A Review of Reinforcement Learning Applications for Monetization-Aware Content Ranking in Large-Scale Recommendation Systems

ARMANDO ORDORICA, University of Toronto, Canada
YURI LAWRYSHYN, University of Toronto, Canada

[Test] Balancing advertisements with organic content in large-scale recommendation engines is a well-known challenge in the industry. While ads are essential for revenue generation in platforms that integrate them alongside organic content, they often underperform in user engagement compared to organic content. The key challenges in achieving this balance include ad selection, ranking, and integration with organic content. These involve decisions about whether to insert an ad, which ad to select, and where to place it.

Traditional methods optimize the probability of immediate ad clickthrough with fixed placements and are typically framed as supervised learning problems. However, these approaches estimate conditional expectations of immediate outcomes under a historical logging policy, treating decisions as independent and optimizing short-horizon objectives. A more principled formulation that accounts for contextual information and policy-induced state evolution, and that optimizes long-horizon value over action sequences to jointly balance engagement and revenue, remains an open research challenge.

Reinforcement Learning (RL) provides a principled framework for modeling recommendation as a sequential decision process under uncertainty. However, implementing RL in large-scale recommendation systems faces challenges including high-dimensional state and action spaces, the risk of poorly chosen reward functions and unsafe exploration, high computational costs, and the difficulty of interpreting online policy learning.

This survey synthesizes RL methodologies and frameworks for jointly optimizing ads and organic content, organizing prior approaches according to how they model state, action, reward, and policy. Our review indicates that while fully online RL is theoretically appealing, documented industrial systems more frequently adopt hybrid architectures that leverage supervised learning for robust short-term estimation while incorporating RL components for long-horizon value optimization. Furthermore, many large-scale deployments rely on constrained RL formulations, such as contextual bandits and offline replay, that balance sequential optimization with practical considerations of scalability, safety, and interpretability.

Additional Key Words and Phrases: Reinforcement Learning, Ad Policy Optimization, Contextual Bandits, Recommendation Systems, Off-Policy Learning, Online Advertising, Deep Reinforcement Learning, Multi-Objective Optimization

Compiled: Saturday 9th May, 2026 at 16:29

1 Introduction

Platforms such as Instagram, TikTok, and Pinterest develop and deploy large-scale content recommendation systems whose revenue models depend on jointly optimizing monetization and long-term engagement in interactive environments [Danescu-Niculescu-Mizil et al. 2010; Zhao et al. 2020].

Failing to achieve an adequate balance can result in issues such as ad fatigue, where users become less responsive to ads due to overexposure, and user churn, where users discontinue using the platform [Sagtani et al. 2023; Silberstein et al. 2023].

Authors' Contact Information: Armando Ordorica, University of Toronto, Toronto, Canada; Yuri Lawryshyn, University of Toronto, Toronto, Canada.

2025. ACM 1557-7341/2025/5-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Jointly optimizing revenue and long-term engagement is structurally difficult for several fundamental reasons. First, the monetization and engagement objectives can conflict: actions that increase immediate monetization, such as higher ad load, may degrade future engagement and retention, ultimately reducing long-term revenue [Sagtani et al. 2023; Yan et al. 2020]. Second, immediate per-interaction signals are directly observable and easier to optimize, whereas long-term outcomes unfold over extended horizons and are harder to measure and attribute to specific actions [Hohnhold et al. 2015b; McDonald et al. 2023b; Wang et al. 2022]. Third, recommendation decisions alter user state dynamics, creating feedback loops between engagement and future monetization capacity [Ie et al. 2019c; Zou et al. 2019]. These properties make the trade-off inherently sequential and policy-dependent.

To address these challenges, large-scale platforms have relied on supervised deep learning models optimized for short-term engagement signals such as clicks, dwell time, and conversions [Cheng et al. 2016; Covington et al. 2016; McMahan et al. 2013; Yi et al. 2014]. By learning rich representations of users, items, and context from large-scale interaction logs, these models estimate conditional expectations of immediate outcomes and use these predictions to guide content selection and ranking [Cheng et al. 2016; Covington et al. 2016]. This paradigm has delivered substantial improvements in short-term engagement and click-through prediction performance [Cheng et al. 2016; Covington et al. 2016; McMahan et al. 2013]. It remains central to modern industrial recommender systems.

While neural networks remain powerful function approximators, optimizing supervised objectives on historical logged data does not directly address the sequential and interactive nature of the system. In such settings, SL is structurally insufficient for two key reasons. First, SL ignores policy-induced distribution shifts. It optimizes conditional expectations of immediate outcomes (e.g., $\mathbb{E}[r_t \mid s_t, a_t]$) using data generated under a historical policy. This implicitly assumes that the training distribution is fixed. In interactive systems, however, the recommendation policy shapes user behavior, and user behavior in turn determines future training data. This violates the stationarity assumptions underlying standard supervised optimization [Dudík et al. 2011; Swaminathan and Joachims 2015]. Second, SL lacks a principled mechanism for evaluating counterfactual trajectories as it relies exclusively on realized trajectories. Because user journeys are highly personalized and path-dependent, most possible interaction sequences are never observed in logged data. Long-term outcomes are therefore not fixed labels, but functions of the policy that generates future trajectories [Agarwal and Chen 2016].

RL provides a framework in recommendation systems to solve the two problems above [Afsar et al. 2022]. First, RL addresses policy-induced distribution shifts by explicitly modeling recommendation as a Markov Decision Process (MDP) in which actions influence future states [Puterman 2014; Sutton and Barto 2018] . Second, RL enables policy evaluation and improvement over complete trajectories through optimization of the expected cumulative return under policy-induced dynamics [Sutton and Barto 2018]. Rather than replacing SL, RL can be viewed as embedding supervised prediction within a sequential control framework [Afsar et al. 2022; Sutton and Barto 2018]. Deep models remain central for representation learning and reward estimation, while RL augments them with the machinery required to optimize policy-dependent, long-term objectives [Chen et al. 2022; Dulac-Arnold et al. 2019; Levine et al. 2020].

Despite the promising theoretical framework that RL offers for optimizing long-term value in recommendation systems, implementing RL in real-world, large-scale platforms remains complex [Dulac-Arnold et al. 2021]. First, the state space, which reflects the current context of the user and the environment, and the action space, which captures how the recommender system can be tuned, are both essentially infinite [Zhu et al. 2021]. This vastness makes learning and optimization computationally intensive and requires scalable approximation methods, as exact dynamic

programming over such state and action spaces is infeasible in practice [Afsar et al. 2022]. Second, designing appropriate reward functions is critical, as poorly chosen reward functions can lead to unintended system behaviors that degrade user experience or negatively affect business revenue [Everitt et al. 2021]. For example, a reward function that overly prioritizes click-based metrics may drive the system to promote clickbait content, boosting short-term engagement but eroding user trust and satisfaction over time [Wu et al. 2017]. Conversely, a reward function too heavily skewed toward revenue metrics could result in excessive ad load, degrading the user experience and accelerating churn. Striking the right balance in reward function design is essential to achieve both user engagement and monetization objectives without compromising the platform's long-term goals. Furthermore, interpretability is a major challenge in RL deployment. Understanding the learned policies can be particularly difficult when policies are updated in real-time during user interactions (online policy learning) [Dulac-Arnold et al. 2019; Glanois et al. 2024; Vouros 2022]. In extreme cases, this lack of transparency could result in non-compliant or unethical behavior, posing significant risks to both users and the business [Lambert et al. 2023].

In light of the challenges outlined above, this paper provides a synthesis of RL approaches to monetization-aware recommendation systems through the lens of the core components of a Markov Decision Process. Although previous surveys address RL in recommendation systems [Afsar et al. 2022; Lin et al. 2023], our systematic literature search (see Section 2) did not identify any survey that explicitly focuses on the **joint optimization of ads and organic content in sequential and interactive environments**, which leaves limited guidance for systems that must jointly reason about monetization and long-term user value. This paper addresses this limitation by reviewing approaches that move beyond supervised objectives defined under fixed logging policies to analyze how RL agents model and optimize sequential trade-offs between revenue and engagement.

This survey introduces component-level taxonomy that clarifies how existing systems operationalize state, action, reward, and policy, and uses it to systematically organize and analyze more than 150 academic and industrial works.

The paper begins with a historical overview of recommendation systems with monetization components, tracing the evolution from rule-based and heuristic approaches to tree-based models and modern deep learning architectures. This progression highlights the limitations of short-horizon optimization and motivates reinforcement learning as a framework for modeling long-horizon, policy-dependent objectives. The remainder of the paper is structured around these components (state, action, reward, and policy) using them as analytical categories to compare and contrast existing systems. Rather than structuring prior work chronologically or by algorithmic family, we compare systems according to how they define and operationalize these components. A minimal RL environment (FrozenLake) is used as a conceptual scaffold to introduce the abstraction before mapping it to large-scale monetization-aware recommender systems.

2 Paper Collection Methodology

To ensure broad coverage, relevant papers were collected from both academic and industry sources. Foundational academic works such as *Reinforcement Learning: An Introduction* [Sutton and Barto 2018], *Statistical Methods for Recommender Systems* [Agarwal and Chen 2016], and *Artificial Intelligence: A Modern Approach* [Russell and Norvig 2016] were used to establish key mathematical foundations, including Markov Decision Processes (MDPs), Contextual Bandits, Off-Policy Learning, and Offline Replay.

To reflect the internal perspective of a production-facing ad policy team, a curated list of 10–15 papers referenced within Pinterest's Ad Policy team was also compiled. Examples include *Jointly Learning to Recommend and Advertise* [Zhao et al. 2020], *DEAR: Deep Reinforcement Learning for Online Advertising Impressions in Recommender Systems* [Zhao et al. 2021], *Ad-load Balancing via*

Off-Policy Learning [Sagtani et al. 2024], and *Practical Bandits* [van den Akker et al. 2024]. In addition, targeted keyword searches such as "Reinforcement Learning ads [company name]" and "Contextual Bandit [company name]" were conducted for leading companies in the advertising and recommendation space (e.g., Google, Meta, Pinterest, Microsoft, TikTok, YouTube, LinkedIn), surfacing additional papers describing real-world systems and methodologies. To complement those efforts, a separate search was conducted on Google Scholar for relevant review papers. While no surveys were found that specifically focused on RL in ad recommendation systems, related reviews on "contextual bandits", "ad fatigue", and "recommender systems" were used to identify additional references.

Finally, manual crawling of citation networks in cornerstone papers was used to identify additional key contributions. For instance, tracing citations from *Jointly Learning to Recommend and Advertise* [Zhao et al. 2020], *DEAR* [Zhao et al. 2021], and *Offline Reinforcement Learning* [Levine et al. 2020] led to works such as *Ads Allocation in Feed via Constrained Optimization* [Yan et al. 2020] and *A Contextual Bandit Bake-off* [Bietti et al. 2021]. In total, 153 papers were selected and reviewed. Papers were ranked according to three main criteria: influence, to capture the foundational or widely adopted ideas in the field; industry relevance, to ensure that the outcomes were sound, implementable, and actionable; and recency, to reflect the latest methodological developments and deployment practices.

While recent and highly cited work was prioritized, papers introducing novel methodologies were also included regardless of citation count. To contextualize the selected works, this survey builds from the ground up. Each component of the Markov Decision Process, including states, actions, rewards, and policies, is defined and illustrated using a toy example such as the Frozen Lake. This foundation is then extended to practice through a review of how each component is operationalized in industry, along with a discussion of the limitations of those approaches. Drawing on existing methods and lessons from industry, the resulting framework is intended to inform practitioners exploring how RL can be applied in their organizations to develop ad policies that balance monetization with organic engagement. A comparison between the toy example and real-world implementation parallels can be found in Table 1.

3 A Brief History of Advertising and Recommendation Systems: From Rule-Based to Deep Learning-Based Joint Optimization

This section traces the evolution of advertising recommendation algorithms, culminating in the adoption of RL to overcome the limitations of earlier approaches. It begins by outlining how early online advertising primarily mirrored offline methods through banner ads and other static formats. It continues by explaining how static online advertising evolved over time to incorporate rule-based models and more sophisticated heuristic techniques. The discussion then transitions to the integration of machine learning methods, including regression-based models, tree-based algorithms, and collaborative filtering, followed by the emergence of deep learning and embedding-based approaches that effectively capture the nonlinear interactions between users and content. Finally, RL is introduced as a promising solution to address the shortcomings of preceding methodologies. This evolution is visually represented in Figure 1, which shows the initial adoption and popularization of techniques and methodologies in online advertising.

Table 2 summarizes three distinct but interrelated evolutionary tracks in ad optimization: (1) pricing and objectives, (2) representation, and (3) sequential horizon of decision-making. These dimensions evolved somewhat in parallel rather than strictly sequentially, and modern systems typically combine advances from all three tracks. This framework helps practitioners identify which

MDP Component	FrozenLake Example	Ad Recommendations
Environment	The frozen lake.	The users.
Goal	Reach the goal (G) from the start position (S) while avoiding holes (H). The agent must avoid falling into a hole, and reaching the goal faster is rewarded.	Maximize revenue without compromising user experience.
Episode	A game.	A time period (T), typically 2 weeks, corresponding to an A/B experiment where enough data is collected to evaluate policy performance across users.
Rewards	Rewarded for reaching the goal (i.e., +100), but penalized for delays (i.e., -1 per step).	The reward is typically represented by a utility function, which could be a weighted sum of metrics like user engagement (clicks, time spent) and revenue generated (ad clicks, conversions). The challenge lies in balancing short-term revenue with long-term user retention.
Agent	The learner.	The Recommendation System.
Actions	Up, down, left and right.	Basically infinite. But when it comes to ad recommendations specifically, some typical actions are the following: • Reserve Price Floor: The minimum price at which an ad impression is sold. • Displacement Cost Threshold: A measure of the opportunity cost for showing an ad. • Max Ad Load: The maximum number of ads that can be shown without reducing user satisfaction. • Ad Quality Thresholds: Standards used to ensure ads meet a quality bar before being shown. • Ad Placement: The specific position of ads within the user feed (e.g., first, third, or fifth position).
States	All possible grid positions the agent can occupy, representing the state space of the environment.	Signals: • User Context: Demographics, user profile • Interaction Data: Previous interactions with ads, organic content • Contextual Features: Time of day, browsing context • Embeddings: Latent feature representations of user interests or ad features
Policy	A mapping of grid positions to optimal action to take for a maximum reward.	A mapping of user segments to ad treatments to maximize both revenue and user experience. This policy can be dynamically adjusted based on real-time feedback or offline simulations.

Table 1. Comparison of Frozen Lake Example and Large-Scale Recommendation Systems for Ad Policy Optimization

dimension requires improvement for their specific business context. Additionally, Table 3 describes complementary evaluation methods used to validate and infer policy performance.

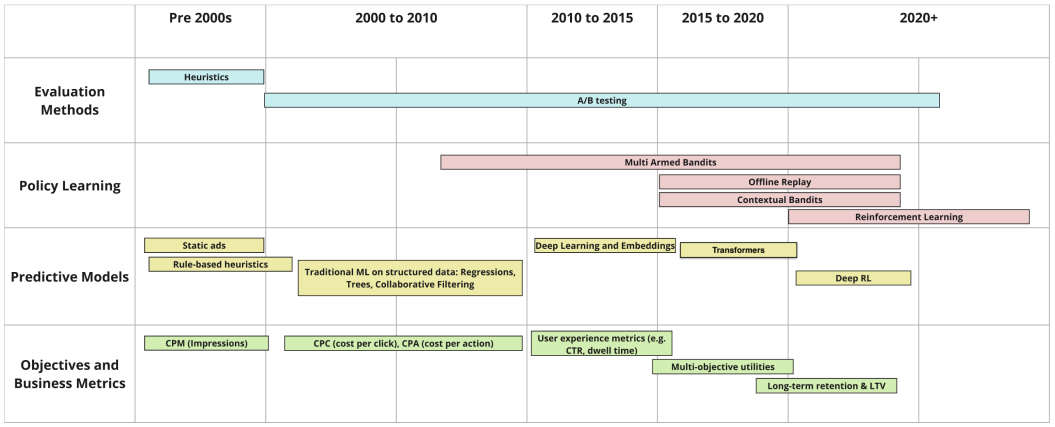

Fig. 1. Timeline Showing Initial Adoption and Popularization of Techniques and Methodologies in Online Advertising

Note: The boundaries and lengths of rectangles are approximate. Innovations and transitions between techniques and objectives occurred gradually and continuously, rather than sharply at specific dates.

