GEAR: A Counterfactual Multi-Agent Reinforcement Learning Framework for Strategic Resource Allocation in Game Recommendation Systems

Lizhe Wang
Shanghai Institute of Commerce & Foreign Languages, College of Art Design & Media; Shanghai, 201399 China.
E-mail: lizhew2025@163.com

Keywords: deep learning, user behavior analysis, personalized recommendation, game recommendation, causal disentanglement, reinforcement learning

Received: July 16, 2025

As the world's game market is still experiencing its explosive expansion, personalized recommender systems have become a necessity in order to enhance player experience and platform stickiness. However, conventional recommendation models, which often try to maximize short-term interaction measures like click-through rates, inevitably lead to content homogenization and degrade the diversity of the content ecosystem. This myopic focus ultimately undermines long-term player retention. To address this root difficulty, this thesis suggests a new paradigm that recasts the recommendation issue as from what to recommend to how to recommend strategically. This work innovatively models a game recommendation platform, which is composed of multiple scenarios, as a cooperative multi-agent system where every scenario is an agent. They have the common objective of optimizing the long-term ecosystem health of the platform. To this end, we design and implement a novel multi-agent reinforcement learning algorithm, GEAR (Guild-based Ecosystem-aware Allocation of Resources). We model this problem as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) and solve it using GEAR, which is based on the Centralized Training with Decentralized Execution (CTDE) framework and features a novel counterfactual credit assignment mechanism. GEAR is run under the Centralized Training with Decentralized Execution (CTDE) framework. Its main innovation is a counterfactual-based credit assignment mechanism, enabling each agent to accurately assess its marginal contribution to a global, long-term utility function—a composite measure of player retention, content diversity, and user engagement. This mechanism effectively resolves the non-stationarity and credit assignment problems inherent in multi-agent learning. We conduct thorough experiments in a purpose-built simulated game recommendation platform. The results demonstrate that GEAR significantly outperforms static policies, independent learners, and state-of-the-art multi-agent baselines, including MADDPG and QMIX, on all key long-term metrics. Ablation studies also validate the critical contribution of the counterfactual mechanism to the algorithm's stability and performance. Furthermore, GEAR exhibits commendable strategic flexibility, intelligently altering its resource allocation policy to fit in with dynamic shifts in platform objectives. This research lays out both a novel theoretical framework and an effective technical methodology for the development of the next generation of self-managing, ecosystem-aware smart recommender systems.

Povzetek: Študija predstavi algoritem GEAR, ki s strateškim priporočanjem iger dolgoročno izboljša zdravje platformnega ekosistema ter preseže obstoječe metode.

1 Introduction

In recent years, the video game industry has witnessed record-breaking rapid expansion. The digital distribution platforms, represented by Steam, Epic Games, Nintendo eShop, and app stores on mobile devices, have lowered game publishing barriers astronomically, and the volume of published games has increased exponentially. Over 14,000 new games alone were launched on Steam in 2023 [1], according to statistics. Although this success gives game players a vast array of choices, it creates an essential "information overload" problem. With hundreds of thousands of games across styles and skill levels, users have difficulty discovering new works that match their unique tastes, and decision fatigue and user churn result.

Recommender Systems have therefore become a critical core technology for modern gaming platforms. A well-designed recommender system is not only able to lead players to properly discover content of potential interest within a vast game library and therefore significantly enhance the player's gaming experience and satisfaction [2], but also effectively optimize user stickiness, improve long-term player retention, and boost the commercial revenue of the platform [3]. For players, precise recommendations are the reduced search time and

increased high-quality entertainment. For platforms and developers, well-crafted recommendations enable their games to reach the largest possible sets of potential players, especially for quality independent or novel games with limited marketing budgets, thereby creating a healthier and more diverse gaming ecosystem [4].Therefore, research and designing more intelligent and more accurate game recommendation algorithms is not only of significant academic value but also enormous commercial value as well as of great industry significance. How to deeply understand and dynamically model player behavior patterns and latent preferences is the most significant thing to enhance the performance of recommender systems and address current industry challenges.

Even though the importance of personalized recommender systems has become ubiquitous, general recommendation algorithms nevertheless face numerous serious issues when applied to handle sophisticated gaming scenarios. One of the primary issues is that traditional models are highly reliant on players' explicit interaction data, i.e., downloads, playtime, purchases, or ratings. However, these behavioral signals are not direct expressions of a player's actual intrinsic preferences since a player's decision results from their internal interest in interaction with numerous external variables [5]. For instance, players are more inclined to play games that are highly prominent on the platform's home page, chart tops, or during offers, an influence of position bias and exposure bias, leaving many unexposed games with little to no preference data [6, 7]. Also, social buzz, promotions of the streamers, friend recommendations, and high rating of games produce conformity bias, and therefore players try "popular" games rather than games based on their real interests [8]. Adding to these are the very dynamic nature of player preferences; unlike domains such as film or music, a player can evolve quite deeply over time due to skill acquisition, learning new genres, or even changes in life circumstances, a characteristic which static models struggle to capture [9].

