NIPS'17 Workshop Criteo Ad Placement Challenge.

Table 1: Performance Comparison of Different Approaches

Method	IPS Score	Standard Deviation
Our Implemented Methods		
SoftMax ($\tau = 0.1$)	46.7	3.72
UCB ($c = 0.2$)	47.22	3.79
Thompson Sampling	46.7	3.72
Epsilon-Greedy ($\epsilon = 0.85$)	39.92	18.31
Actor-Critic	79.36	28.9
Top Competition Performers		
FTRL-Proximal (ololo)	55.6	4.1
geffy	54.6	1.9
Group	54.3	1.6
atsky	54.1	3.0
mortiarty	48.6	1.2

6.2 Discussion

From our experiments and the competition results, several insights emerge:
Note: The **Epsilon-Greedy** and **Actor-Critic** algorithms were evaluated only on a smaller subset of the dataset. This was due to our use of logistic regression in Epsilon-Greedy and due to the neural nature of Actor-Critic, which required computational resources beyond the available scope for full-scale training on the entire dataset.

6.2.1 Method Comparison

Among our implemented approaches, the **UCB** exploration strategy achieved the highest IPS score of **47.22** (±3.79), followed closely by both **SoftMax** and **Thompson Sampling**, each with an IPS of **46.7** (±3.72). This suggests that temperature-based and posterior-sampling-based exploration strategies provided a competitive balance between exploration and exploitation for this particular dataset, though UCB slightly outperformed them. The **Epsilon-Greedy** method lagged behind with an IPS score of **39.92** (±18.31), likely due to its simplistic exploration mechanism and the fact that it was evaluated only on a smaller subset of the dataset due to computational constraints with logistic regression. Interestingly, the **Actor-Critic** method reported the highest IPS score of **79.36** (±28.9), but this result should be interpreted cautiously since it was also evaluated only on a limited subset of the data. Its high variance and restricted evaluation suggest that the score may not generalize to the full dataset. This limitation stems from the computational overhead of training neural networks on sparse, high-dimensional data in an off-policy setting.

6.2.2 Comparison with Competition Winners

While our best approach (UCB) performed competitively with the fifth-place competition entry (mortiarty), there remains a significant gap between our methods and the top-performing solutions. The winning team (ololo) [4] achieved an IPS of 55.6 using FTRL-Proximal, which is approximately 16% higher than our best method.

It's worth noting that the winning solution employed FTRL-Proximal, an online linear classification algorithm specifically designed for large-scale sparse problems like ad click prediction. The effectiveness of this approach highlights the importance of:

- Efficient handling of extremely sparse, high-dimensional feature spaces
- Appropriate regularization strategies to prevent overfitting
- Post-processing techniques that account for the behavior of the IPS metric

References

[1] Shipra Agrawal and Navin Goyal. "Analysis of Thompson Sampling for the Multi-armed Bandit Problem". In: *Proceedings of the 25th Annual Conference on Learning Theory*. Ed. by Shie Mannor, Nathan Srebro, and Robert C. Williamson. Vol. 23. Proceedings of Machine Learning Research. Edinburgh, Scotland: PMLR, 2012, pp. 39.1–39.26. URL: `https://proceedings.mlr.press/v23/agrawal12.html`.

[2] Bolin Gao and Lacra Pavel. "On the properties of the softmax function with application in game theory and reinforcement learning". In: *arXiv preprint arXiv:1704.00805* (2017).

[3] Aurélien Garivier and Eric Moulines. "On Upper-Confidence Bound Policies for Switching Bandit Problems". In: *Algorithmic Learning Theory*. Ed. by Jyrki Kivinen et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 174–188. ISBN: 978-3-642-24412-4.

[4] Alexey Grigorev. *Approaching the Ad Placement Problem with Online Linear Classification: The winning solution to the NIPS'17 Ad Placement Challenge*. 2017. DOI: 10.48550/ARXIV.1712.01913. URL: https://arxiv.org/abs/1712.01913.

[5] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1861–1870. URL: https://proceedings.mlr.press/v80/haarnoja18b.html.

[6] Damien Lefortier et al. *Large-scale Validation of Counterfactual Learning Methods: A Test-Bed*. 2016. DOI: 10.48550/ARXIV.1612.00367. URL: https://arxiv.org/abs/1612.00367.

[7] X Yang and Z Ji. "Accelerating Multi-step Sparse Reward Reinforcement Learning". In: *Proceedings of the Cardiff University School of Engineering Research Conference 2023*. Cardiff University Press, 2024, pp. 86–90. URL: http://www.jstor.org/stable/jj.15454013.23 (visited on 04/30/2025).