Method	Objective	Use Case
A/B Testing	Validate experiments; causal attribution	New model launches; feature comparisons; statistical significance required
Offline Replay	Infer policy performance from logs; counterfactual estimation	Policy evaluation before deployment; large parameter search; risk-averse exploration

Table 3. Complementary Evaluation Methods. A/B testing validates experiments online, while Offline Replay enables inference on logged data. Both can be used alongside any decision-making approach in Track 3.

3.1 From Rule-Based Advertising to Traditional Machine Learning

Around the early 2000s, online advertising largely mirrored offline practices, heavily relying on non-targeted formats such as banner ads [Kant 2021]. These generic advertising methods conflicted with the objective of search engines of efficiently directing users to relevant content [Fain and Pedersen 2006]. As a marginal improvement on nontargeted ad formats like banner ads, advertisers began to utilize simple `if-then` logic with predefined rules to select and display ads based on basic criteria like user demographics. While rule-based systems were straightforward to implement, their lack of flexibility and personalization became evident as ad fatigue, defined as reduced user engagement due to repeated exposure to the same advertisement, persisted [Abrams and Vee 2007]. This issue underscored the necessity of not only leveraging user signals but also optimizing ad load tuning parameters, such as frequency and placement, to maintain user interest and maximize click-through rates [Abrams and Vee 2007]. For instance, researchers at LinkedIn introduced concepts like top slot - highest eligible position for an ad within the feed - and minimum gap - the required distance between consecutive ads to prevent oversaturation - as ad load tuning parameters [Yan et al. 2020]. Their research indicated that users are more likely to click on ads that are spaced farther apart [Yan et al. 2020].

Track 1: Evolution of Pricing & Objectives

Stage	What It Optimizes	Shortcomings	When Sufficient
CPM (Cost per Mille)	Impressions; brand exposure	No incentive for relevance; misaligned with user value	Brand awareness; guaranteed reach campaigns
CPC (Cost per Click)	Clicks; measurable engagement	Incentivizes clickbait; ignores post-click behavior	Direct response; traffic-focused goals
CPA (Cost per Action)	Conversions; purchases, sign-ups	Sparse signal; attribution challenges	E-commerce; clear conversion funnels
Utility Functions	Multi-objective (revenue + engagement + fatigue)	Weight tuning is subjective; hard to validate offline	When explicit trade-offs between objectives are required

Track 2: Evolution of Representation

Stage	How Users/Items Are Encoded	Shortcomings	When Sufficient
Demographics / Rules	Hand-crafted segments (age, gender, location)	Coarse; no personalization; ad fatigue	Small catalogs; regulatory constraints on data use
Tabular Features	Engineered features (click history, session stats)	Manual feature engineering; misses nonlinear patterns	Interpretable models required; limited compute
Embeddings (Two-tower)	Dense vectors capturing user-item similarity	Pointwise; doesn't model sequences or context shifts	Large-scale retrieval; content with rich metadata
Attention / Transformers	Contextual, sequential representations; multi-head	Complex tuning; heads vary across users/time	Sequential sessions; long-range dependencies matter

Track 3: Evolution of Sequential Decision-Making

Stage	Sequence Consideration	Shortcomings	When Sufficient
Rule-based	None; static if-then logic	No learning; manual updates; ignores user behavior	Low traffic; strict compliance; simple business rules
Supervised ML (LR, GBM, DL)	Pointwise; each decision independent	Ignores sequential dependencies; no exploration	When decisions are approximately independent
Multi-Armed Bandits	One-step; no context	No personalization; same arm for all users	Few discrete treatments; homogeneous user base
Contextual Bandits	One-step with context; special case of RL	Actions don't influence future states; fixed action set	Real-time personalization; immediate rewards suffice
Full RL	Multi-step; actions shape future states	Sample inefficiency; reward design; safety/interpretability	Extended sessions; ad fatigue; long-horizon objectives

Table 2. Three Evolutionary Tracks in Ad Optimization. Modern systems combine advances from all three tracks. Practitioners should identify which dimension is the bottleneck: Are objectives misaligned (Track 1)? Are representations too coarse (Track 2)? Or is the decision horizon too myopic (Track 3)?

The evolution of ad pricing, from paying for impressions to paying for user actions, created pressure within the online advertising industry to improve relevance and performance-based metrics [Hu 2004]. Sponsored search, introduced by GoTo (later Overture), allowed advertisers to bid on specific keywords, directly linking ads to user queries and addressing the limitations of generic advertising [Fain and Pedersen 2006]. Foundational pricing models, such as Cost per Mille

(CPM), charged advertisers per 1,000 impressions and closely mirrored offline practices [Wikipedia contributors 2025]. Yahoo!, through the acquisition of Overture [Staff 2003], later introduced Cost per Click (CPC) [Ellam and Ottaviani 2003], a performance-based pricing model that shifted costs to reflect measurable user engagement. This evolution continued with the introduction of Cost per Action (CPA), which tied fees to specific user actions beyond clicks, such as purchases or sign-ups. These changes incentivized the advertising industry to innovate, leading to the development of more relevance-focused strategies. These foundational pricing models remain integral to the utility functions that underpin contemporary ranking algorithms [Fain and Pedersen 2006]. As advertisers increasingly demanded performance-based pricing, the ad tech industry was compelled to innovate within a competitive yet lucrative market, addressing the limitations of rule-based and heuristic models through the adoption of machine learning techniques to enable more dynamic and data-driven approaches [Zhang et al. 2018].

The next set of innovations following heuristics and performance-based targeting was led by machine learning (ML) and A/B testing [Barajas et al. 2022]. From 2010 to 2015, ad selection techniques evolved significantly with the adoption of machine learning models such as Logistic Regression, Decision Trees, and Gradient Boosting Machines [Cui et al. 2015]. By leveraging historical data and diverse user behavior signals, these models enabled more accurate predictions of user engagement [Cui et al. 2015]. In parallel, A/B testing allowed practitioners to systematically compare models and features, ensuring that any new implementation delivered measurable performance gains [Barajas et al. 2022]. Despite these successes, traditional tabular models struggled to capture nuanced user intent, particularly when processing more complex data such as images or text. In the early 2010s, deep learning gained momentum with breakthroughs like AlexNet [Krizhevsky et al. 2012], which demonstrated the power of neural networks to handle unstructured high-dimensional data. This ability to capture complex, nonlinear interactions dramatically improved ad recommendations, especially through the use of embeddings, which are dense vector representations that map words or images into high-dimensional spaces. Metrics like cosine similarity further refined personalization by aligning recommendations with user interests [Covington et al. 2016]. Embedding-based models, particularly two-tower architectures, presented a major leap forward in ad recommendation systems by effectively capturing intricate relationships between users and ads. These models utilize two neural networks: one dedicated to encoding user information and the other to encoding ad (item) information. The outputs from these networks are then combined to predict user engagement with ads [Vorotilov and Shugaepov 2023]. This approach not only boosted accuracy but also scaled effectively. Despite the significant improvements that embedding-based models introduced in terms of predictive accuracy and scalability of ad recommendation systems, the practical challenges of balancing multiple objectives, such as maximizing revenue, sustaining long-term user engagement, and mitigating ad fatigue remained unresolved.

Multi-Armed Bandits (MABs) emerged as a powerful method for online advertising around the 2010s to tackle the explore–exploit tradeoff, which consists of finding the optimal balance between choosing new ad policies with unknown performance and showing older ad policies with known positive performance [Burtini et al. 2015; Schwartz et al. 2017]. MABs are a class of algorithms for dynamic decision-making under uncertainty, originally developed in mid-20th-century probability theory [Robbins 1952]. While MABs had been widely applied in areas such as clinical trials [Thompson 1933; Villar et al. 2015], they were adopted in large-scale web and advertising platforms during the early 2010s [Chapelle and Li 2011; Li et al. 2010; Schwartz et al. 2017].

MABs allow platforms to reduce unnecessary exposure to a suboptimal treatment by continuously updating the estimates of expected reward and uncertainty of which treatment is most likely to yield the highest reward [Kuleshov and Precup 2014]. In the context of online advertising, “treatments”

are typically specific ad configurations, defined by parameters such as displacement cost, reserve price, ad load, and ad placement. "Rewards" typically represent user engagement metrics such as clicks or conversions [Schwartz et al. 2017].

In contrast to traditional A/B tests, where there are randomized treatments assigned for a predetermined amount of time, MABs dynamically learn which configuration yields the highest reward for each user at a particular time. Although this adaptive learning framework manages the trade-off between minimizing regret from suboptimal treatments and exploring potentially superior configurations, defining the precise reward function to optimize remains an open challenge [Wen et al. 2019].

To formalize multi-objective trade-offs such as balancing revenue and user satisfaction, utility functions have been employed in online advertising since the early 2000s [Engel and Chickering 2008]. Drawing on microeconomic utility theory, platforms formalized multi-objective trade-offs through scalar utility functions that encode multiple objectives — such as long term revenue and engagement — into a single metric, enabling principled comparison and optimization [Engel and Chickering 2008]. When integrated into MAB frameworks, this unified reward signal guides exploration toward policies that balance short-term performance (e.g., clicks or conversions) with long-term considerations such as user trust and fatigue.

Although utility-weighted bandits can encode long-term targets via proxy rewards, they remain structurally one-step: they estimate conditional expectations under a logging policy and do not represent policy-induced feedback loops, thereby limiting counterfactual evaluation of multi-step trajectories under alternative policies.

3.2 Advances in Deep Learning: Attention Models, Transformers, and Contextual Bandits

To reiterate, a key goal of recommendation systems is to optimize for both revenue and engagement over the long run.

While embedding-based models allow to encode non-linear representations of users, items, and their interactions. However, when these representations are optimized with pointwise objectives that treat impressions or clicks as independent events, the resulting system remains fundamentally myopic.

This limitation makes sequential modeling essential. Transformers are well suited to this setting because self-attention enables the model to capture both short- and long-range dependencies across user interaction histories more effectively and efficiently than recurrent architectures [Vaswani 2017].

One of the key innovations of the Transformer architecture is its multi-head attention mechanism [Vaswani 2017]. In an advertising context, multi-head attention naturally accommodates multi-objective learning [Ma et al. 2018]. For example, one head could be tuned to prioritize short-term engagement metrics (e.g., click-through rate or CTR) by attending to recent user activities and relevant ad features, while another head could emphasize longer-term outcomes, such as mitigating ad fatigue or preserving user satisfaction [Ma et al. 2018; Zhao et al. 2019]. Each head learns a distinct set of attention weights, enabling the model to isolate and capture unique signals that serve different goals [An and Joe 2024]. The outputs from these multiple heads are then aggregated into a single representation or score, providing a unified but multi-faceted embedding of user–ad interactions [Lyu et al. 2023; Vaswani 2017]. However, determining the appropriate attention heads and their optimal weighting remains a complex challenge [Kang and McAuley 2018a; Vaswani 2017; Voita et al. 2019]. The optimal weights may vary not only across different users and content types but also over time [Zhou et al. 2019, 2018]. Additionally, shifting business objectives, engineering

constraints, and competing optimization objectives further complicate the design process [Jannach and Abdollahpouri 2023; Mehrotra et al. 2018, 2020].

Selecting the right heads in Multi-Task Attention Models and determining their optimal weighting is not a deterministic process [Michel et al. 2019; Voita et al. 2019]. It typically requires iterative experimentation, involving both offline analysis (e.g., hyperparameter tuning, head ablation studies) and online testing (A/B experiments) [Kang and McAuley 2018b; Michel et al. 2019]. This process is particularly challenging for new users and new content, where slow adaptation can result in suboptimal personalization and early churn [Bobadilla et al. 2013; Schein et al. 2002].

To address difficulties related to policy optimization, like weight tuning and head selection, *Offline Replay* is often employed to find optimal policies in large parameter search spaces [Agarwal and Chen 2016; Agarwal et al. 2020; Levine et al. 2020]. This technique typically begins with data collection via A/B/n experiments, where different policies are randomly assigned to users [Agarwal et al. 2019; Chen et al. 2009]. Afterward, Offline Replay simulates the effects of various ad policies across different user segments and estimates the potential rewards based on logged interactions [Agarwal and Chen 2016]. In this context, "reward" refers to the performance metric we aim to maximize (i.e. clicks, likes, shares, revenue) [Wu et al. 2017]. However, in many cases, the true optimal treatment is not explicitly tested [Agarwal et al. 2019]. Here, importance sampling corrects for differences between the observed behavior policy and the target policy, improving estimation accuracy [Dudík et al. 2011; Sutton and Barto 2018]. Based on the collected data on random assignments between policies and users typically involved in Offline Replay, one can use different policy learning methods to map the optimal policies from user ids, content types, and other contextual signals for the optimal weights at any point in time [Swaminathan and Joachims 2015].

Offline replay often assumes that user preferences and item availability do not change significantly during data collection. However, in real ad systems, these factors can shift, causing offline data to become stale or biased [Agarwal et al. 2014; Levine et al. 2020]. Contextual bandits address this limitation by leveraging continuous feedback from live user interactions, preventing the policy from being locked into a static offline dataset [Li et al. 2010]. Contextual bandits can be thought of as a special case of RL, where an action selection does not influence future states [Bubeck et al. 2012; Sutton and Barto 2018]. Unlike full RL, which models long-term state transitions, contextual bandits assume that each decision is independent, optimizing for immediate rewards rather than a multi-step trajectory [Bubeck et al. 2012; Sutton and Barto 2018]. This makes them particularly well-suited for recommender systems, where ranking decisions can be treated as independent events [Afsar et al. 2022; Zhou and Brunskill 2016]. Additionally, contextual bandits naturally balance exploration and exploitation, allowing them to adapt dynamically to user behavior in real time [Li et al. 2010]. However, a practical limitation of contextual bandits is that they select actions from a fixed set defined at deployment time, such as preconfigured ad policies or treatment arms [Bietti et al. 2021]. As a result, the policy's performance is inherently constrained by the quality and granularity of the available states and actions at the time of inference [Dimitrakakis and Ortner 2018].

Although contextual bandits inherently focus on short-term rewards, they can be tuned to approximate long-term objectives by encoding historical or delayed outcomes into the feature space or the immediate reward function [McDonald et al. 2023a; Wu et al. 2017]. For instance, if an attention head emphasizing ad engagement indirectly correlates with better retention, a contextual bandit can increase its weight over time, provided that retention signals are reflected in the contextual features or the reward proxy [Yi et al. 2023]. However, full RL remains more appropriate for scenarios demanding explicit multi-step optimization, where actions taken now significantly shape future user states [Gauci et al. 2018]. Contextual bandits thus offer a practical

trade-off, providing continuous adaptation and efficient online learning without the computational complexity of fully modeling multi-step state transitions [Gauci et al. 2018; Levine et al. 2020].

Despite the practical advantages that contextual bandits offer for real-time experimentation and short-horizon reward maximization, they remain limited in scenarios where user states evolve over repeated interactions [Sutton and Barto 2018]. Modern advertising platforms often contend with extended user sessions and multiple recurring visits, demanding an approach capable of modeling multi-step feedback loops. RL naturally extends these capabilities by incorporating stateful dynamics and cumulative rewards, capturing phenomena like ad fatigue [Çelik et al. 2023] and long-term retention [Wu et al. 2017]. Recent industrial deployments underscore RL's growing importance in ads: ByteDance uses RL-based systems [Zhao et al. 2021, 2020] to balance immediate revenue with session-level engagement in short-video feeds, Meta (Facebook) has leveraged policy-gradient methods for sequencing notifications and ad placements over evolving user states [Gauci et al. 2018], and companies like Google are experimenting with RL-driven ad scheduling that accounts for shifting user contexts at scale [Zhao and Liu 2024]. These efforts demonstrate how RL can transcend the myopic focus of bandits to address long-horizon outcomes, multi-objective trade-offs, and dynamic user behaviors. In the sections that follow, we explore how these RL paradigms are integrated into large-scale advertising stacks, focusing on core algorithms, design decisions, and lessons learned from real-world implementations.

4 The RL Formulation of Jointly Optimizing Ads and Content

4.1 Reward Design for Monetization-Aware Recommendation Systems

This subsection provides a structured synthesis of reward design in monetization-aware recommendation systems. It first discusses the properties a reward function should ideally satisfy and why reward specification is difficult in practice due to partial observability, sparse and delayed long-term outcomes, and competing objectives such as revenue, engagement, and ad fatigue. It then reviews how prior work operationalizes these objectives through proxy metrics and blended utility formulations, along with the rationale for these choices and their main limitations.