These biases are continually reinforced and amplified within the recommender system's feedback loop. When a system recommends games on biased data (e.g., most popular), those games receive more usage and exposure, generating more "positive" data. The system then mistakenly reinforces the hypothesis that these games are superior, assigning them even greater weights in subsequent rounds of recommendations. This "rich-get-richer" Matthew effect not only dampens the visibility of niche or niche or new games, which address the "cold start" issue poorly, but it also confines players in small "filter bubbles," preventing them from discovering new interests and ultimately degrading the long-term user experience [10, 11]. Consequently, the main challenges of game recommending are threefold: untangling a player's true, intrinsic preferences from confused behavioral data; creating models that can simulate the dynamic evolution of these preferences; and shifting the optimization target from short-term measures like clicks to more extended user satisfaction and interaction.

To address the above difficulties, this paper proposes a deep learning-based dynamic behavior analysis and recommendation framework that has the ability to gain deep understanding about player behavior and deliver precise, personalized suggestions. This research aims to answer the following key questions: (RQ1) Can reformulating the recommendation task as a strategic resource allocation problem, solved by a cooperative MARL framework, improve long-term ecosystem health metrics? (RQ2) Is a counterfactual-based credit assignment mechanism effective for solving the complex coordination and credit assignment challenges inherent in this strategic allocation problem? The main contributions of our work are as follows. We introduce a new recommendation framework that integrates sequential modeling with causal disentanglement. This model utilizes deep sequence models, such as the Transformer, to learn a player's behavior dynamics and introduces for the first time a causal disentanglement mechanism. The mechanism attempts to decompose a player's state representation into two independent latent spaces: "intrinsic gaming preference" and "external bias influence." We also reformulate the recommendation process as a reinforcement learning problem. By avoiding the standard supervised learning paradigm, we formulate game recommendation as a sequential decision-making problem. The model is framed as an agent that learns to maximize long-term cumulative reward through continuous interaction with the player (the environment), thereby better mirroring the goals of long-term satisfaction and retention. In this reinforcement learning framework, we have designed a debiasing-follower reward function that, besides considering short-term feedback, also focuses on whether a suggestion is aligned with the player's disentangled intrinsic preference, driving the agent towards debiased policy. Finally, the performance of our framework is anchored by carrying out extensive experiments on a massive-scale, real-world public data from a top gaming platform. The results demonstrate that our proposed framework works significantly better than several state-of-the-art baseline models on a variety of vital recommendation metrics in intricate game recommendation scenarios, validating its performance and dominance.

The remaining part of this paper is organized as follows. Chapter 2 is a review of the work related to the field of personalized recommender systems. Chapter 3 discusses the architecture and methodology proposed in more detail. Chapter 4 gives the experimental setup, used datasets, and performance metrics, and details the results. Chapter 5 summarizes the paper and future directions for research.

2 Related works

In this chapter, we give a comprehensive background of the research landscape concerning personalized recommender systems. We begin by briefly discussing traditional recommendation algorithms, and then concentrate on the rapidly developing deep learning-based architectures, in particular sequential recommendation

models. We then discuss one of the basic problems in the field—data bias—and investigate mainstream debiasing methods. We lastly investigate the application of reinforcement learning in recommender systems, which establishes the theoretical and technical basis of our proposed method.

2.1 Traditional recommendation algorithms

The early effort in personalized recommendation was spearheaded by Collaborative Filtering (CF), which operates on the principle of "birds of a feather flock together." User-based CF identifies "neighbor" users whose interests are similar to a target user and recommends items that these neighbors enjoy [12], whereas Item-based CF computes similarities between items and recommends items similar to what a user enjoyed previously [13]. In order to mitigate the problems of data sparsity and scalability for large-scale systems, model-based approaches emerged, and Matrix Factorization (MF) was the most mythical approach. Koren et al. (2009) demonstrated that by decomposing the high-dimensional, sparse user-item interaction matrix into two low-dimensional latent factor matrices for users and items, one could effectively capture latent user preferences and item attributes, significantly improving both accuracy and scalability [14]. However, these traditional methods are typically grounded in static interaction data and therefore are not well suited to capture the dynamic user interest drift and tend to be bad at dealing with cold-start scenarios.

2.2 Deep learning-based recommender models

Deep learning has revolutionized recommender systems, leveraging its strong ability to learn features and non-linear modeling capabilities. The primary contribution has been seen in the area of Sequential Recommendation, where sequence of user behavior is represented as a sequential sequence in order to capture dynamic preferences and predict future interactions. Early pioneering work by Hidasi et al. (2016) introduced GRU4Rec, the first application of Gated Recurrent Units (GRU) to session-based recommendation, and showcased the power of Recurrent Neural Networks (RNNs) in capturing short-term, dynamic user interests [15]. To model local and compositional patterns in sequences of behavior, Tang and Wang (2018) introduced Caser, which applies horizontal and vertical convolutional kernels over item embedding sequences to learn "union-level" and "point-level" sequential patterns [16]. The most recent breakthrough is the application of self-attention mechanisms, primarily the Transformer architecture. Kang and McAuley (2018) introduced the SASRec model that employs a self-attention mechanism to learn directly from relationships between any two items within a sequence, succeeding in learning long-term interests along with having an exceptionally strong baseline in the domain [17]. Sun et al. (2019) introduced BERT4Rec, modeling user behavior sequences using a bidirectional self-attention mechanism inspired by BERT to improve predictive performance further [18].