Relatively simple reward functions may be sufficient in controlled settings, where the objective is directly specified and outcomes are immediately observable. Reward design is substantially harder in monetization-aware recommendation systems, where the platform must infer long-term user value from noisy behavioral traces while balancing monetization against user experience.

In principle, an effective reward signal for monetization-aware recommendation systems should satisfy at least three properties. First, it should be sufficiently informative about the user's latent preferences and intent, even though these quantities are not directly observed. Second, it should provide feedback at a timescale and frequency that makes learning feasible, even when the platform ultimately cares about sparse and delayed long-term outcomes such as retention or habit formation. Third, it should faithfully encode trade-offs among competing objectives, such as revenue, engagement, and ad fatigue, rather than overemphasize any single short-term metric in isolation.

In practice, however, satisfying these properties is difficult for at least three reasons. First, reward design is constrained by *partial observability*: the system must infer users' latent preferences and intent through a fixed set of observable signals that capture different stages of response, such as immediate reactions (clicks), deeper consumption (dwell time or completion), and downstream value signals (saves, purchases, or return visits) that provide only a noisy and incomplete view of the underlying user state. Second, many of the outcomes platforms ultimately care about are *sparse and delayed*: retention and longer-term satisfaction may take weeks or months to manifest, and the effect of any single intervention is often subtle, making meaningful changes difficult to detect within the short time horizons of typical experiments [McDonald et al. 2023b]. Third, reward

specification is inherently *multi-objective*: recommendation engines must simultaneously account for potentially competing goals such as revenue, engagement, and ad fatigue, rather than optimize a single scalar outcome in isolation [Sagtani et al. 2024; Yan et al. 2020; Zhang et al. 2018].

Metrics such as click-through rate (CTR), predicted CTR, and ad impressions are commonly used as proxies for monetization, while session duration, saves, shares, and return-based signals stand in for longer-term engagement [Wang et al. 2022]. Their appeal is obvious: they provide dense, behaviorally grounded feedback that is relatively easy to log, predict, and optimize. Yet this same convenience is also their main limitation. When optimized in isolation, such proxies can misrepresent the platform's true long-term objective [Mazoure et al. 2021]. Click-based signals provide a clear example: although they are frequent and immediately observable, high CTR does not necessarily imply high ad quality, and repeated exposure to low-quality or poorly matched ads can degrade user satisfaction and future responsiveness to ads [Hohnhold et al. 2015a; Silberstein et al. 2020; Yan et al. 2020]. Even within the same family of signals, semantics can vary: short clicks may function as implicit negative feedback, whereas longer clicks or post-click dwell may better reflect genuine value. Similar tensions arise elsewhere. Diversity-related signals, for instance, may be desirable when they reflect broader use-case coverage over time, which prior work associates with improved long-term user experience and retention [Wang et al. 2022; Xu et al. 2023], yet high click entropy within a session can also indicate confusion, low relevance, or friction [Sagtani et al. 2023]. Tables 4, 5, and 6 summarize representative proxies for revenue, engagement, and ad fatigue. Overall, the issue is not the absence of informative signals, but the fact that each captures only a narrow and context-dependent slice of user value. This is precisely what motivates composite utility functions that blend multiple signals rather than relying on any single proxy in isolation.

While proxy metrics ensure that the system has informative observable signals for multiple latent aspects of user behavior, the utility function determines how those signals are combined into a single optimization objective. Some desirable properties of utility functions overlap with those of proxy metrics: the objective should remain aligned with long-term platform value, provide reward signals dense enough to support learning, and capture the relevant trade-offs among competing goals. However, utility functions also introduce an additional requirement that is distinct from proxy design: structural correctness of aggregation. In particular, the utility must specify not only which components enter the objective, but also the direction of each component's contribution, its relative magnitude, and the functional form through which components interact. This includes whether effects are linear or nonlinear, additive or interaction-dependent, and whether they exhibit saturation, thresholds, or state dependence. In monetization-aware recommendation, this distinction is critical because engagement, revenue, and fatigue signals rarely combine in a purely additive or context-invariant way. As a result, even well-chosen proxies may yield a misspecified objective if they are combined with incorrect signs, poorly calibrated coefficients, or an inappropriate functional form. For example, ad revenue may increase utility up to a point but reduce it beyond a saturation threshold; dwell time may be informative only within a certain range; and an ad click followed by a hide may convey a different signal than either event in isolation.

Proxy Metric	Short Description	Polarity
CTR (Click-Through Rate)	Approximates revenue potential; higher CTR often correlates with higher ad clicks [Yan et al. 2020; Zhang et al. 2018; Zhao et al. 2020].	**Positive** (higher is better)
pCTR (Predicted CTR)	Model-estimated probability that a user will click on an ad.	**Positive** (higher is better)
Predicted Revenue	Often a function of bids and pCTR to estimate expected revenue [Yan et al. 2020; Zhang et al. 2018].	**Positive** (higher is better)
Click Yield	Ratio of total clicks (ads + organic) to total impressions on a page; helps avoid optimizing CTR in isolation [Zhang et al. 2018].	**Positive** (higher is better)
Ad Impressions	Number of times an ad is displayed to users.	**Mixed** (context-dependent)

Table 4. Revenue proxy metrics and their polarity.

Proxy Metric	Short Description	Polarity
Session Duration	Tracks how long users stay engaged per session (proxy for longer-term engagement).	**Positive** (higher is better)
30-day Total Sessions	Aggregated count of user sessions over a 30-day window, proxy for habit or stickiness [Xu et al. 2023].	**Positive** (higher is better)
Likes / Adds to Library	High-intent actions indicating deeper user satisfaction and long-term retention [McDonald et al. 2023b].	**Positive** (higher is better)
Shares	High-intent action reflecting strong engagement.	**Positive** (higher is better)
High-quality Consumption	Consumption of content with high engagement (e.g., full video views, long reads) [Wang et al. 2022].	**Positive** (higher is better)
User Return Rate (URR)	Fraction of users who revisit the platform after a fixed interval [Wu et al. 2017].	**Positive** (higher is better)
Relative Watch Time	Percentage of a video watched, normalized by video length [Wu et al. 2018b].	**Positive** (higher is better)
Dwell Time	Actual time spent viewing/reading an item, capturing depth of engagement [Yi et al. 2014].	**Mixed** (threshold-dependent: < 5s negative; ≥ 10s positive)
Ratio-based Diversity	Proportion of unique topic clusters in consumption history [Wang et al. 2022].	**Mixed** (context-dependent)
Repeated Consumption	How often users revisit similar content, reflecting satisfaction [Wang et al. 2022].	**Mixed** (context-dependent)
Distribution-based Diversity	Entropy-based spread of consumed topics [Wang et al. 2022].	**Mixed** (context-dependent)
Short-view Rate (SVR)	Fraction of impressions where watch time is below a low threshold (e.g., < 5s) [Pan et al. 2023].	**Negative** (higher is worse)
Early-skip Rate	Fraction of videos skipped or swiped away within the first few seconds (e.g., < 2 to 3s) [Pan et al. 2023].	**Negative** (higher is worse)
Skip / Not-click Rate	Fraction of impressions with explicit negative feedback signals (e.g., skips or non-clicks), indicating low relevance [Zhao et al. 2018b].	**Negative** (higher is worse)
Non-completion Rate	Fraction of plays that end before a minimum completion threshold (e.g., $< 20\%$ watched) [Pan et al. 2023].	**Negative** (higher is worse)
Time to Revisit	Interval between consecutive visits; shorter intervals suggest higher retention [Wang et al. 2022].	**Negative** (higher is worse)

Table 5. Engagement proxy metrics and their polarity.

Proxy Metric	Short Description	Polarity
Depth	Reduction in engagement (e.g., fewer impressions per session) over time [Sagtani et al. 2023].	**Negative** (higher is worse)
Length	Decrease in time spent per session [Sagtani et al. 2023].	**Negative** (higher is worse)
Frequency	Decrease in number of sessions per week [Sagtani et al. 2023].	**Negative** (higher is worse)
Ad Load (Sessions)	Number of sessions that end (or quit) during an ad [Sagtani et al. 2023].	**Negative** (higher is worse)
Ad Load (Absolute)	Absolute ad volume across sessions and changes in that volume [Sagtani et al. 2023].	**Negative** (higher is worse)
Click Entropy	Uncertainty/diversity in ad clicks for a query; high values can indicate lower relevance [Sagtani et al. 2023].	**Negative** (higher is worse)

Table 6. Ad fatigue proxy metrics and their polarity.

4.1.1 Why Blending Ad and Organic Reward Signals Is Structurally Hard. Combining ad and organic signals into a single utility is harder than combining signals within either content type alone. Two structural asymmetries between the two content types make this particularly challenging. First, they operate on different reward timescales. Monetization outcomes such as clicks and conversions are typically attributed within a short, well-defined window and therefore generate relatively dense, observable feedback. Conversely, organic engagement outcomes such as retention, habit formation, and long-term satisfaction manifest over much longer horizons, often measured across weeks or months [Hohnhold et al. 2015b; McDonald et al. 2023b]. A simple additive utility collapses this temporal distinction into a single scalar, implicitly treating a same-day ad click as equivalent to a longer-term retention signal.

Second, the two objectives are expressed in incommensurable units. Ad utility is typically computed as bid $\times$ pCTR, a monetary quantity [Yan et al. 2020; Zhang et al. 2018], whereas organic utility is expressed in behavioral units such as saves, shares, or dwell time [McDonald et al. 2023b; Wang et al. 2022]. Combining them requires a conversion factor that translates engagement into revenue-equivalent units. The appropriate value of this factor is context-dependent: it can shift across user segments, content types, and business priorities [Hohnhold et al. 2015b], and different platforms operationalize it differently—through constrained optimization [Yan et al. 2020], context-sensitive valuation [Carrion et al. 2021], or separate reward functions with learned weighting [Zhao et al. 2020].

A canonical utility function at the policy layer to blend ads and organic content can be represented as follows [Carrion et al. 2021; Sagtani et al. 2024; Yan et al. 2020]:

$$U(x) = \alpha \times \text{Revenue} + \beta \times \text{Engagement} - \gamma \times \text{Ad Fatigue}$$

where α, β, and γ are weights that determine the relative importance of each factor. To compute the utilities of "Revenue", "Engagement", and "Ad Fatigue", typically many signals feed into each component. These signals include, but are not limited to, the ones shown in Tables 4, 5, and 6. In a bandit or RL context, $U(x)$ serves as the reward signal guiding policy updates.

Validating and refining the utility function in practice. Validating and improving a utility function involves two distinct problems. Reward validation consists of verifying that a given configuration of weights, signs, and functional form produces the intended outcomes: whether $U(x)$ as specified actually tracks the platform objectives it is meant to represent. In practice, this is assessed through online A/B experiments that track top-line platform metrics such as daily, weekly, and monthly

active users (DAU/WAU/MAU) for engagement and direct revenue for monetization [Gauci et al. 2018; Hohnhold et al. 2015b]. Standard A/B tests are typically run long enough to account for day-of-week variation and reach statistical significance (commonly on the order of one to two weeks in practice) and are well-suited for detecting short-term changes in click-through rate, revenue, and session engagement [Hohnhold et al. 2015b]. However, certain effects manifest over longer periods and are not reliably detectable within such short windows [Hohnhold et al. 2015b]. To capture these delayed effects, platforms employ long-term holdout groups, in which a subset of users is withheld from a new policy for an extended period (e.g., several months), allowing the cumulative effect to be measured before a comparison is drawn [Hohnhold et al. 2015b; McDonald et al. 2023b]. Iterative live validation of reward weights is also reported in the literature. Zhao et al. [Zhao et al. 2020] describe a multi-objective reward framework at ByteDance in which separate reward functions are defined for organic content and ads, with the relative weights between them validated through a sequence of live experiments.

The second problem, reward design, is substantially harder: the goal is to identify a better-specified $U(x)$ by iterating over the reward function itself, that is, its component selection, coefficient magnitudes, signs, and functional form. The number of candidate configurations grows combinatorially with the number of parameters: each additional weight or proxy component multiplies the number of combinations that must be evaluated, making exhaustive A/B testing prohibitively expensive at scale [Gauci et al. 2018]. This creates two compounding problems. First, the sample size required to detect meaningful differences across a large number of treatment arms grows with the number of comparisons, straining experimentation infrastructure. Second, A/B testing is inherently limited to the treatments that are explicitly tested [Dudík et al. 2011; Swaminathan and Joachims 2015]; it provides no information about the performance of configurations that were never deployed. Offline replay addresses this second limitation by enabling counterfactual estimation: logged interaction data from randomized experiments is re-weighted using importance sampling to estimate how untested configurations would have performed under historical traffic [Dudík et al. 2011; Levine et al. 2020; Swaminathan and Joachims 2015]. This allows practitioners to evaluate a much larger set of candidate configurations cheaply before committing to a live experiment.

Offline RL extends this further by not only evaluating fixed configurations but actively searching for improved reward formulations within the space of possible configurations, using logged data as a surrogate for live interaction [Levine et al. 2020]. Rather than testing each configuration independently, offline RL methods learn a policy that maximizes the blended utility while staying close to the behavior policy that generated the data, thereby limiting the risk of out-of-distribution extrapolation [Chen et al. 2022; Levine et al. 2020]. This makes offline RL a natural fit for reward design in large-scale systems, where the cost of live experimentation is high and the search space is too large for exhaustive enumeration.

In some cases, a very simple unified function can suffice over a complex utility function. Researchers at Alibaba introduced the concept of Click Yield, which is defined as the ratio of the total number of clicks on all items (ads + organic results) to the total number of impressions on a page [Zhang et al. 2018]. Click Yield provides a holistic evaluation of page performance, accounting for the interactive effects between ads and organic content, and helping to mitigate the bias that arises from examining CTR in isolation. Their optimization task maximizes total revenue while ensuring that the Click Yield does not fall below a certain threshold (T) [Zhang et al. 2018]. A comparison of utility function formulations is shown in Table 7.

Company	Paper	Organic Content Reward/Utility	Ads Reward/Utility	Blended Reward/Utility
TikTok	[Zhao et al. 2021]	User Experience (r_t^{ex}): Measures the negative impact of displaying an ad on the user's experience, with a binary outcome. A positive reward (1) is given if the user continues browsing after seeing the ad, while a negative reward (-1) is given if the user leaves the platform after seeing the ad.	Ad Income (r_t^{ad}): Represents the immediate revenue generated from displaying an ad. It provides a positive value if an ad is displayed, contributing to maximizing advertising income.	$r_t(s_t, a_t) = r_t^{ad} + \alpha \cdot r_t^{ex}$ Here, α is a hyperparameter that controls the trade-off between maximizing ad revenue and minimizing negative user experience.
TikTok	[Zhao et al. 2020]	$R_{\text{rec}}(s_t, a_t)$ Engagement metric (e.g., dwell time, clicks, purchases)	$R_{\text{ad}}(s_t, a_t) = \begin{cases} 1 & \text{if user stays after ad} \\ 0 & \text{if user leaves after ad} \end{cases}$	$R_{\text{total}}(s_t, a_t) = \alpha \cdot R_{\text{rec}}(s_t, a_t) + \beta \cdot R_{\text{ad}}(s_t, a_t)$
Alibaba	[Zhang et al. 2018]	Click Yield (CY): Measures the overall performance of the entire page, considering both ads and organic content. It is defined as the ratio of total clicks on all items (both ads and organic) over the total number of impressions. $CY_{ij} = \frac{\sum_{n=1}^{N} Click(i,j,n)}{Imp(i,j)}$ Where $Click(i,j,n)$ is the click for item n in list j, and $Imp(i,j)$ is the number of impressions.	Revenue (REV): For sponsored content, revenue is computed based on the click-through rate (CTR) and cost per click (CPC). $REV_{ij} = \sum_{n=1}^{N} CTR(i,j,n) \times CPC(i,j,n)$ The CPC is determined by the auction mechanism (usually Generalized Second Price Auction).	The system optimizes the trade-off between revenue (REV) and Click Yield (CY). The optimization problem is defined as: $\max \sum_{i,j} REV_{ij} \cdot x_{ij}$ Subject to the constraint: $\frac{\sum_{i,j} CY_{ij} \cdot x_{ij}}{\sum_{i,j} x_{ij}} \geq T$ Where T is a threshold for the average Click Yield, ensuring user satisfaction.
LinkedIn	[Yan et al. 2020]	Expected Engagement Utility: Measures user actions like clicks, comments, shares, or time spent on organic content. The goal is to maximize engagement with organic content.	Expected Revenue Utility (for ads): Evaluates the revenue generated from ads, typically calculated as the product of the bid amount and the predicted click-through rate (pCTR).	The utility function for ads and organic content is formulated as: $\max \sum_i x_i r_i - \frac{w}{2}\|x\|^2$ Subject to the engagement constraint: $\sum_i x_i u_{ai} + (1 - x_i) u_{oi} \geq C$ Where: • x_i is a decision variable determining whether to place an ad in position i, • r_i represents the revenue utility for ad i, • u_{ai} and u_{oi} represent the engagement utilities for ads and organic content, respectively, • C is the engagement threshold. The shadow bid α acts as a conversion factor to blend the two objectives by equating engagement utility and revenue in a comparable manner.

Table 7. Comparison of Utility/Reward Formulation Approaches across Different Companies

Critical comparison of utility formulations. Table 7 presents four distinct utility formulations from three companies. While all four seek to balance ad revenue with user experience, they differ substantially in how they represent each objective, how the two objectives are combined, and what signals are excluded. Understanding these differences is essential for practitioners selecting or adapting a formulation for their own systems.

Blending mechanism. Two of the four formulations use a linear weighted sum of separate ad and organic reward signals. In DEAR [Zhao et al. 2021], the blended reward is $r_t = r_t^{ad} + \alpha \cdot r_t^{ex}$, where α is a scalar hyperparameter set manually. Similarly, Zhao et al. [Zhao et al. 2020] use $R_{\text{total}} = \alpha \cdot R_{\text{rec}} + \beta \cdot R_{\text{ad}}$, with α and β tuned via live experiments. These additive forms are simple to implement and straightforward to interpret, but they implicitly assume that one unit of ad revenue is commensurable with one unit of engagement, an assumption that is context-dependent and difficult to validate without careful calibration [Hohnhold et al. 2015b]. In contrast, Alibaba [Zhang et al. 2018] avoids direct blending altogether by formulating the problem as constrained optimization: revenue is maximized subject to a minimum Click Yield threshold T. This sidesteps the unit-conversion problem by treating engagement as a hard constraint rather than a term in the objective, but it introduces a different design choice: setting T requires understanding what level of Click Yield is acceptable, which must itself be validated. LinkedIn [Yan et al. 2020] takes a third approach, using a shadow bid α as an implicit conversion factor that translates engagement utility into revenue-equivalent units during the optimization, making the exchange rate explicit and learnable, though it still requires regularization via w to prevent the revenue term from dominating.

Representation of ad reward. The formulations differ considerably in how they define the ad-side reward. DEAR [Zhao et al. 2021] and Zhao et al. [Zhao et al. 2020] use binary signals: a reward of +1 if the user continues browsing after the ad and 0 or −1 if they leave. This is an intuitive proxy for user tolerance of an ad, but it is extremely coarse. It treats all ad exposures that do not immediately cause churn as equally valuable, regardless of whether the user clicked, converted, or simply ignored the ad. Importantly, it also does not capture revenue magnitude: a high-value ad that the user tolerates is treated identically to a low-value ad. Alibaba [Zhang et al. 2018] and LinkedIn [Yan et al. 2020] instead define ad utility in monetary terms, as CTR × CPC and bid × pCTR respectively. These formulations are more aligned with the actual revenue objective of the platform, but they require accurate CTR or pCTR estimates, which introduce their own modeling challenges, particularly for rare or cold-start ads [McMahan et al. 2013].

Representation of organic reward. The organic-side signals also vary in richness. Zhao et al. [Zhao et al. 2020] define R_{rec} broadly as an engagement metric (dwell time, clicks, or purchases), leaving the specific operationalization unspecified. DEAR [Zhao et al. 2021] frames the organic reward as a binary user-experience signal identical in structure to its ad reward, which again sacrifices nuance for simplicity. Alibaba's Click Yield [Zhang et al. 2018] is a page-level aggregate that treats organic and ad clicks interchangeably in the numerator, which avoids the need to specify separate organic and ad utilities but loses the ability to distinguish between the two content types or to penalize ad fatigue. LinkedIn [Yan et al. 2020] explicitly separates engagement utilities for ads (u_{ai}) and organic content (u_{oi}) into a shared constraint, making the trade-off structure the most transparent of the four.

Limitations of existing utility formulations. A consistent limitation across all four formulations is the absence of explicit ad fatigue or long-term engagement signals. None of the blended utilities incorporate signals such as session exit rates due to ads, long-term return rates, or repeated exposure effects. As a result, policies optimized under these objectives may extract short-term revenue at the cost of gradual engagement degradation that accumulates over weeks or months and falls outside the observation window of typical A/B tests [Hohnhold et al. 2015b; McDonald et al. 2023b]. Additionally, none of the formulations account for session-level dynamics: the reward at each step is evaluated independently, without modeling how a sequence of ad exposures within a session compounds to affect the final user state. LinkedIn's constrained formulation comes closest to capturing session-level budget effects through its engagement constraint C, but this is a per-decision threshold rather than a cumulative session-level model.

Reported results. Zhao et al. [Zhao et al. 2021] report that their DEAR system, deployed on TikTok's short-video feed, achieves significant improvements in ad revenue without statistically significant degradation in organic engagement metrics in online A/B tests, though exact magnitudes are not publicly disclosed. Zhao et al. [Zhao et al. 2020] similarly report that their jointly-trained RL policy outperforms baseline bandit approaches in both revenue and engagement metrics in live experiments, with the relative weights α and β validated across a sequence of experiments. Zhang et al. [Zhang et al. 2018] report that their whole-page optimization at Alibaba yields improved revenue while maintaining Click Yield above the specified threshold T in production, demonstrating that a constrained formulation can be practically deployed at scale. Yan et al. [Yan et al. 2020] report improvements in both revenue and user engagement metrics at LinkedIn's feed, with the shadow bid approach enabling the system to adaptively balance monetization and organic quality as traffic conditions change.

Guidance for practitioners. The choice of utility formulation involves a fundamental trade-off between simplicity, revenue fidelity, and organic representativeness. The binary reward formulations (DEAR, Zhao et al. 2020) are the easiest to implement and require no revenue model, making them suitable for early-stage systems or platforms where revenue attribution is difficult. However, their coarseness limits optimization precision. The monetary utility formulations (Alibaba, LinkedIn) more faithfully represent actual platform revenue objectives, but they depend on accurate pCTR models and require setting additional hyperparameters such as the Click Yield threshold T or the regularization weight w. The constrained formulation of Alibaba is particularly well-suited when engagement is treated as a non-negotiable floor, since it directly encodes that requirement without requiring a precise conversion factor between engagement and revenue. LinkedIn's shadow-bid approach is appropriate when the conversion factor itself is a key design variable and is expected to vary across user segments or over time. Across all four approaches, none addresses long-term ad fatigue directly; practitioners operating in contexts where user retention is a primary concern should augment these formulations with explicit long-term holdout metrics [Hohnhold et al. 2015b] or consider incorporating delayed engagement signals into the reward function [McDonald et al. 2023b].

4.2 Action Space Representation

This subsection addresses a design question in monetization-aware recommendation systems: how should the action space be structured to support effective policy learning? We introduce four structural properties that industrial and academic systems implicitly or explicitly appeal to when designing ad-policy action spaces: expressiveness, orthogonality, interpretability, and consistency. This taxonomy is synthesized from recurring design considerations across published systems rather than inherited from any single source, and we use it to compare how those systems trade the properties against each other. The subsection first defines the action space formally, then analyzes each property with the trade-offs and design choices it implies, and finally summarizes how these properties map to the concrete levers used in production.

In RL, an action $a \in \mathcal{A}$ is a decision or intervention taken by the agent based on its current state to influence the environment. The action space $\mathcal{A}$ is the complete set of actions available to the agent at any decision point [Sutton and Barto 2018]. In ad recommendation, $\mathcal{A}$ consists of all combinations of ads, placements, and formats that the system can deliver [Ie et al. 2019b]. Selecting among these actions aims to maximize platform revenue while maintaining or improving user engagement [Zhao et al. 2020]. Although $\mathcal{A}$ is technically discrete, the number of combinations grows exponentially with the number of candidate ads and available slots. Enumerating or evaluating every action individually is therefore computationally infeasible, and the space must be treated using methods

designed for high-dimensional discrete decision problems [Dulac-Arnold et al. 2021; Ie et al. 2019a]. Narrowing $\mathcal{A}$ to a tractable subset involves trade-offs that the four properties below make explicit.

Expressiveness. The action space must be rich enough to represent every meaningful intervention the system can make, but richer spaces enlarge the learning problem. SlateQ [Ie et al. 2019b] exposes every item-slot combination in a slate and recovers tractability through a per-item Q-value decomposition under user-choice assumptions. DEAR [Zhao et al. 2021] restricts the action to a small discrete set of ad-placement decisions per request, trading representational power for sample efficiency and offline stability. LinkedIn's constrained formulation [Yan et al. 2020] encodes the action as a continuous shadow-bid multiplier, which retains fine-grained control without the combinatorial blow-up of slate-level actions. The design choice is therefore a trade-off between coverage of possible interventions and the sample complexity of learning an effective policy over them [Dulac-Arnold et al. 2021].

Orthogonality. Independently adjustable levers simplify attribution and A/B analysis, but treating levers as orthogonal can misrepresent coupled mechanisms in the underlying system [Koutsopoulos and Spentzouris 2016]. In production, ad load, placement, and quality are rarely independent: increasing ad load typically forces lower-quality creatives into later slots, a coupling explicitly modeled in Alibaba's whole-page optimization [Zhang et al. 2018]. Systems that treat levers as independent, such as slot-level controllers in DEAR [Zhao et al. 2021], gain debuggability and simpler offline evaluation at the cost of ignoring these couplings, which can manifest as unmodeled variance during deployment [Hohnhold et al. 2015b]. The trade-off is between analytic clarity and faithfulness to the real coupling structure of the system.

Interpretability. Actions with human-readable meaning support pre-launch review, but restricting the action space to such forms limits the behaviors a policy can express. Horizon [Gauci et al. 2018] exposes high-level business levers to engineers, which enables tracing learned decisions back to upstream signals such as commercial intent scores. LinkedIn's shadow bid [Yan et al. 2020] is similarly interpretable because operators can read the exchange rate between engagement and revenue directly from the learned parameter. SlateQ [Ie et al. 2019b], by contrast, exposes only a state-conditional slate of (ad, slot) pairs with no low-dimensional summary, so reviewers cannot characterize the policy's behavior by inspection and must defer to live A/B measurement [Dulac-Arnold et al. 2021; Glanois et al. 2024; Verma et al. 2018]. The trade-off is between the breadth of behaviors the policy can express and the ability of operators to vet it before deployment.

Consistency. Taking the same action in the same state should yield statistically predictable outcomes over many episodes [Puterman 2014]. The expected engagement and monetization outcomes for a given state-action pair should be stable enough that repeated exposure enables accurate estimation of long-term value [McMahan et al. 2013; Zhao et al. 2018a]. In practice this assumption is violated by non-stationary user preferences, seasonality, and auction dynamics [Dulac-Arnold et al. 2021; Hohnhold et al. 2015b]. Systems address the gap with different mechanisms: off-policy evaluation with importance weighting [Swaminathan and Joachims 2015], off-policy bandit learning that re-estimates under the current logging policy [Sagtani et al. 2024], and frequent retraining on rolling windows in deployed platforms [Gauci et al. 2018]. The trade-off is between the data efficiency gained from assuming stationarity and the drift risk accumulated over long training windows.

Table 8 summarizes how four representative systems operationalize the four properties. No system satisfies all four cleanly; each makes an explicit trade-off based on its deployment context.

The four properties are operationalized in production through levers that map directly onto the entries in Table 8. Typical levers include the following [Koutsopoulos and Spentzouris 2016]:

Property	DEAR [Zhao et al. 2021]	SlateQ [Ie et al. 2019b]	LinkedIn [Yan et al. 2020]	Alibaba [Zhang et al. 2018]
Expressiveness	Small discrete set per request	Full slate, per-item decomposition	Continuous shadow-bid multiplier	Whole-page joint selection
Orthogonality	Levers independent per slot	Coupled via user-choice model	Coupled via engagement constraint	Coupled across page positions
Interpretability	Binary placement lever, readable	Per-item Q-values, moderately readable	Direct engagement-revenue exchange rate	Page-level utility, low readability
Consistency mechanism	Offline replay on logged data	Simulator-based training	Off-policy logging with re-estimation	Constrained optimization under live traffic
Primary trade-off	Simplicity vs. fine-grained control	Expressiveness vs. choice-model assumption	Flexibility vs. accuracy of pCTR estimates	Global optimality vs. attribution clarity

Table 8. Comparison of action-space design choices across representative ad-policy systems. Each row corresponds to one of the four structural properties introduced in this subsection; each column describes how a published system operationalizes that property. The final row states the primary trade-off each system accepts as a consequence of its choices.

- **Ad Load.** The frequency or spacing of ads in the feed. Placing ads too closely together can lead to user fatigue and reduced click probabilities [Abrams and Vee 2007; Zhao et al. 2020].
- **Ad Quality.** Intrinsic features of the ad, such as design format, product type, accompanying text, and landing-page quality, all of which influence user engagement [Nielsen 2017; Yan et al. 2009].
- **Ad Placement.** The position of the ad within the feed. Ads placed earlier are more likely to be clicked, though relevance to surrounding content also plays a significant role [Saha et al. 2021; Yuan et al. 2020; Zhao et al. 2020].
- **Ad Relevance.** The alignment between the ad and user preferences or intent. Ads shown adjacent to related organic content, such as a hotel ad next to vacation-related search results, are more likely to drive engagement [Chakrabarti et al. 2008; Zhou et al. 2018].

Taken together, the four properties and their associated trade-offs provide a structured way to compare published systems and to reason about which design choices are appropriate for a given deployment context. The next subsection turns to the state space, which defines the context in which these actions are taken.

4.3 State Space Representation

The preceding subsection examined how to structure the action space. This subsection addresses the corresponding question for the state space in policies that jointly rank ads and organic content in large-scale recommendation systems: what information must the state encode so that the policy can balance short-term monetization against long-term user engagement across successive interactions? To compare representative systems, we use five structural properties as an analytical lens: Markovianity, expressiveness, stability under distribution shift, sample efficiency, and interpretability. As with the action-space taxonomy in the preceding subsection, this set is synthesized from recurring design considerations across the systems reviewed here rather than inherited from any single source or presented as a field-wide consensus taxonomy. The subsection first defines the

state space formally, then analyzes each property with the trade-offs and design choices it implies, and finally summarizes how representative systems operationalize them.

In RL, a *state* $s \in \mathcal{S}$ denotes the information on which the agent conditions when selecting an action. Together with the action space $\mathcal{A}$, a transition kernel $P(s' \mid s, a)$, a reward function $R(s, a)$, and a discount factor $\gamma \in [0, 1)$, the state space is one component of the Markov decision process (MDP) used as the standard formal framework for RL [Puterman 2014; Sutton and Barto 2018]. In monetization-aware ranking, a single state instance typically aggregates user-level signals (e.g., identifiers, demographics, long-term preference embeddings), session-level context (e.g., recent engagement, dwell time, device), and platform-level features (e.g., current ad load, inventory composition, the slate rendered so far) at the time of a decision [Ie et al. 2019b; Xia et al. 2023; Zhao et al. 2021]. Because only a subset of the latent user intent and platform dynamics is observable through these signals, constructing s is itself a design problem.