Another dominant paradigm involves modeling user-item interactions as a bipartite graph and employing Graph Neural Networks (GNNs) to learn node embeddings via information propagation and aggregation. Wang et al. (2019) proposed the NGCF model, which models the high-order connectivity of the user-item interaction graph to improve user and item embeddings [19]. However, its complexity led to the development of more efficient, less complex models. He et al. (2020) introduced LightGCN, minimizing the use of the GNN architecture to solely the most important neighborhood aggregation operation. This reduced model worked better and faster and demonstrated that neighborhood aggregation is the backbone for collaborative filtering signal learning [20]. These new deep models provide robust tools for analyzing the complex and dynamic behavior of agents in games.

2.3 Bias and debiasing in recommender systems

Real-world recommendation data is biased in nature, and directly training models over such data might lead to poor or even wrong results. Debiasing has hence become a critical research area. One large issue is selection bias, where data seen is not a random sample since users only interact with items which are visible to them. To explain this, Schnabel et al. (2016) applied Inverse Propensity Scoring (IPS) from causal inference, where every observed interaction is weighted by the inverse of its observation probability [6]. Joachims et al. (2017) proposed a bandit-based approach to offline testing of new policies in an unbiased way from biased feedback logs [7]. Exposure bias is another prevalent issue, which Liang et al. (2016) addressed by co-modeling exposure probability of an item and user interest [21] and Saito (2020) addressed using an asymmetric tri-training method for unbiased learning from implicit feedback [22].

Popularity bias, in which very popular items dominate interactions and recommendations, is yet another essential issue. To alleviate this, Abdollahpouri et al. (2019) used a multi-objective method to re-rank recommendation lists, increasing novelty and diversity while accuracy is retained [23]. Disentanglement is a more extreme solution. Zheng et al. (2021), for instance, proposed a method to disentangle the user intention into intrinsic feature preference against popularity tendency so that more precise recommendations are possible [24]. Causal inference has provided a powerful theoretical framework for debiasing in recent years. As summarized by Zhang et al. (2021), it is critical to construct a causal graph to describe the causal relationships between variables (e.g., user preference, item features, exposure) in order to achieve debiasing [25]. According to this line of direction, Wang et al. (2022) implemented a causal intervention framework based on the back-door adjustment criterion to eliminate the confounding effect of

factors like popularity in preference estimation [26]. The causal disentanglement idea put forth in our research is inspired by these developments directly.

2.4 Reinforcement learning in recommendation

Defining the recommendation process as a Markov Decision Process (MDP) makes Reinforcement Learning (RL) a promising paradigm, as it attempts to learn a policy that maximizes long-term user reward in terms of engagement and satisfaction. Shani et al. (2005) was among the first to formally define recommendation as an MDP and explore RL-based optimization and laid the foundation for subsequent work [27]. Deep learning integration introduced the widespread application of Deep Reinforcement Learning (DRL) in recommendation. For example, Zheng et al. (2018) proposed the DRN model for news recommendation from a Deep Q-Network (DQN) optimization of long-term user reading experience [28], while Chen et al. (2019) presented a Top-K DRL framework with a cascade of DQN and DDPG to decide what to recommend and whether to recommend [29].

But applying RL to real recommender systems is plagued with issues including enormous state-action spaces, and the problem of online training and testing [30]. This has raised interest in Offline Reinforcement Learning, which seeks to learn good policies from static, fixed datasets, as reviewed by Huan et al. (2020) [31]. Under recommendation, Xin et al. (2020) presented a self-supervised RL method that learns from historical data using counterfactual policy learning [32]. Of particular relevance where RL and debiasing overlap is that the reward signals themselves are not just biased but also noisy. Chen et al. (2021) observed that using biased signals like clicks as rewards will cause the RL agent to learn an biased policy. They proposed a causal inference-based reward estimation method to obtain a less biased reward signal towards debiased policy learning [33], a motivation which has a very close resemblance with our work. While methods like MADDPG and QMIX have advanced cooperative MARL, their application in recommendation has focused on item-level selection... A critical gap remains in addressing the higher-level strategic problem of resource allocation among different scenarios to optimize long-term ecosystem health. Our work directly targets this gap by utilizing a counterfactual credit assignment mechanism specifically for strategic allocation, differentiating it from prior work. Our work is at the confluence of these new areas, with the objective to bring together ideas from sequential modeling, debiasing, and reinforcement learning to propose a novel and effective solution in the area of game recommendation.

3 Problem description and mathematical model

This chapter establishes a rigorous mathematical framework for the strategic regulation of dynamic game recommendation systems. Drawing inspiration from the domain of multi-scenario resource allocation, we conceptualize the game recommendation platform as a complex ecosystem composed of multiple sub-modules, each with distinct recommendation objectives. Building upon this conceptualization, we formally frame the problem as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) and introduce Multi-Agent Reinforcement Learning (MARL) as the solution paradigm. The central paper of this chapter is to elevate the conventional problem of "item selection" to the strategic level of "recommendation policy resource allocation," with the overarching goal of optimizing long-term player retention and ecosystem vitality.