Markovianity. A state is Markovian when it contains everything needed to predict the next user-level outcome and monetization signal given the chosen action, with no additional information to be gained from the prior trajectory [Puterman 2014]. Markovianity matters because the value estimators, off-policy methods, and policy-gradient updates used to learn ad ranking decisions all assume the next reward depends only on (s, a). When it does not, $Q(s, a)$ averages over unobserved trajectory states, increasing the variance of counterfactual value estimates learned from logged feedback [Swaminathan and Joachims 2015]. In ad policies, exact Markovianity is generally unrealistic: user intent and long-term satisfaction are partially observable, seasonality and auction dynamics shift over days to weeks, and new inventory arrives continuously [Dulac-Arnold et al. 2021; Hohnhold et al. 2015b]. The practical question is therefore how Markovian the state must be for those measurements to remain stable across retraining cycles. Published systems address this issue by extending the state with longer behavioral histories (DIN [Zhou et al. 2018], PinnerFormer [Pancha et al. 2022]) or by compressing long histories into fixed-size representations through self-attention modules (SASRec [Kang and McAuley 2018b], TransAct [Xia et al. 2023]). The trade-off is between the bias introduced by ignoring trajectory history and the sample complexity added by enlarging the effective state dimension.

Expressiveness. A useful state representation should encode the axes a joint ad / organic policy needs in order to distinguish between users, sessions, and slate contexts. In monetization-aware ranking, expressiveness has three components: user-level signals that separate long-term commercial intent from casual browsing (PinnerFormer [Pancha et al. 2022], DIN [Zhou et al. 2018]), session-level signals that capture short-term context such as recent dwell or query intent (TransAct [Xia et al. 2023]), and platform-level signals that describe what has already been rendered on the page, including the slate-so-far state used by SlateQ [Ie et al. 2019b] and the request-level features used by DEAR [Zhao et al. 2021]. Adding axes can broaden the policy class, but it also enlarges s, which raises serving latency, model size, and the variance of the off-policy gradient estimator [Liu et al. 2022; Naumov et al. 2019; Swaminathan and Joachims 2015]. The trade-off is between covering the decision-relevant variation in the population and keeping s small enough that value estimates remain statistically stable.

Stability under distribution shift. The state representation should remain informative as the logging policy, ad inventory, and user behavior change between training cycles [Gama et al. 2014; Hohnhold et al. 2015b]. One recurring pattern in large-scale recommender systems combines slower-changing user or item representations with features refreshed closer to serving time, such as recent engagement, trending content, or current ad load [Liu et al. 2022; Xia et al. 2023; Zhang et al. 2024]. The slower-changing component is intended to smooth short-horizon noise and remain stable across retraining cycles, while the near-real-time component captures drift fast enough that the policy does not become stale between deployments. Representations that lean entirely on

dense online features chase short-term signal at the cost of unstable Q-value estimates across A/B windows; representations that lean entirely on offline features are robust to drift but slow to react to new inventory or campaign launches. The trade-off is between responsiveness to behavioral drift and the stability of value estimates over the retraining horizon.

Sample efficiency. Off-policy estimators, policy-gradient updates, and counterfactual evaluators all require enough (s, a, r) tuples per region of state space to estimate $Q(s, a)$ with controllable variance [Levine et al. 2020; Swaminathan and Joachims 2015]. As the dimensionality of s grows, the effective sample density per region drops, inflating the variance of importance-sampled estimates and slowing actor convergence [Dulac-Arnold et al. 2021]. Engineered low-dimensional features such as commercial-intent scores, ad-load history, or dwell quantiles can converge faster from the same logged volume, but they encode the designer's prior about which axes matter, and that prior bounds the policy class [Zhao et al. 2021]. The trade-off is between the bias accepted from feature engineering and the sample volume required to learn over a higher-dimensional learned representation.

Interpretability. State components with human-readable meaning let operators trace policy behavior to upstream signals, support pre-launch review, and isolate regressions when A/B lifts move [Gauci et al. 2018; Glanois et al. 2024]. Engineered features such as commercial-intent scores, recency buckets, or ad-load counters allow a reviewer to predict roughly how the policy will respond before traffic is sent. Embedding-dominated states such as those used by PinnerFormer [Pancha et al. 2022] and TransAct [Xia et al. 2023] often expose no low-dimensional summary an operator can read, so behavioral validation defers to live A/B measurement. The trade-off is between the breadth of behavior the state can support and the ability of operators to vet the policy before deployment, mirroring the same trade-off identified in the action-space subsection.

Table 9 summarizes how four representative systems operationalize the five properties. As with the action-space comparison in the preceding subsection, no system satisfies all five cleanly; each accepts an explicit trade-off based on its deployment context.

Reported results. Zhao et al. [Zhao et al. 2021] evaluate DEAR on TikTok's short-video feed and report that the request-level state representation supports ad revenue improvements without statistically significant degradation in organic engagement in online A/B tests, though exact magnitudes are not publicly disclosed. Ie et al. [Ie et al. 2019b] evaluate SlateQ in the RecSim simulator [Ie et al. 2019a] before deploying to a YouTube-scale service, and report that state representations including the slate-so-far context yield gains on a long-term user satisfaction metric relative to myopic slot-level baselines. Pancha et al. [Pancha et al. 2022] evaluate PinnerFormer at Pinterest against prior user-embedding baselines and report that a single user embedding trained on a multi-week engagement sequence yields meaningful offline retrieval-quality gains and online engagement lift across Home feed and related surfaces. Xia et al. [Xia et al. 2023] evaluate TransAct at Pinterest as a ranking-stage augmentation over PinnerFormer, and report that combining the long-horizon embedding with a real-time session sequence yields additional engagement lift, with the offline component absorbing long-term user signal and the real-time sequence capturing immediate intent. Zhou et al. [Zhou et al. 2018] evaluate DIN on Alibaba display advertising and report AUC improvements over a deep CTR baseline by attending to prior user behavior conditional on the candidate ad. Across these reports, state-representation choices are evaluated through a combination of offline replay on logged interactions, simulator-based ablation, and live A/B measurement; in our review of these papers, none share random seeds, hyperparameter search spaces, or full reproducibility artifacts, which limits cross-system comparison to the qualitative axes in Table 9.

Before discussing the levers that shape these properties, we summarize the representational building blocks that s is typically assembled from in ad + organic systems. *User embeddings* encode

Property	DEAR [Zhao et al. 2021]	SlateQ [Ie et al. 2019b]	PinnerFormer [Pancha et al. 2022]	TransAct [Xia et al. 2023]
Markovianity	Request-level features with short engagement history	User context with slate-so-far; user-choice model assumed	Fixed-size embedding summarizing multi-week engagement trajectory	Long-term embedding combined with real-time session sequence
Expressiveness	Engineered low-dimensional per-request features	Full slate with per-item decomposition	Single user embedding capturing long-term preference structure	Session sequence plus long-term embedding covering both horizons
Stability under distribution shift	Drift absorbed through frequent retraining on rolling windows	Simulator-based training; drift handling not explicitly reported	Daily batch refresh; stable across retrain cycles, slow to inventory shifts	Explicit offline / online split, directly targeting drift
Sample efficiency	Low-dimensional state converges quickly from logged data	Per-item Q decomposition reduces sample complexity per slate	Pre-trained embedding shares signal across downstream tasks	Self-attention compression controls cost of higher state dimensionality
Interpretability	Engineered features readable by operators	Per-item Q-values partially readable; slate behavior not	Embedding-only; opaque to pre-launch review	Embedding-dominated; behavioral validation requires A/B
Primary trade-off	Simplicity vs. coverage of long-term behavior	Choice-model assumption vs. slate-level expressiveness	Long-horizon stability vs. responsiveness to fresh inventory	Expressiveness and stability vs. serving latency and model size

Table 9. Comparison of state-representation design choices across representative ad-recsys systems. Each row corresponds to one of the five structural properties introduced in this subsection; each column describes how a published system operationalizes that property. The final row states the primary trade-off each system accepts as a consequence of its choices.

long-term commercial intent, preference structure, and behavioral regularities (PinnerFormer [Pancha et al. 2022], DIN [Zhou et al. 2018]); they give the policy a population-level prior over which ad categories are relevant before any session-specific signal arrives. *Ad features* encode sparse attributes such as ad identity, campaign, product category, or advertiser context, helping the policy generalize across ads with limited interaction history [Cheng et al. 2016; McMahan et al. 2013]. *User-ad interaction features*, whether computed as dot products, cross features, or attention over recent impressions [Cheng et al. 2016; Zhou et al. 2018], expose the fit between a specific ad and the current user; they are the axis most directly predictive of click and conversion and are the one most sensitive to distribution shift. *Contextual signals* (location, device, time, slate-so-far) anchor the representation when longer histories are weak or absent [Zhao et al. 2021]. Each lever below acts on one or more of these building blocks.

The five properties are operationalized in production through a small set of state-representation levers that recur across published systems:

- **Long-history user embeddings.** Extend s along the temporal axis to reduce history dependence and broaden expressiveness, at the cost of sample efficiency (PinnerFormer [Pancha et al. 2022], DIN [Zhou et al. 2018]).

- **Sequence compression.** Apply self-attention modules to reduce a long behavioral history to a fixed-size summary, recovering sample efficiency without discarding the temporal signal (SASRec [Kang and McAuley 2018b], TransAct [Xia et al. 2023]).
- **Offline / online representation decomposition.** Combine slow-changing batch-computed representations with near-real-time user or context features to target stability under distribution shift while controlling serving latency [Liu et al. 2022; Xia et al. 2023; Zhang et al. 2024].
- **Engineered contextual features.** Anchor s with low-dimensional human-readable signals (location, device, ad-load history, onboarding interests) to support interpretability and address the cold-start regime when long histories are unavailable [Covington et al. 2016].

These levers are typically combined: heterogeneous embeddings are concatenated and projected through learned linear layers into a shared dimension [Covington et al. 2016; Grbovic and Cheng 2018; Pancha et al. 2022], with zero-padding and truncation avoided because they inject noise or cause information loss [Xia et al. 2023]. The set of levers a system enables, rather than any single architectural choice, determines where it lands on the property trade-offs above.

Guidance for practitioners. The choice of state representation involves a trade-off between expressiveness, sample efficiency, serving cost, and the ability to audit policy behavior before deployment. Engineered low-dimensional states such as those used in DEAR [Zhao et al. 2021] are appropriate for early-stage systems, surfaces with limited logged volume, or deployments where pre-launch interpretability is a regulatory or product constraint; their sample efficiency and readability come at the cost of coverage of long-term user behavior. Slate-so-far representations with user-choice decomposition, as in SlateQ [Ie et al. 2019b], are appropriate when the action space is genuinely slate-level and the user-choice assumption is defensible for the target surface; they extend expressiveness without forcing exhaustive enumeration of slates, but require simulator or counterfactual infrastructure to validate the user-choice model itself. Long-horizon user embeddings such as PinnerFormer [Pancha et al. 2022] and DIN [Zhou et al. 2018] are appropriate when long-term engagement is the primary objective, the platform has sufficient daily active users to stabilize embedding training, and mature batch-compute infrastructure is already in place; practitioners operating below this scale should expect the sample-efficiency trade-off to dominate. Hybrid offline / online representations such as TransAct [Xia et al. 2023] and large-scale online user-representation systems for ads personalization [Zhang et al. 2024] are appropriate for mature production systems where both drift responsiveness and long-term signal matter, and where the serving stack can accept the latency overhead of real-time sequence encoding. Among the systems compared here, none directly addresses the joint ad + organic-content attribution problem in its state design; practitioners optimizing for the balance between monetization and organic engagement should augment the state with organic-slot context (surrounding organic quality, recent organic dwell) or combine state-space extensions with the constrained-reward formulations discussed in Section 4.2 [Yan et al. 2020; Zhang et al. 2018].

Taken together, the five properties, the comparison in Table 9, and the levers above provide a structured way to compare how published systems construct s and to reason about which representational choices are appropriate for a given deployment context. The next subsection turns to policy learning, which determines how the agent uses (s, a, r) tuples drawn from this state space to update its decision rule.

4.4 Policy Learning

This subsection defines the ad-serving policy, contrasts deterministic and stochastic formulations, and walks through the two policy-learning design decisions that recur most often in published large-scale ad systems: offline versus online learning, and value-based versus policy-based optimization. Examples from production recommendation and ad-serving systems illustrate the trade-offs each decision imposes on safety, sample efficiency, and end-to-end pipeline integration.

In the context of ad policy optimization, the policy $\pi(a|s)$ defines how the system selects ads a to display to a user in a particular state s, using levers such as ad load, placement, and price thresholds, as detailed in Section 4.2 [Ie et al. 2019c; Zhang et al. 2018; Zhao et al. 2021]. The state s may include the user's profile, browsing history, session context, and inferred commercial intent, as discussed in Section 4.3 [Pancha et al. 2022; Xia et al. 2023; Zhou et al. 2019]. The policy can be deterministic, meaning that the same state always leads to the same ad decision, or stochastic, meaning that the system samples from a probability distribution over possible ad decisions [Levine et al. 2020; Sutton and Barto 2018].

In many large-scale recommendation and ad-serving systems, learned policies are stochastic or include stochastic exploration mechanisms, whereas simpler or older systems may use deterministic rules, such as inserting a fixed ad after a fixed number of organic impressions [Gauci et al. 2018; Zhao et al. 2021]. Stochastic policies are useful because they support exploration: the system can sample from multiple plausible actions rather than always selecting the historically best-performing one [Russo et al. 2018; Sutton and Barto 2018]. This can help uncover ad-user pairings that may improve long-term outcomes [Mazoure et al. 2021; Zou et al. 2019]. Stochastic policies also provide a practical mechanism for managing uncertainty and partial information [Dulac-Arnold et al. 2021; Levine et al. 2020]. When no single action clearly serves objectives such as monetization and long-term engagement, randomizing over plausible actions allows the system to spread risk and continue learning [Jannach and Abdollahpouri 2023; Mehrotra et al. 2020]. Deterministic policies can encode complex tradeoffs, but they may hard-code assumptions that do not generalize well or require retuning as user behavior shifts [Dulac-Arnold et al. 2021]. Stochastic policies, by contrast, can adapt through probabilistic weighting as more data becomes available [Gauci et al. 2018; Levine et al. 2020].

4.4.1 Offline vs Online Policy Learning. This subsubsection explains why large-scale ad systems often begin with offline policy learning before introducing online updates, and compares the safety, scalability, and adaptation trade-offs of the two approaches. Depending on whether the agent learns by training on historical data or by interacting with the environment in real time, policy learning can be categorized as either offline or online [Sutton and Barto 2018]. In offline policy learning, models are trained using logged data from past user interactions without actively modifying the live system. This is often the safer and more scalable starting point in large platforms because it avoids deploying untested policies directly to users and is especially useful when real-time exploration is expensive [Gauci et al. 2018; Levine et al. 2020; Mehrotra et al. 2020].

Monte Carlo (MC) methods estimate the expected return of a policy by averaging cumulative rewards across complete episodes of logged interactions [Sutton and Barto 2018]. Because they require full trajectories before updates, they are most useful when rewards are delayed and live exploration is costly, which makes them a plausible early-stage offline learning tool for extracting long-term value signals from existing logs [Levine et al. 2020]. However, MC methods have important limitations: waiting for complete episodes can make them sample-inefficient, long or sparse-reward episodes can delay policy improvement, and noisy or stochastic environments can produce high-variance estimates unless variance-reduction techniques are used [Dulac-Arnold et al. 2019; Sutton and Barto 2018]. Monte Carlo approaches also assume that logged episodes are

representative of the environment; if the logging policy is narrow or biased, importance sampling corrections become necessary to avoid skewed estimates [Swaminathan and Joachims 2015]. Due to these limitations, MC methods are typically not used in high-frequency online systems where quick updates are critical [Dulac-Arnold et al. 2019]. As an illustrative example of offline learning, an ad-ranking system may train a policy from historical impression logs that record the user state, the ad or slate shown, and later reward signals such as clicks, conversions, revenue, or retention proxies. The learner must infer how to improve ad selection from these logged trajectories without trying new placements or ad mixes in the live system.

Logged data only contains actions chosen by the historical policy, which means there is no direct visibility into what would have happened if alternative actions had been taken in the same context [Swaminathan and Joachims 2015]. This lack of counterfactual information makes it difficult to evaluate or iterate on new policies [Dudík et al. 2011]. However, methods such as inverse propensity scoring (IPS), which re-weights logged rewards to approximate outcomes under a different policy, and doubly robust estimators, which combine IPS with model-based value predictions, can help address this challenge [Dudík et al. 2011; Levine et al. 2020; Swaminathan and Joachims 2015]. These approaches enable limited forms of counterfactual reasoning, especially when the new policy does not diverge too far from the one that generated the data [Dudík et al. 2011; Swaminathan and Joachims 2015].