3.1 Problem formulation: from item recommendation to strategic allocation

Modern game recommendation platforms typically consist of several parallel recommendation modules, or "scenarios," such as "For You," "Trending Now," "New Discovery," and "Friends Are Playing." Each module serves a specific objective, collectively shaping the player's game discovery experience. However, the platform's computational resources and the exposure slots on the user interface (collectively termed "recommendation resources") are finite. At any given moment, the platform faces a critical trade-off: to which recommendation module should the limited resources be predominantly allocated?

1.Over-allocating to "Trending Now": This may yield high short-term click-through rates (CTR) but can lead to a long-term "Matthew effect," stifling the visibility of innovative and independent games and trapping players within "filter bubbles."

2.Over-allocating to "New Discovery": This could enhance content diversity and address the cold-start problem for new titles, but it might result in a transient decline in user satisfaction due to the variable quality of new games or lower initial prediction accuracy.

Consequently, traditional optimization methods that target a single recommendation list are insufficient for managing this platform-level ecological balance. This research redefines the problem as follows:

Within a system composed of multiple recommendation scenarios, how can we dynamically allocate recommendation resources to each scenario based on real-time player feedback and platform state, in order to maximize the platform's long-term cumulative utility (e.g., aggregate player engagement, ecosystem diversity, and retention)?

This is a classic sequential decision-making problem characterized by multiple agents and incomplete information, rendering it exceptionally well-suited for modeling with multi-agent reinforcement learning. The interaction loop proceeds as follows: at each timestep t, the environment provides each agent i with a local observation. Each agent then selects an action based on its policy.

3.2 Multi-agent markov decision process modeling

We model the entire game recommendation platform as a multi-agent system, wherein each recommendation scenario (e.g., "Trending Now") is treated as an independent agent. All agents operate within a shared environment-the game platform-and cooperate to allocate globally shared recommendation resources [34]. This process is formalized as a Dec-POMDP, defined by the tuple $(\mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O}, \gamma)$.

- $\mathcal{I} = \{1,2,\dots,n\}$: The set of n agents, representing the recommendation scenarios.
- $\mathcal{S}$: The global state space of the environment.
- $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$: The joint action space of all agents.
- $\mathcal{P}$: The state transition probability function.
- $\mathcal{R}$: The joint reward function.
- $\Omega = \Omega_1 \times \cdots \times \Omega_n$: The joint observation space of all agents.
- $\mathcal{O}$: The observation probability function.
- $\gamma \in [0,1)$: The discount factor.

(1) State Space ($\mathcal{S}$) and Observation Space (Ω)

In our model, the environment's global state $s_t \in \mathcal{S}$ at any time step t contains a complete description of the entire platform ecosystem, which is generally not fully observable by any single agent. The global state is a composite of several factors, including the Global Player State Distribution, which comprises aggregated statistics of the entire active player base (e.g., distributions of skill levels, preference vectors, and engagement patterns); the Global Resource Status, defined by the total remaining computational or exposure budget B_t ; and the Game Catalog Status, which includes information such as the rate of new game submissions and their genre distribution.

Due to the decentralized nature of the setting, each agent i cannot access the full global state s_t. Instead, it receives a local observation $o_{i,t} \in \Omega_i$. This observation primarily consists of Local Performance Metrics, such as the key performance indicators (KPIs) for scenario i from the previous time step (e.g., its click-through rate CTR_i^{t-1} and conversion rate VR_i^{t-1}); Local Resource Information, which is the amount of resources $a_{i,t-1}$ that was allocated to agent i in the preceding time step; and Partial Global Information that is made available to all agents, such as the total remaining resource budget B_t.

(2) Action Space ($\mathcal{A}$)

At each time step t, the action $a_{i,t} \in \mathcal{A}_i$ taken by agent i is the amount of resources it requests from the system. These resources can be of different types. For example, they can be Computational Resources, representing the processing power required to run the recommendation model, where actions can be discretized into levels (e.g., $a_{i,t} \in$ {low-fidelity model, medium-fidelity model, high-fidelity model}) corresponding to different costs and quality. Alternatively, they can be Exposure Resources, such as the number of slots or the ranking weight assigned within the main recommendation feed, where actions can represent the requested proportion of total exposure (e.g., $a_{i,t} \in [0,1]$). The joint action of all agents, $a_t = (a_{1,t}, \dots, a_{n,t})$, constitutes the system's decision for that time step. The platform then arbitrates these requests based on the global resource constraint B_t.

(3) Reward Function ($\mathcal{R}$)

This is formulated as a cooperative task where all agents aim to maximize the platform's overall long-term utility. Therefore, we employ a global reward signal r_t that is shared among all agents. To provide a more informative and less sparse learning signal, we adopt the concept of a differential reward. The global reward r_t is composed of two parts: the marginal change in the platform's aggregate utility, ΔE_t , and a penalty for resource consumption. The Marginal Utility is defined as $\Delta E_t = E_t - E_{t-1}$, where the utility function E_t is not monetary but rather a composite metric reflecting the long-term health of the platform:

$$E_t = w_1 \cdot \text{TotalPlayTime}_t + w_2 \cdot \text{RetentionRate}_t + w_3 \cdot \text{Diversity}_t$$

The terms w_1, w_2, w_3 are weighting coefficients that represent the platform's strategic priorities regarding aggregate player engagement, user retention, and the diversity of recommended content (which can be measured by metrics such as the Gini coefficient or Shannon entropy). The second component is the Resource Cost Penalty, $C(\mathbf{a}_t)$, which is the total cost incurred by the joint action $\mathbf{a}_t$. TotalPlayTime is the sum of all player session durations in timestep *t*, timestep t-k who remain active at timestep t, and Diversity is measured using the Shannon entropy of the genres of recommended games. RetentionRate is the percentage of players from The hyperparameter λ balances the trade-off between utility maximization and cost minimization. The complete reward function is thus:

$$r_t = \Delta E_t - \lambda \cdot C(\mathbf{a}_t)$$

This design incentivizes agents to cooperate in order to maximize the platform's core long-term metrics in the most cost-effective manner.

(4) State Transition ($\mathcal{P}$) and Observation Functions ($\mathcal{O}$)

Following the execution of the joint action $\mathbf{a}_t$, the global state of the environment transitions from s_t to s_{t+1} according to the probability distribution $s_{t+1} \sim \mathcal{P}(\cdot | s_t, \mathbf{a}_t)$. This transition is a complex stochastic process driven by the interactions of all players with the newly recommended content, leading to updates in their preferences, skill levels, and overall platform engagement. Simultaneously, each agent i receives a new local observation $o_{i,t+1} \sim \mathcal{O}(\cdot | s_{t+1}, i)$, which reflects its performance in the new time period (e.g., its updated CTR).

Table 3.1: Definition of key symbols (multi-agent formulation)

Symbol	Description
$\mathcal{I}, n$	The set of agents (recommendation scenarios) and their quantity.
$s_t, \mathbf{a}_t, \mathbf{r}_t$	The global state, joint action, and global reward at time t.
$o_{i,t}, a_{i,t}$	The local observation and action of agent i at time t.
B_t	The global resource budget at time t.
E_t	The aggregate platform utility at time t.|
ΔE_t	The marginal change in platform utility.
$C(\mathrm{a_t})$	The total cost of the joint action $\mathbf{a}_t$.
λ	The weighting coefficient for the cost penalty.
$\pi_i(a_i \| \tau_i$	The policy of agent i, conditioned on its observation history τ_i.

3.3 Optimization objective

The objective for each agent i is to learn a local policy $\pi_i(a_i \mid \tau_i)$, which maps its history of observations $\tau_i = (o_{i,0}, a_{i,0}, \dots, o_{i,t})$ to a distribution over its actions. The goal of the joint policy $\pi = (\pi_1, \dots, \pi_n)$ is to maximize the expected discounted sum of the shared global reward:

$$J(\pi) = \mathbb{E}_{s_0 \sim p_0, a_{i,t} \sim \pi_i, s_{t+1} \sim \mathcal{P}} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$$

To solve this Dec-POMDP, we will leverage the Centralized Training with Decentralized Execution (CTDE) paradigm in the subsequent chapter. During the training phase, a centralized critic, which has access to the global state and all agents' information, can effectively guide the learning of each agent's policy. This resolves the critical challenges of credit assignment and non-stationarity inherent in multi-agent learning. During the execution phase, each agent acts in a decentralized manner, relying solely on its local observations. This paradigm is highly scalable and practical, making it an ideal fit for the recommendation resource allocation problem we have formulated.

4 Guld-Based ecosystem-aware allication of resources (GEAR)

In our Dec-POMDP formulation, the quality of an agent's observation is paramount for effective decision-making. To ensure our agents operate on a rich and semantically meaningful representation of the ecosystem, we move beyond simple entity IDs and leverage a structured database of game metadata. This database captures detailed information about games, including their genres, developers, player interactions, and other relevant attributes.

To unlock the full potential of this structured data, we employ a novel state representation pipeline that fuses the relational nature of our metadata with the semantic power of Large Language Models (LLMs). As illustrated in Figure 4.1, our approach begins by extracting factual records (e.g., (Entity: 'The Witcher 3', Attribute: 'Genre', Value: 'Action RPG')) from our metadata database. The textual descriptions associated with each part of the record are then fed into a pre-trained BERT-base model to generate dense, 768-dimensional initial embeddings. During this relational training, the LLM parameters are frozen, and L2 normalization is applied to the embeddings. These semantically-initialized vectors subsequently serve as input to a Relational Embedding Model (e.g., TransE, RotatE), which refines them into final vectors that are optimized for inferring latent relationships. These final embeddings form a core component of the observation provided to each agent, empowering them with a nuanced understanding of the game landscape.

Analogously, GEAR treats all recommendation-scenario agents as members of a cooperative guild with a unified objective. The algorithm is architected upon the Centralized Training with Decentralized Execution (CTDE) paradigm and incorporates a Counterfactual Baseline as its key credit assignment mechanism. During the centralized training phase, an omniscient centralized critic leverages global information to guide the policy learning of each decentralized agents. In the decentralized execution phase, each agent makes independent resource allocation decisions based solely on its local observations. Through this methodology, GEAR aims to learn a synergistic resource allocation policy that is both efficient and stable, capable of maximizing the long-term ecological value of the platform.