In ad policy optimization, this limitation arises when the log contains little or no data for a candidate ad action in a given user context. A function approximator trained over related state-action pairs may still estimate the action's expected return. It does so by generalizing from similar users, sessions, ads, and placements. These estimates can support limited counterfactual reasoning when the target action is close to regions with sufficient logged support. If the logged data contains few similar examples, however, the estimate becomes much less reliable [Dudík et al. 2011; Levine et al. 2020; Swaminathan and Joachims 2015].

To overcome the constraints of limited exploration in offline settings, online policy learning enables agents to refine their policies by interacting with the environment and learning from new experiences [Dimitrakakis and Ortner 2018]. The agent is able to explore different actions, observe the consequences in real time, gather feedback, and update its behavior accordingly. This ability to learn directly through interaction with the environment makes online learning more flexible and adaptive, although it also introduces greater risk [Dulac-Arnold et al. 2019].

To mitigate risks and costs of online exploration, large-scale systems commonly begin with offline training or offline evaluation before controlled online rollout [Chen et al. 2022; Gauci et al. 2018; Levine et al. 2020]. Online is introduced gradually via controlled bandit loops, feature gates, or soft rollouts. This hybrid approach of offline-then-online policy learning is commonly seen in industry. In their offline RL system, YouTube trained an actor-critic model entirely on logged user interaction data, using Monte Carlo returns to estimate long-term value, and only deployed policies to production after thorough offline evaluation [Chen et al. 2022]. At Meta, the Horizon platform first trains policies offline using counterfactual estimators like Doubly Robust evaluation, and then deploys them through gated rollouts, which refer to the controlled and staged release of new policies to a limited portion of traffic in order to minimize risk during online learning [Gauci et al. 2018]. Spotify similarly adopts a cautious approach by first optimizing contextual bandit policies offline, and then introducing online learning through incremental updates and careful exploration strategies [Mehrotra et al. 2020].

4.4.2 Value-based vs Policy-based Learning. RL algorithms can be divided into value-based and policy-based methods based on how they learn to make decisions to maximize cumulative rewards [Russell and Norvig 2016; Sutton and Barto 2018]. Value-based methods first estimate the value of

states or state-action pairs and then derive a policy from these estimates, whereas policy-based methods learn the policy directly by optimizing expected cumulative reward [Schulman et al. 2017]. In other words, value-based methods focus on learning the utility of states or actions and derive the optimal behavior indirectly by acting greedily with respect to this learned value. Policy-based methods, in contrast, directly optimize the policy without estimating value functions.

Because value-based methods rely on greedy action selection to determine the next best action, they require comparing the estimated value of all possible actions at each decision point (e.g. each state) [Dulac-Arnold et al. 2019; Sutton and Barto 2018]. As the number of available actions grows, and as the agent encounters more states during interaction, this exhaustive comparison becomes computationally intractable [Dulac-Arnold et al. 2019]. For this reason, value-based methods are particularly well-suited for environments with discrete or low-dimensional action spaces, where enumerating and comparing action values remains computationally feasible. Common algorithms used in value-based learning include Q-learning, SARSA, and Deep Q-Networks (DQN) [Mnih et al. 2015; Sutton and Barto 2018].

In Policy-based learning, the agent tries out different strategies and tweaks their behavior directly based on what works well overall, rather than scoring every possible move. Formally, the goal is to learn a parameterized policy $\pi_\theta(a \mid s)$, where θ denotes the parameters of the policy [Schulman et al. 2017; Sutton and Barto 2018]. In practice, especially in modern industrial applications, these parameters are often the weights of a neural network, though other function approximators are possible [Levine et al. 2020; Mnih et al. 2016]. Neural network-based policies are common in modern industrial RL and recommendation settings due to their expressiveness and scalability, making them particularly well-suited for environments with complex and high-dimensional state spaces [Chen et al. 2022; Gauci et al. 2018; Lin et al. 2023]. Formally, the objective function to be maximized is the expected cumulative reward under the policy:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \right] \tag{1}$$

where τ denotes a trajectory: a sequence of states and actions generated by following the policy π_θ [Sutton and Barto 2018]. To optimize this objective, the policy gradient theorem provides a way to compute the gradient of the expected return with respect to the policy parameters, given by:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a \mid s) \cdot Q^{\pi_\theta}(s, a) \right] \tag{2}$$

where $Q^{\pi_\theta}(s, a)$ is the action-value function under the current policy [Sutton and Barto 2018]. Since the goal is to maximize return, gradient ascent, not descent, is used. Common policy-based RL algorithms include REINFORCE, Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO), and various Actor-Critic methods, which combine both policy and value learning in a hybrid architecture [Schulman et al. 2015, 2017; Sutton and Barto 2018; Williams 1992].

In theory, either value-based or policy-based learning is sufficient to solve an RL problem [Kaelbling et al. 1996; Sutton and Barto 2018]. However, policy-based methods tend to be more suitable for ad policy optimization for a number of reasons [Chen et al. 2022; Gauci et al. 2018]. First, policy-based methods are preferred in scenarios where stochastic policies are desired because they explicitly parameterize the policy as a probability distribution over actions, rather than implicitly choosing a single best action [Williams 1992]. What policy-based methods learn are the stochastic policy $\pi_\theta(a \mid s)$ directly, whereby the output is itself a probability distribution (e.g. softmax over action logits, Gaussian in continuous space) [Schulman et al. 2015]. Moreover, policy-based methods are also desirable when careful tuning of exploration-exploitation is required, as they can learn when

and how to be random, whereas value-based methods cannot [Schulman et al. 2015; Williams 1992]. In contrast, because value-based methods learn the action-value function $Q(s, a)$ and then derive the deterministic policy $\arg\max_a Q(s, a)$, value-based methods cannot directly represent probability distributions over actions [Dulac-Arnold et al. 2019]. To accommodate for stochasticity, value-based methods resort to ϵ-greedy approaches, which are ad hoc and not a learned methodology [Kaelbling et al. 1996]. Finally, Policy-based methods are often used when we want to optimize a policy across an entire ML pipeline [Chen et al. 2022]. It is common for ad-ranking systems to start with a transformer-based encoder for ad retrieval and to conclude with a multi-head attention blending layer that predicts a downstream action, where end-to-end differentiability is required to allow the gradient to flow and therefore learn a cohesive policy [Pancha et al. 2022; Xia et al. 2023; Zhao et al. 2021]. This is difficult to achieve with value-based methods, which estimate a value function $Q(s, a)$ and then select actions via $\arg\max_a Q(s, a)$ [Dulac-Arnold et al. 2019]. Because arg max is non-differentiable, it breaks gradient flow in neural networks [Dulac-Arnold et al. 2019]. In contrast, policy-based methods directly parameterize the policy $\pi_\theta(a \mid s)$ using differentiable functions (e.g., softmax or Gaussians) [Williams 1992]. As a result, gradients can backpropagate through the entire pipeline, from the final action probabilities to earlier neural layers, enabling a fully integrated, end-to-end approach that can incorporate embeddings, attention mechanisms, and encoder-decoder architectures [Chen et al. 2022; Xia et al. 2023; Zhao et al. 2021].

While policy-based methods are powerful and overcome many of the limitations of value-based methods, policy-based methods also have their shortcomings [Levine et al. 2020]. Policy-based methods assume that the performance of the gradient can be accurately estimated and optimized through sampling [Williams 1992]. However, these estimates often suffer from high variance, which can lead to unstable learning and slow convergence [Schulman et al. 2015]. Because of the absence of an explicit value function, policy-based methods tend to require a larger number of samples compared to value-based methods, making them less sample-efficient [Dulac-Arnold et al. 2019; Levine et al. 2020]. To overcome these limitations, many large-scale production systems adopt hybrid approaches that combine both value-based and policy-based learning, most notably in the form of actor-critic architectures [Chen et al. 2022; Gauci et al. 2018]. In these methods, the actor is responsible for learning the policy, while the critic estimates the value function to guide and stabilize the policy updates [Sutton and Barto 2018; Williams 1992]. This combination improves learning stability, speeds up convergence, and enhances interpretability, making it well-suited for complex, real-world applications such as recommendation and ad-serving systems [Chen et al. 2022; Dulac-Arnold et al. 2019; Gauci et al. 2018].

Put simply, value-based and policy-based methods differ in what the model learns. Consider a request in which the system must choose between two ad-load actions: in a simplified hypothetical setting, a low ad load might mean ads occupying 10% of the feed slots, whereas a high ad load might mean ads occupying 50% of the feed slots. A value-based method first estimates the expected return of each action in the current user context s, for example $Q(s, \text{low ad load})$ and $Q(s, \text{high ad load})$, and then derives a policy by choosing the action with the larger estimated value [Sutton and Barto 2018]. A policy-based method instead trains a model whose direct output is the action distribution $\pi_\theta(a \mid s)$; for the same request, the model might assign 0.75 probability to low ad load and 0.25 probability to high ad load. This distinction matters in ad policy optimization because value-based methods are easier to apply when the candidate actions are small and enumerable, whereas policy-based methods are more natural when the system needs a stochastic policy or an end-to-end differentiable ranking pipeline.

The value-based vs policy-based taxonomy offers a useful lens for categorizing RL approaches in ad policy optimization, based on how decisions are learned to maximize cumulative rewards [Lin et al. 2023; Sutton and Barto 2018; Zhao et al. 2021]. Large-scale ad systems such as Google

Ads have long relied on supervised prediction models for immediate outcomes such as click prediction [McMahan et al. 2013]. Value-based RL formulations, by contrast, have been explored for personalized ad recommendation, sponsored-search real-time bidding, and budget-constrained display advertising, where the learner estimates the long-term value of candidate ad actions before deriving a policy [Theocharous et al. 2015; Wu et al. 2018a; Zhao et al. 2018a]. Value-based learning works well when the action space is discrete, small, and easily enumerable, allowing the agent to derive a policy through the maximization of the learned Q-function [Dulac-Arnold et al. 2019; Sutton and Barto 2018]. In contrast, policy-based approaches are better suited for continuous, high-dimensional, or unstructured decision spaces [Dulac-Arnold et al. 2019; Schulman et al. 2015; Sutton and Barto 2018]. It is also possible to optimize for the ad policy directly in a more flexible formulation [Sutton and Barto 2018; Williams 1992; Zhao et al. 2021]. For instance, rather than predefining a limited set of treatment arms, TikTok used Proximal Policy Optimization (PPO) to learn a stochastic ranking policy that directly maps user context to slates of ads [Zhao et al. 2021]. This approach can handle potentially large or complex sets of ads more flexibly, allowing for direct parameterization of the stochastic policy [Dulac-Arnold et al. 2019; Ie et al. 2019b; Zhao et al. 2021]. The built-in stochasticity facilitates exploration (helping to uncover effective ad configurations) and adaptation to shifting user behaviors, without constraining the policy to a small, predefined action space [Sutton and Barto 2018; Williams 1992; Zhao et al. 2021].

Actor-critic architectures present a hybrid method that leverages the benefits of both value-based and policy-based methods [Sutton and Barto 2018]. The actor learns the policy, while the critic provides value estimates to stabilize and accelerate learning [Lin et al. 2023; Sutton and Barto 2018]. The critic acts as a baseline to reduce the high variance of policy gradient methods (e.g. REINFORCE), and the actor's explicit policy handles large or complex action spaces better than value-based approaches (e.g. DQN) [Chen et al. 2022; Mnih et al. 2016; Sutton and Barto 2018].

There are well-documented industry examples of the use of actor-critic methods in ad recommendations. For example, Facebook reported that RL models (including off-policy actor-critic methods) trained on its Horizon platform outperformed and even replaced previous supervised recommendation systems in tasks like notifications and video ranking [Gauci et al. 2018]. YouTube introduced an off-policy actor-critic approach to augment their previous REINFORCE agent (policy-gradient based) by adding a critic network that estimates Q-values via temporal difference (TD) learning [Chen et al. 2022]. The use of Q-value estimates enables lower-variance policy updates compared to Monte Carlo returns; one-step importance correction offers more robust handling of off-policy data than full-trajectory importance sampling; and bootstrapping with one-step TD targets improves the estimation of long-term rewards [Chen et al. 2022].

Actor-critic methods are particularly useful in multi-objective settings where both revenue and user engagement must be balanced, and where using logged data for offline training is desired [Cai et al. 2023]. To this end, two-staged constrained actor-critic methods have been effectively used to optimize the main objective while enforcing guardrail constraints through a constrained MDP formulation [Cai et al. 2023]. In stage one, multiple actor-critic agents learn to optimize auxiliary objectives (e.g. engagement such as likes, saves, and shares) [Cai et al. 2023]. In stage two, a primary actor-critic learns a policy maximizing the main objective (e.g. revenue, ad clicks) while staying close to the auxiliary policies to satisfy those constraints. This two-tier actor-critic approach achieved a superior balance: offline experiments showed it outperformed alternative RL methods on the trade-off between watch time and other interaction metrics [Cai et al. 2023].

Actor-critic RL brings the bias–variance trade-off under control: the critic (value-based element) introduces bias but provides low-variance evaluation, and the actor (policy-based element) ensures we still directly optimize the true objective [Chen et al. 2022; Sutton and Barto 2018]. Understanding the structure and cardinality of the action and state spaces, the importance of calibrating exploration

vs exploitation, the available computational budget, and the degree of cohesion required across an ML pipeline are some key factors that help determine which paradigm to use in practice [Dulac-Arnold et al. 2019; Schulman et al. 2017; Zhu et al. 2021].

4.5 Balancing Exploration and Exploitation in Policy Learning

Balancing exploration and exploitation is a fundamental challenge in policy learning [Sutton and Barto 2018], particularly at large scale, whereby exhaustively exploring all options is computationally infeasible, the cost of suboptimal decisions compounds quickly, and sustained bias toward early winners can prevent the discovery of truly optimal policies over time [Dulac-Arnold et al. 2021].

In ad policy optimization, tuning exploration and exploitation is critical across multiple dimensions, notably ad selection and ad load. In *ad selection*, a system must decide between exploring new ads that might yield higher rewards and exploiting ads known to perform well [Auer et al. 2002; Bietti et al. 2021; Russo et al. 2018]. In setting *ad load* configuration, platforms must weigh the exploration of higher ad exposure for queries with high commercial intent against the risk of degrading user experience [Carrion et al. 2021; Hohnhold et al. 2015b; Zhang et al. 2018]. While exploring more aggressive ad load configurations may unlock revenue gains, it risks triggering ad fatigue, reduced engagement, and eventual user attrition [Sagtani et al. 2023; Silberstein et al. 2023].

In this section, we introduce the commonly used methods in industry as well as their pros and cons, and the contexts in which they are most effective.

4.5.1 ***ϵ-Greedy Strategy.*** One of the simplest and most widely used exploration methods, the epsilon-greedy strategy allows the agent to primarily exploit known successful actions (with probability $1 - \epsilon$) while occasionally exploring random actions (with probability ϵ) [Russell and Norvig 2016; Sutton and Barto 2018]. Over time, ϵ *can* be reduced, shifting focus toward exploitation as the agent gains more experience [Russo et al. 2018; Sutton and Barto 2018]. However, deciding how much and when to decrease ϵ is non-trivial, as reducing exploration too quickly can lead to premature convergence on suboptimal actions [Dulac-Arnold et al. 2019; Sutton and Barto 2018].

ϵ-Greedy based exploration/exploitation is a relatively easy method to implement, computationally efficient, and thus well-suited for real-time applications [Russell and Norvig 2016; Sutton and Barto 2018]. However, because it treats all actions uniformly during exploration, ϵ-greedy does not differentiate between tried and untried actions, account for uncertainty in value estimates, or prioritize actions with higher potential expected improvement [Dulac-Arnold et al. 2019; Sutton and Barto 2018]. This can lead to missed opportunities, especially in environments with large action spaces, because as the number of possible actions increases, the probability of selecting any particular action during exploration decreases [Bubeck et al. 2012; Sutton and Barto 2018]. As a result, many potentially high-value actions may never be explored, or may be explored too infrequently to accurately estimate their value [Dulac-Arnold et al. 2019; Sutton and Barto 2018]. ϵ-greedy is most effective in smaller action spaces, where exploration remains tractable and the risk of encountering extremely low-value actions is lower due to the reduced likelihood of extreme outcomes [Dulac-Arnold et al. 2019; Sutton and Barto 2018].