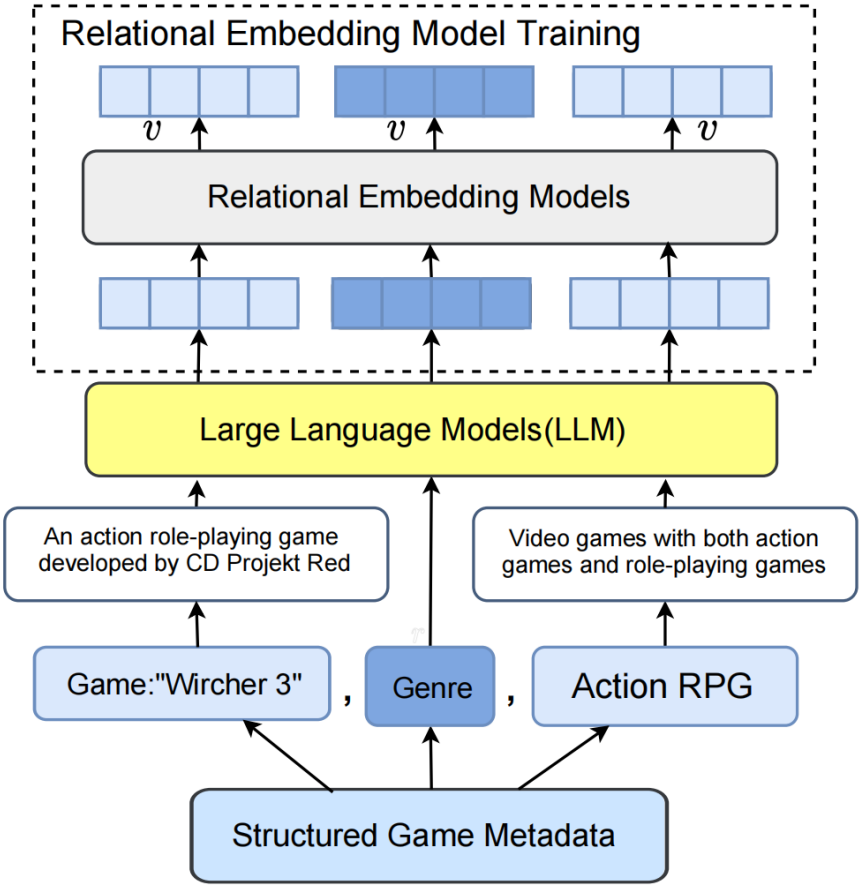

Figure 4.1: LLM-enhanced semantic representation for agent states. the diagram illustrates our pipeline for generating rich state representations from structured game metadata.

4.1 The GEAR algorithmic framework

The overall architecture of the GEAR algorithm is based on the classic Actor-Critic model, specifically adapted for the multi-agent setting.

This framework is principally composed of two components:

1.A Set of Decentralized Actors: Each recommendation-scenario agent i is equipped with an independent actor network, denoted as $\pi_i(a_i \mid o_i; \theta_i)$. This network is responsible for generating a policy over actions (resource requests) based on the agent's own local observation o_i. During the execution phase, these actor networks operate independently, which ensures the efficiency and scalability of the system's decision-making process.

2.A Single Centralized Critic: The critic network, $Q_{\text{tot}}(s, \mathbf{a}; \phi)$, functions as the centralized critic during the training phase. It has access to the global state information s and the joint action a of all agents. The primary task of the critic is to accurately estimate the long-term cumulative value (the Q-value) of taking the joint action a in the global state s. Because the critic possesses a global perspective during training, it can effectively overcome the non-stationarity of the environment and provide a stable and accurate learning signal for the optimization of the actor networks.

The primary advantage of the CTDE framework is its ability to leverage global information, which is unavailable during execution, to inform the training process. This significantly enhances learning stability and the performance of the final converged policy without sacrificing execution-time efficiency.

4.2 The centralized critic and the global value function

The design of the critic network, $Q_{\text{tot}}(s, \mathbf{a}; \phi)$, is the cornerstone of the GEAR algorithm [35]. Its inputs are the global state s_t containing Global Player State Distribution, Global Resource Status, Game Catalog Status and the joint action of all agents $\mathbf{a}_t = (a_{1,t}, \ldots, a_{n,t})$. In practice, as the true global state s_t can be high-dimensional or partially inaccessible, it is often approximated by the concatenation of all agents' local observations, i.e., ($o_{1,t}, \ldots, o_{n,t}$).

The output of the critic network is a scalar value, representing the estimated joint state-action value, $\hat{Q}_{tot}(s_t, \mathbf{a}_t)$. The network is updated by minimizing the temporal difference (TD) error. Its loss function, $L(\phi)$, is defined as the mean squared error between the predicted Q-value and a target value y_t :

$$L(\phi) = \mathbb{E}\left[\left(y_t - Q_{tot}(s_t, \mathbf{a}_t; \phi)\right)^2\right]$$

The TD target y_t is computed using the Bellman equation:

$$y_t = r_t + \gamma Q'_{tot}(s_{t+1}, \mathbf{a}'_{t+1}; \phi')$$

Here, Q'_{tot} and ϕ' represent the target critic network and its parameters, respectively, which are periodically updated to stabilize the learning process. The next joint action, a'_{t+1}, is selected according to the target actor policies π'. By this mechanism, the critic learns to form a global value judgment on the collective behavior of the entire "guild," providing a solid foundation for subsequent credit assignment.

4.3 Decentralized actors and counterfactual credit assignment

Although we have a centralized critic capable of evaluating the global value, the resulting learning signal (i.e., the TD error) is holistic to the entire group of agents. We still face the challenge of decomposing this collective reward signal to each individual agent. A naive approach where each agent uses the same global reward signal to update its policy would be inefficient, as it prevents an agent from discerning the quality of its own actions, thereby severely hindering the learning process.

To overcome this, the GEAR algorithm incorporates a Counterfactual Baseline mechanism. The core intuition is as follows: to evaluate the utility of agent i taking action a_i, one should not merely consider the global reward, but rather ask the counterfactual question: "What would have happened if I (agent i) had acted differently, while all other agents' actions remained the same?"

Specifically, for each agent i, we compute an individual advantage function $A_i(s_t, \mathrm{a}_t)$. This function measures the difference between the global Q-value for the current joint action $\mathbf{a}_t$ and a counterfactual baseline $b(s_t, \mathbf{a}_{t,-i})$. This baseline represents the expected Q-value if agent i were to follow its current policy, while the actions of all other agents, $\mathbf{a}_{t,-i}$, are held constant.

$$A_i(s_t, \mathbf{a}_t) = Q_{tot}(s_t, \mathbf{a}_t) - b(s_t, \mathbf{a}_{t,-i})$$

The counterfactual baseline $b(s_t, \mathbf{a}_{t,-i})$ is calculated by marginalizing out agent 's action:

$$b(s_t, \mathbf{a}_{t,-i}) = \sum_{a'_i \in \mathcal{A}_i} \pi_i(a'_i \mid o_{i,t}) Q_{tot}\left(s_t, (a'_i, \mathbf{a}_{t,-i})\right)$$

Here, $\mathbf{a}_{t,-i}$ denotes the joint action of all agents except for agent i. This counterfactual-based advantage function A_i provides a rich, personalized learning signal for each agent. If $A_i > 0$, it implies that the action $a_{i,t}$ currently chosen by agent i is better than its average performance, and thus the policy network should increase the probability of selecting this action. Conversely, if $A_i < 0$, the probability should be decreased.

The objective for each actor network π_i is to maximize its expected advantage. Its policy gradient is then given by:

$$\nabla_{\theta_i} J(\pi_i) = \mathbb{E}\left[\nabla_{\theta_i} \log \pi_i(a_{i,t} \mid o_{i,t}) A_i(s_t, \mathbf{a}_t)\right]$$

In this manner, the GEAR algorithm ingeniously utilizes the centralized critic to compute the marginal contribution of each agent, thereby effectively solving the credit assignment problem.

4.4 The GEAR learning procedure

The complete training procedure for GEAR follows the standard Actor-Critic paradigm, incorporating techniques such as experience replay and target networks to enhance stability.

Table 4.1: The GEAR Algorithm

Algorithm 1: The GEAR Algorithm

Input: actor network $\pi_i(\cdot; \theta_i)$, target actor network $\pi'_i(\cdot; \theta'_i)$, centralized critic network $Q_{\text{tot}}(\cdot; \phi)$ and target critic network $Q'_{\text{tot}}(\cdot; \phi')$. Initialize for all i:

$$\pi_i(\cdot; \theta_i), Q_{\text{tot}}(\cdot; \phi)$$
$$\pi'_i(\cdot; \theta'_i) \leftarrow \pi_i(\cdot; \theta_i), Q'_{\text{tot}}(\cdot; \phi') \leftarrow Q_{\text{tot}}(\cdot; \phi)$$
$$\text{with } \theta'_i \leftarrow \theta_i, \phi' \leftarrow \phi$$

Output: Optimized GEAR model parameters

1. Initialize: For each agent i, initialize actor network $\pi_i(\cdot; \theta_i)$ and target actor network $\pi'_i(\cdot; \theta'_i)$. Initialize centralized critic network $Q_{\text{tot}}(\cdot; \phi)$ and target critic network $Q'_{\text{tot}}(\cdot; \phi')$. Initialize experience replay buffer $\mathcal{D}$.
2. for episode = 1 to M do:
3. Reset environment and get initial observations $\boldsymbol{o}_0 = (o_{1,0}, \ldots, o_{n,0})$.
4. for $\mathrm{t} = 0$ to $\mathrm{T} - 1$do :