4.5.2 ***Upper Confidence Bound (UCB).*** In contrast to ϵ-greedy strategies, which explore uniformly at random and do not incorporate knowledge of an action's estimated value or the uncertainty in that estimate, the UCB algorithm ranks actions using a composite score that integrates both expected reward and uncertainty [Auer et al. 2002]. At each time step t, UCB selects the action a

that maximizes:

$$\text{UCB}_t(a) = \hat{Q}_t(a) + \alpha \cdot \sqrt{\frac{\log t}{N_t(a)}} \tag{3}$$

where the first term $\hat{Q}_t(a)$ is the estimated average reward of action a, and the second term is called the exploration bonus, which reflects the uncertainty in the reward estimate [Auer et al. 2002; Bubeck et al. 2012]. $N_t(a)$ is the number of times action a has been selected, which shrinks the bonus as the action is chosen more often and reduces exploration as a result [Auer et al. 2002; Bubeck et al. 2012]. Similarly, $\log t$ grows slowly over time, which ensures continued exploration in later rounds. This formulation with the exploration and exploitation terms ensures that actions are prioritized either because they have performed well (high $\hat{Q}_t(a)$) or because they are underexplored (high uncertainty bonus) [Auer et al. 2002; Bubeck et al. 2012]. α is a tunable parameter that determines how much exploration is valued over exploitation [Auer et al. 2002; Bubeck et al. 2012; Sutton and Barto 2018]. As an action is selected more frequently, its estimate becomes more reliable, uncertainty decreases, and the confidence interval narrows [Auer et al. 2002; Bubeck et al. 2012]. As a result, the UCB score may increase or decrease: if the action is good, its estimated reward improves and the score may remain high or even rise despite the shrinking bonus; if the action is bad, the estimate stays low and the diminishing uncertainty term no longer inflates its score, causing it to be deprioritized over time [Auer et al. 2002; Bubeck et al. 2012].

The unified scoring function in UCB naturally balances exploration and exploitation [Auer et al. 2002; Bubeck et al. 2012]. It allows the agent to identify promising but underexplored actions while avoiding those that are confidently known to perform poorly [Auer et al. 2002]. While UCB provides tighter bounds on long-term regret, it can be more computationally expensive than simpler methods like ϵ-greedy and may over-explore suboptimal actions with high uncertainty, leading to short-term performance losses [Bubeck et al. 2012; Dulac-Arnold et al. 2019; Sutton and Barto 2018].

LinkedIn used UCB-style methods to optimize ad placement and pacing in user feeds. In the early stages of their system, UCB was employed to evaluate which treatment arms, such as different spacing, reserve price, and ad load configurations, performed best across user segments. UCB helped balance exploration of new pacing parameters with exploitation of those already showing promising results [Yan et al. 2020]. Microsoft, through the Contextual Bandit Bake-Off, found that UCB and its linear variant, LinUCB, were competitive or superior to ϵ-greedy approaches across multiple offline datasets for ad personalization and content recommendation [Bietti et al. 2021]. Facebook (Meta) incorporated UCB-inspired bandit methods within its Horizon RL platform, particularly for notification ranking and ad sequencing tasks. These methods were used in early-stage exploration, particularly in cold-start scenarios or on surfaces with limited historical data, before transitioning to more complex actor-critic architectures for long-term optimization [Gauci et al. 2018].

UCB performs best in environments where data is sparse, optimization horizons are long, and the cost of regret is high, making it well-suited for ad policy optimization where long-term performance (like maximizing user lifetime value or engagement) is key [Bubeck et al. 2012; Dulac-Arnold et al. 2019; Theocharous et al. 2015]. However, in real-time applications where immediate results are prioritized, UCB's tendency to over-explore may be less suitable [Dulac-Arnold et al. 2019; McMahan et al. 2013; Sutton and Barto 2018]. Additionally, the choice of the exploration parameter α can be brittle as too conservative a setting may limit the discovery of better actions, while too aggressive a setting may amplify noise and degrade performance.

4.5.3 ***Thompson Sampling.*** Thompson Sampling (TS) uses a Bayesian approach in which actions are selected by sampling from the posterior distribution over rewards and choosing the action with

the highest sampled value [Russo et al. 2018; Sutton and Barto 2018]. Formally, the selected action at time t is:

$$a_t = \arg\max_{a \in \mathcal{A}} \tilde{Q}_t(a) \tag{4}$$

Formally, the selected action at time t is: where $\mathcal{A}$ is the action set and $\tilde{Q}_t(a)$ is a sample from the posterior distribution of the expected reward for action a at time t [Russo et al. 2018].

While each individual decision is based on a single random sample, across many realizations, the likelihood that an action is selected reflects the posterior probability that it is the optimal choice [Russo et al. 2018]. Initially, TS relies on a prior distribution that reflects the initial belief about the reward of each action, usually based on some default assumptions or domain knowledge [Russo et al. 2018]. As more data is collected through interactions (e.g., clicks or conversions in an ad setting), the prior is updated to form a posterior distribution, representing the system's refined belief about expected rewards [Russo et al. 2018]. Actions with greater uncertainty (i.e., wider posterior distributions) are more likely to generate extreme reward samples, occasionally leading to their selection even if their mean reward is lower [Russo et al. 2018]. In contrast, actions with higher expected rewards and narrower posterior distributions are selected more consistently due to their reliably higher sampled values [Russo et al. 2018]. Thus, TS leverages posterior variance to simultaneously enable exploration and exploitation without requiring an explicit exploration bonus as used in UCB methods [Russo et al. 2018].

While both UCB and TS leverage estimates of variability to manage exploration and exploitation, they differ in how they quantify uncertainty, handle randomness, and select actions [Russo et al. 2018; Sutton and Barto 2018]. UCB constructs an upper confidence bound for each action and always selects the action with the highest upper bound [Auer et al. 2002], whereas TS samples from the posterior distribution over rewards and chooses the action with the highest sampled value [Russo et al. 2018]. As a result, UCB deterministically selects the same action given the same inputs, while TS's choice varies across runs due to its sampling-based nature [Russo et al. 2018]. Although UCB is often easier to implement, as it does not require specifying prior or posterior distributions, it can be more brittle to tune (e.g., selecting an appropriate exploration constant α) [Auer et al. 2002; Sutton and Barto 2018]. In contrast, TS is generally more robust to hyperparameter choices and tends to perform better in non-stationary environments [Chapelle and Li 2011; Russo et al. 2018].

Studies by Chapelle and Li [Chapelle and Li 2011] show that TS can perform as well as, or better than, UCB in terms of cumulative regret, defined as the total loss incurred from not consistently selecting the optimal action [Chapelle and Li 2011; Russo et al. 2018]. This is especially true in scenarios where data is sparse or feedback is delayed [Chapelle and Li 2011]. In contrast to UCB, which might over-explore uncertain actions with low potential returns, TS ensures a more balanced and probabilistic form of exploration [Russo et al. 2018]. TS is naturally well-suited for industry problems like selecting ads for new or existing users, particularly in dynamic environments such as online advertising, where user preferences shift frequently and new content is introduced continuously. However, because TS relies on randomized sampling, it can exhibit high variance, which may lead to less stable performance in environments that prioritize short-term accuracy or require highly reliable decision-making [Russo et al. 2018].

4.5.4 ***Broader Considerations for Exploration and Exploitation.*** While the previous discussion focused on canonical exploration strategies that are widely deployed and highly effective in industrial recommender and ad systems (e.g., ϵ-Greedy, UCB, Thompson Sampling), it is also important to recognize that additional techniques are often employed to address more complex,

high-dimensional, or rapidly evolving environments. For instance, in dynamic ad allocation systems, parameters are often tuned in real-time to balance short-term revenue generation against user experience metrics such as Click Yield, which measures the rate of user interactions per ad impression [Zhang et al. 2018]. By adjusting trade-off parameters in response to shifting engagement patterns, platforms can dynamically prioritize either immediate monetization or sustained user satisfaction. Moreover, alternative approaches such as entropy-regularized policies [Mnih et al. 2016], Bayesian optimization [Shahriari et al. 2015], and policy gradient methods like Proximal Policy Optimization (PPO) [Schulman et al. 2017] extend the exploration-exploitation framework in settings where continuous adaptation is critical.

Method	How It Works	Pros	Cons	Best Use Cases
Epsilon-Greedy	Selects the best-known action with probability $1-\epsilon$ and explores randomly with probability ϵ.	• Simple to implement • Low computational cost • Easy to understand	• Fixed exploration rate • Might miss better actions • Not adaptive to context	• Simple, static environments • Prioritize computational efficiency
Upper Confidence Bound (UCB)	Selects actions based on confidence intervals combining reward estimates and uncertainty.	• Systematic exploration • Minimizes long-term regret • Strong theoretical foundations	• May overexplore uncertain actions • Higher computational complexity	• Real-time ad placement • Long-term optimization
Regret Confidence Bounds (RegCB)	Extends UCB by using regression oracles to compute confidence intervals based on contextual features.	• Handles high-dimensional, sparse data • More accurate reward estimates • Well-suited for complex environments	• High computational cost • Difficult to scale	• High-dimensional environments • Needs precise reward prediction
Thompson Sampling (TS)	Uses Bayesian sampling from posterior distributions to estimate optimal actions.	• Adapts quickly to new data • Naturally balances exploration • Effective for cold-start problems	• Can exhibit high variance • Moderate computational complexity	• Cold-start and sparse data • Dynamic, shifting environments

Table 10. Comparison of Exploration-Exploitation Methods in Ad Policy Optimization

5 Conclusions

The integration of RL in policy optimization for large-scale recommender systems has gained popularity as a promising approach to balance long-term and short-term objectives in environments with complex user behavior and revenue dynamics. However, scaling RL in practice presents challenges, including vast state and action spaces, reliance on proxy metrics, and significant computational demands. This paper provides an overview of effective, scalable techniques that address these challenges, offering insights into how RL can be applied at scale. Unlike traditional methods that focus on optimizing short-term outcomes, RL's cumulative reward framework enables it to account for both immediate and delayed feedback. Together, contextual bandits and offline replay offer scalable, interpretable solutions that approximate full RL while managing real-time constraints. Contextual bandits support selecting optimal actions within each interaction based on

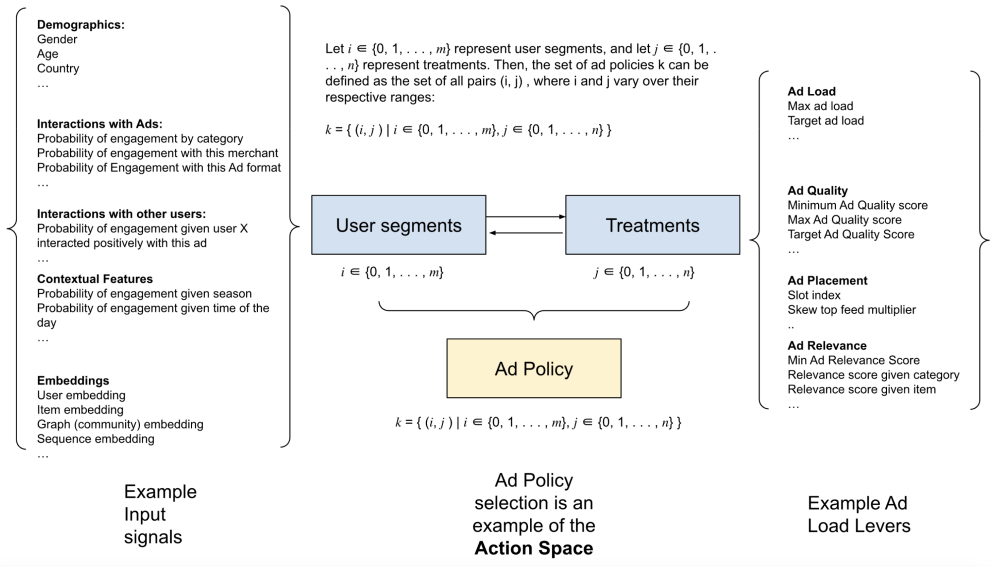

Fig. 2. Example of an action set in the context of ad policies in recommendation systems

the current state, maximizing immediate rewards, which are typically proxy metrics of the true long-term reward. Offline replay complements this by enabling counterfactual learning: multiple treatments can be tested across randomized populations to evaluate causal effects, refining policies for diverse user segments.

Future research may focus on enhancing interpretability, refining reward functions, and exploring decentralized approaches to handle complex decision layers. These advancements will bring RL closer to fulfilling its potential in providing tailored, dynamic ad recommendations that align user satisfaction with business goals in large-scale recommendation systems.

Acknowledgments

I would like to thank Adam Obeng, Bee-Chung Chen, and Jay Adams for their valuable guidance, critical coaching, and insights, which have significantly shaped my approach and understanding of industry practices. Their expertise has been instrumental in aligning academic research with practical applications.

References

Zoë Abrams and Erik Vee. 2007. Personalized ad delivery when ads fatigue: An approximation algorithm. In *International Workshop on Web and Internet Economics*. Springer, 535–540.

M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement learning based recommender systems: A survey. *Comput. Surveys* 55, 7 (2022), 1–38.

Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. 2014. Taming the monster: A fast and simple algorithm for contextual bandits. In *International conference on machine learning*. PMLR, 1638–1646.

Deepak Agarwal, David Cornell, Bin Huang, Bee-Chung Chen Lee, Ruoxi Tang, Le Zhang, Avery Wolf, Yuchen Shi, Ye Wang, Yin Guo, Srijan Kumar Garimella, Nikita Korolko, and Sandeep Muthukrishnan. 2019. Online Parameter Selection for Web-based Services. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. 1645–1653.

Deepak K Agarwal and Bee-Chung Chen. 2016. *Statistical methods for recommender systems*. Cambridge University Press.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. 2020. An Optimistic Perspective on Offline Reinforcement Learning. In *International Conference on Machine Learning (ICML)*. 104–114.

Zijian An and Inwhee Joe. 2024. TMH: Two-Tower Multi-Head Attention neural network for CTR prediction. *Plos one* 19, 3 (2024), e0295440.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-Time Analysis of the Multiarmed Bandit Problem. *Machine learning* 47 (2002), 235–256.

Joel Barajas, Narayan Bhamidipati, and James G Shanahan. 2022. Online advertising incrementality testing: Practical lessons, paid search and Emerging challenges. In *European Conference on Information Retrieval*. Springer, 575–581.

Alberto Bietti, Alekh Agarwal, and John Langford. 2021. A contextual bandit bake-off. *Journal of Machine Learning Research* 22, 133 (2021), 1–49.

Jesus Bobadilla, Fernando Ortega, Antonio Hernando, and Arturo Guti
'errez. 2013. Recommender systems survey. *Knowledge-Based Systems* 46 (2013), 109–132. doi:10.1016/j.knosys.2013.03.012

Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning* 5, 1 (2012), 1–122.

Giuseppe Burtini, Jason L. Loeppky, and Ramon Lawrence. 2015. Improving Online Marketing Experiments with Drifting Multi-armed Bandits. In *Proceedings of the 7th International Conference on Agents and Artificial Intelligence (ICAART)*, Vol. 1. SCITEPRESS, 135–146.

Qingpeng Cai, Zhenghai Xue, Chi Zhang, Wanqi Xue, Shuchang Liu, Ruohan Zhan, Xueliang Wang, Tianyou Zuo, Wentao Xie, Dong Zheng, et al. 2023. Two-stage constrained actor-critic for short video recommendation. In *Proceedings of the ACM Web Conference 2023*. 865–875.

Carlos Carrion, Zenan Wang, Harikesh Nair, Xianghong Luo, Yulin Lei, Xiliang Lin, Wenlong Chen, Qiyu Hu, Changping Peng, Yongjun Bao, et al. 2021. Blending advertising with organic content in e-commerce: A virtual bids optimization approach. *arXiv preprint arXiv:2105.13556* (2021).

Fatih Çelik, Mehmet Safa Çam, and Mehmet Ali Koseoglu. 2023. Ad avoidance in the digital context: A systematic literature review and research agenda. *International Journal of Consumer Studies* 47, 6 (2023), 2071–2105.

Deepayan Chakrabarti, Deepak Agarwal, and Vanja Josifovski. 2008. Contextual advertising by combining relevance with click feedback. In *Proceedings of the 17th international conference on World Wide Web*. 417–426.

Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. *Advances in neural information processing systems* 24 (2011).

Bee-Chung Chen, Dmitry Pavlov, John F Canny, and Lodewijk Rijckaert. 2009. Large-scale Behavioral Targeting. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '09)*. 209–218.

Minmin Chen, Can Xu, Vince Gatto, Devanshu Jain, Aviral Kumar, and Ed Chi. 2022. Off-policy actor-critic for recommender systems. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 338–349.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

Qing Cui, Feng-Shan Bai, Bin Gao, and Tie-Yan Liu. 2015. Global optimization for advertisement selection in sponsored search. *Journal of Computer Science and Technology* 30 (2015), 295–310.

Cristian Danescu-Niculescu-Mizil, Andrei Z Broder, Evgeniy Gabrilovich, Vanja Josifovski, and Bo Pang. 2010. Competing for users' attention: on the interplay between organic and sponsored search results. In *Proceedings of the 19th international conference on World wide web*. 291–300.

Christos Dimitrakakis and Ronald Ortner. 2018. Decision making under uncertainty and reinforcement learning. *Book available at http://www. cse. chalmers. se* (2018).

Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601* (2011).

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning* 110, 9 (2021), 2419–2468.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. 2019. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901* (2019).

Andrew Ellam and Marco Ottaviani. 2003. Overture and Google: Internet Pay-Per-Click (PPC) Advertising Auctions. *London Business School Case Study CS-03-022* (2003).

Yagil Engel and David Maxwell Chickering. 2008. Incorporating user utility into sponsored-search auctions. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*. 1565–1568.

Tom Everitt, Marcus Hutter, Ramana Kumar, and Victoria Krakovna. 2021. Reward tampering problems and solutions in reinforcement learning: A causal influence diagram perspective. *Synthese* 198, Suppl 27 (2021), 6435–6467.

Daniel C Fain and Jan O Pedersen. 2006. Sponsored search: A brief history. *Bulletin-American Society For Information Science And Technology* 32, 2 (2006), 12.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM computing surveys (CSUR)* 46, 4 (2014), 1–37.

Jason Gauci, Edoardo Conti, Yitao Liang, Kittipat Virochsiri, Yuchen He, Zachary Kaden, Vivek Narayanan, Xiaohui Ye, Zhengxing Chen, and Scott Fujimoto. 2018. Horizon: Facebook's open source applied reinforcement learning platform. *arXiv preprint arXiv:1811.00260* (2018).

Claire Glanois, Paul Weng, Matthieu Zimmer, Dong Li, Tianpei Yang, Jianye Hao, and Wulong Liu. 2024. A survey on interpretable reinforcement learning. *Machine Learning* (2024), 1–44.

Mihajlo Grbovic and Haibin Cheng. 2018. Real-time personalization using embeddings for search ranking at airbnb. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 311–320.

Henning Hohnhold, Deirdre O'Brien, and Diane Tang. 2015a. Focus on the Long-Term: It's better for Users and Business. In *Proceedings 21st Conference on Knowledge Discovery and Data Mining*. Sydney, Australia. http://dl.acm.org/citation.cfm?doid=2783258.2788583

Henning Hohnhold, Deirdre O'Brien, and Diane Tang. 2015b. Focusing on the long-term: It's good for users and business. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1849–1858.

Yu Jeffrey Hu. 2004. Performance-based pricing models in online advertising. *Available at SSRN 501082* (2004).

Eugene Ie, Chih-wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. 2019a. Recsim: A configurable simulation platform for recommender systems. *arXiv preprint arXiv:1909.04847* (2019).

Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. 2019b. SlateQ: A Tractable Decomposition for Reinforcement Learning with Recommendation Sets.. In *IJCAI*, Vol. 19. 2592–2599.

Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Morgane Lustman, Vince Gatto, Paul Covington, et al. 2019c. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. *arXiv preprint arXiv:1905.12767* (2019).

Dietmar Jannach and Himan Abdollahpouri. 2023. A Survey on Multi-Objective Recommender Systems. *Frontiers in Big Data* 6 (2023). doi:10.3389/fdata.2023.1157899

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.

Wang-Cheng Kang and Julian McAuley. 2018a. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. 197–206. doi:10.1109/ICDM.2018.00035

Wang-Cheng Kang and Julian McAuley. 2018b. Self-Attentive Sequential Recommendation. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM)*. 197–206. doi:10.1109/ICDM.2018.00035

Tanya Kant. 2021. A History of the Data-Tracked User. *The MIT Press Reader* (2021). https://thereader.mitpress.mit.edu/a-history-of-the-data-tracked-user/

Iordanis Koutsopoulos and Panagiotis Spentzouris. 2016. Native advertisement selection and allocation in social media post feeds. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*. Springer, 588–603.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

Volodymyr Kuleshov and Doina Precup. 2014. Algorithms for Multi-Armed Bandit Problems. *arXiv preprint arXiv:1402.6028* (2014). https://arxiv.org/abs/1402.6028

Nathan Lambert, Thomas Krendl Gilbert, and Tom Zick. 2023. Entangled preferences: The history and risks of reinforcement learning and human feedback. *arXiv preprint arXiv:2310.13595* (2023).

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. 661–670.

Yuanguo Lin, Yong Liu, Fan Lin, Lixin Zou, Pengcheng Wu, Wenhua Zeng, Huanhuan Chen, and Chunyan Miao. 2023. A survey on reinforcement learning for recommender systems. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

Zhuoran Liu, Leqi Zou, Xuan Zou, Caihua Wang, Biao Zhang, Da Tang, Bolin Zhu, Yijie Zhu, Peng Wu, Ke Wang, et al. 2022. Monolith: real time recommendation system with collisionless embedding table. *arXiv preprint arXiv:2209.07663* (2022).

Shanshan Lyu, Qiwei Chen, Tao Zhuang, and Junfeng Ge. 2023. Entire Space Learning Framework: Unbias Conversion Rate Prediction in Full Stages of Recommender System. *arXiv preprint arXiv:2303.00276* (2023).

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge*

discovery & data mining. 1930–1939.

Bogdan Mazoure, Paul Mineiro, Pavithra Srinath, Reza Sharifi Sedeh, Doina Precup, and Adith Swaminathan. 2021. Improving long-term metrics in recommendation systems using short-horizon reinforcement learning. *arXiv preprint arXiv:2106.00589* (2021).

Thomas M McDonald, Lucas Maystre, Mounia Lalmas, Daniel Russo, and Kamil Ciosek. 2023a. Impatient bandits: Optimizing recommendations for the long-term without delay. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1687–1697.

Thomas M. McDonald, Lucas Maystre, Mounia Lalmas, Daniel Russo, and Kamil Ciosek. 2023b. Optimizing for the Long-Term Without Delay. Spotify Research Blog. https://research.atspotify.com/2023/07/optimizing-for-the-long-term-without-delay/

H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1222–1230.

Rishabh Mehrotra, James McInerney, Hugues Bouchard, Mounia Lalmas, and Fernando Diaz. 2018. Towards a Fair Marketplace: Counterfactual Evaluation of the Trade-off between Relevance, Fairness & Satisfaction in Recommendation Systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*.

Rishabh Mehrotra, Niannan Xue, and Mounia Lalmas. 2020. Bandit Based Optimization of Multiple Objectives on a Music Streaming Platform. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. 3224–3233. doi:10.1145/3394486.3403392

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are Sixteen Heads Really Better Than One?. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*. 14014–14024.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533. doi:10.1038/nature14236

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. 2019. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091* (2019).

Nielsen. 2017. When It Comes to Advertising Effectiveness, What Is Key? Nielsen Insights Online Article. https://www.nielsen.com/insights/2017/when-it-comes-to-advertising-effectiveness-what-is-key/ Accessed: 2025-03-27.

Yunzhu Pan, Nian Li, Chen Gao, Jianxin Chang, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2023. Learning and optimization of implicit negative feedback for industrial short-video recommender system. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 4787–4793.

Nikil Pancha, Andrew Zhai, Jure Leskovec, and Charles Rosenberg. 2022. Pinnerformer: Sequence modeling for user representation at pinterest. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 3702–3712.

Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Herbert Robbins. 1952. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.* 58, 5 (1952), 527–535. doi:10.1090/S0002-9904-1952-09620-8

Stuart J Russell and Peter Norvig. 2016. *Artificial intelligence: a modern approach*. pearson.

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. 2018. A Tutorial on Thompson Sampling. *Foundations and Trends® in Machine Learning* 11, 1 (2018), 1–96.

Hitesh Sagtani, Madan Gopal Jhawar, Akshat Gupta, and Rishabh Mehrotra. 2023. Quantifying and leveraging user fatigue for interventions in recommender systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2293–2297.

Hitesh Sagtani, Madan Gopal Jhawar, Rishabh Mehrotra, and Olivier Jeunen. 2024. Ad-load Balancing via Off-policy Learning in a Content Marketplace. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 586–595.

Koustuv Saha, Yozen Liu, Nicholas Vincent, Farhan Asif Chowdhury, Leonardo Neves, Neil Shah, and Maarten W Bos. 2021. Advertiming matters: Examining user ad consumption for effective ad allocations on social media. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–18.

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. 2002. Methods and Metrics for Cold-start Recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '02)*. 253–260. doi:10.1145/564376.564421

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*. PMLR, 1889–1897. https://proceedings.mlr.press/v37/schulman15.pdf

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Eric M Schwartz, Eric T Bradlow, and Peter S Fader. 2017. Customer acquisition via display advertising using multi-armed bandit experiments. *Marketing Science* 36, 4 (2017), 500–522.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2015), 148–175.

Natalia Silberstein, Or Shoham, and Assaf Klein. 2023. Combating Ad Fatigue via Frequency-Recency Features in Online Advertising Systems. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 4822–4828.

Natalia Silberstein, Oren Somekh, Yair Koren, Michal Aharon, Dror Porat, Avi Shahar, and Tingyi Wu. 2020. Ad close mitigation for improved user experience in native advertisements. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 546–554.

Wall Street Journal Staff. 2003. Yahoo Agrees to Acquire Overture For $1.63 Billion. *The Wall Street Journal* (2003). https://www.wsj.com/articles/SB105818965623153600

Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (second ed.). The MIT Press. http://incompleteideas.net/book/the-book-2nd.html

Adith Swaminathan and Thorsten Joachims. 2015. Counterfactual risk minimization: Learning from logged bandit feedback. In *International conference on machine learning*. PMLR, 814–823.

Georgios Theocharous, Philip S. Thomas, and Mohammad Ghavamzadeh. 2015. Personalized ad recommendation for life-time value optimization. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*. 1807–1813.

William R. Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* 25, 3-4 (1933), 285–294.

Bram van den Akker, Olivier Jeunen, Ying Li, Ben London, Zahra Nazari, and Devesh Parekh. 2024. Practical Bandits: An Industry Perspective. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 1132–1135.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. 2018. Programmatically interpretable reinforcement learning. In *International conference on machine learning*. PMLR, 5045–5054.

Sofía S. Villar, Jack Bowden, and James Wason. 2015. Multi-armed Bandit Models for the Optimal Design of Clinical Trials: Benefits and Challenges. *Statist. Sci.* 30, 2 (2015), 199–215. doi:10.1214/14-STS504

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*. 5797–5808.

Vladislav Vorotilov and Ilnur Shugaepov. 2023. Scaling the Instagram Explore Recommendations System. *Engineering at Meta* (2023). https://engineering.fb.com/2023/08/09/ml-applications/scaling-instagram-explore-recommendations-system/

George A Vouros. 2022. Explainable deep reinforcement learning: state of the art and challenges. *Comput. Surveys* 55, 5 (2022), 1–39.

Yuyan Wang, Mohit Sharma, Can Xu, Sriraj Badam, Qian Sun, Lee Richardson, Lisa Chung, Ed H Chi, and Minmin Chen. 2022. Surrogate for long-term user experience in recommender systems. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 4100–4109.

Ying Wen, Yao Mu, and Martin Ester. 2019. Learning Multi-Objective Rewards and User Utility Function in Recommender Systems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*. International Joint Conferences on Artificial Intelligence Organization, 3719–3725. doi:10.24963/ijcai.2019/517

Wikipedia contributors. 2025. Cost per mille — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Cost_per_mille [Online; accessed 23-January-2025].

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8, 3-4 (1992), 229–256. doi:10.1007/BF00992696

Di Wu, Xiujun Chen, Xun Yang, Hao Wang, Qing Tan, Xiaoxun Zhang, Jian Xu, and Kun Gai. 2018a. Budget constrained bidding by model-free reinforcement learning in display advertising. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1443–1451.

Qingyun Wu, Hongning Wang, Liangjie Hong, and Yue Shi. 2017. Returning is believing: Optimizing long-term user engagement in recommender systems. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1927–1936.

Siqi Wu, Marian-Andrei Rizoiu, and Lexing Xie. 2018b. Beyond views: Measuring and predicting engagement in online videos. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 12.

Xue Xia, Pong Eksombatchai, Nikil Pancha, Dhruvil Deven Badani, Po-Wei Wang, Neng Gu, Saurabh Vishwas Joshi, Nazanin Farahpour, Zhiyuan Zhang, and Andrew Zhai. 2023. Transact: Transformer-based realtime user action model for recommendation at pinterest. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5249–5259.

Ruiyang Xu, Jalaj Bhandari, Dmytro Korenkevych, Fan Liu, Yuchen He, Alex Nikulkov, and Zheqing Zhu. 2023. Optimizing long-term value for auction-based recommender systems via on-policy reinforcement learning. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 955–962.

Jun Yan, Ning Liu, Gang Wang, Wen Zhang, Yun Jiang, and Zheng Chen. 2009. How much can behavioral targeting help online advertising?. In *Proceedings of the 18th international conference on World wide web*. 261–270.

Jinyun Yan, Zhiyuan Xu, Birjodh Tiwana, and Shaunak Chatterjee. 2020. Ads allocation in feed via constrained optimization. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3386–3394.

Congrui Yi, David Zumwalt, Zijian Ni, and Shreya Chakrabarti. 2023. Progressive horizon learning: Adaptive long term optimization for personalized recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 940–946.

Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. 2014. Beyond clicks: dwell time for personalization. In *Proceedings of the 8th ACM Conference on Recommender systems*. 113–120.

Bowen Yuan, Yaxu Liu, Jui-Yang Hsia, Zhenhua Dong, and Chih-Jen Lin. 2020. Unbiased Ad click prediction for position-aware advertising systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 368–377.

Wei Zhang, Dai Li, Chen Liang, Fang Zhou, Zhongke Zhang, Xuewei Wang, Ru Li, Yi Zhou, Yaning Huang, Dong Liang, et al. 2024. Scaling User Modeling: Large-scale Online User Representations for Ads Personalization in Meta. In *Companion Proceedings of the ACM Web Conference 2024*. 47–55.

Weiru Zhang, Chao Wei, Xiaonan Meng, Yi Hu, and Hao Wang. 2018. The whole-page optimization via dynamic ad allocation. In *Companion Proceedings of the The Web Conference 2018*. 1407–1411.

Jun Zhao, Guang Qiu, Ziyu Guan, Wei Zhao, and Xiaofei He. 2018a. Deep reinforcement learning for sponsored search real-time bidding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1021–1030.

Xiangyu Zhao, Changsheng Gu, Haoshenglun Zhang, Xiwang Yang, Xiaobing Liu, Jiliang Tang, and Hui Liu. 2021. Dear: Deep reinforcement learning for online advertising impression in recommender systems. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 750–758.

Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018b. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1040–1048.

Xiangyu Zhao, Xudong Zheng, Xiwang Yang, Xiaobing Liu, and Jiliang Tang. 2020. Jointly Learning to Recommend and Advertise. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, New York, NY, USA, 3319–3327. doi:10.1145/3394486.3403233

Yu Zhao and Fang Liu. 2024. A survey of retrieval algorithms in ad and content recommendation systems. *arXiv preprint arXiv:2407.01712* (2024).

Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM conference on recommender systems*. 43–51.

Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep Interest Evolution Network for Click-Through Rate Prediction. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*. 5941–5948. doi:10.1609/aaai.v33i01.33015941

Guorui Zhou, Chengru Song, Xiaoqiang Zhu, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. 1059–1068. doi:10.1145/3219819.3219823

Li Zhou and Emma Brunskill. 2016. Latent contextual bandits and their application to personalized recommendations for new users. *arXiv preprint arXiv:1604.06743* (2016).

Jie Zhu, Fengge Wu, and Junsuo Zhao. 2021. An overview of the action space for deep reinforcement learning. In *Proceedings of the 2021 4th International Conference on Algorithms, Computing and Artificial Intelligence*. 1–10.

Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement learning to optimize long-term user engagement in recommender systems. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2810–2818.