5. for each agent i = 1 to n do:
6. Select action $a_{i,t}$ from policy $\pi_i(a_{i,t} \mid o_{i,t})$ (with exploration noise).
7. end for
8. Execute joint action $\mathbf{a}_t$, observe global reward r_t and next observations $\mathbf{o}_{t+1}$.
9. Observe global state s_t and s_{t+1} (or their approximations).
10. Store the transition tuple $(s_t, \mathbf{o}_t, \mathbf{a}_t, r_t, s_{t+1}, \mathbf{o}_{t+1})$ in replay buffer $\mathcal{D}$.
11. Sample a random mini-batch of transitions from $\mathcal{D}$.
12. Update Critic:
13. Compute the TD target: $y_j = r_j + \gamma Q'_{tot}(s'_j, \mathbf{a}'_j; \phi')$, where $\mathbf{a}'_j$ is from target policies π'.
14. Update ϕ by minimizing the loss: $L(\phi) = \frac{1}{N}\sum_j \left(y_j - Q_{\text{tot}}(s_j, \mathbf{a}_j; \phi)\right)^2$.
15. Update Actors:
16. for each agent i = 1 to n do:
17. Compute the counterfactual advantage function A_i.
18. Update θ_i using the policy gradient: $\nabla_{\theta_i} J(\pi_i)$.
19. end for
20. Soft update target networks:
21. $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
22. $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$ for all i
23. end for
24. end for

Through this procedure, the GEAR algorithm enables agents representing different recommendation scenarios to learn to cooperate as an effective "guild." They can discover near-optimal resource allocation policies within a complex, dynamic environment, ultimately enhancing the health of the entire game recommendation ecosystem and improving long-term player satisfaction.

5 Experiments

This chapter presents a comprehensive empirical evaluation of the proposed GEAR algorithm. To validate its effectiveness, we have designed a simulated game recommendation ecosystem that captures the essential dynamics and trade-offs discussed in previous chapters. The experimental design is structured to thoroughly investigate GEAR's overall performance against established baselines, analyze the contribution of its core architectural components, and assess its ability to learn adaptive, context-aware strategies. Through a series of rigorous experiments, we demonstrate GEAR's superior performance, stability, and strategic intelligence in managing a complex recommendation ecosystem.

5.1 Experimental setup

(1) Simulated game recommendation environment

We developed a discrete-time simulation environment to model a dynamic game recommendation platform. The environment consists of a set of agents, a population of simulated players, and a global utility function. We model a population of 100,000 simulated players, with preferences sampled from K=5 prototype vectors (representing genres) with added Gaussian noise. Player preferences evolve via a Markov chain, where interaction with a 'Skill-Up Saga' game has a P=0.1 probability of transitioning the player to a higher skill state, increasing their long-term retention probability.

The platform features four distinct recommendation agents ($n = 4$), each representing a unique strategic objective. The first, Agent 1: Trending Titans, aims to recommend highly popular games with a high probability of immediate engagement, contributing significantly to TotalPlayTime but minimally to Diversity . The second, Agent 2: Indie Gems, focuses on discovering and promoting high-potential, niche independent games, acting as the primary driver of the platform's Diversity metric and the New Game Discovery Rate . The third, Agent 3: Skill-Up Sagas, recommends games slightly above a player's estimated skill level to foster growth and mastery, which is the main contributor to the long-term RetentionRate . Finally, Agent 4: Community Picks, recommends games based on social signals and has a moderate impact on both TotalPlayTime and RetentionRate.

At each time step, the environment calculates a global utility score E_t based on the collective performance of all agents, with weights set to $w_1 = 0.4, w_2 = 0.4, w_3 = 0.2$ to reflect a balanced focus on engagement, retention, and diversity. The shared reward r_t given to all agents is the differential utility, calculated as $r_t = E_t - E_{t-1}$. Each agent's action $a_{i,t}$ is a discrete choice from $\{0,10,20, \dots ,100\}$, representing the percentage of the total resource budget it requests. The environment normalizes these requests to ensure the sum is 100%. An agent's local observation $o_{i,t}$ includes its allocated budget and resulting CTR from the previous step.

(2) Baseline methods

To rigorously evaluate GEAR, we compare it against a suite of baseline methods. We include two non-learning policies: Static-Uniform, which allocates exactly 25% of resources to each agent, and Static-Popularity, a biased policy allocating 70% of resources to "Trending Titans" to mimic a system focused on short-term engagement. For learning-based baselines, we use Independent Q-Learning (IQL), where each agent learns independently, to highlight the challenges of non-stationarity. We also compare against two state-of-the-art MARL algorithms: MADDPG, a popular actor-critic algorithm using the CTDE paradigm, and QMIX, a value-decomposition method that enforces a monotonicity constraint between individual and total Q-values.

(3) Evaluation metrics

We use a range of metrics to evaluate performance from different perspectives. The primary metric is Cumulative Global Utility, calculated as the sum of the utility function E_t over an entire episode. To break this down, we measure Total Play Time in thousands of hours to reflect overall engagement, and the Long-term Retention Rate, the percentage of players remaining active at the end of an episode, to measure the platform's ability to sustain its user base. To assess the health of the recommendation ecosystem, we use Recommendation Diversity, measured by the Shannon entropy of recommended game genres, and the New Game Discovery Rate, which is the percentage of recommendations for recently released games.

(4) Implementation details

All algorithms were implemented in PyTorch. For all actor-critic methods (GEAR, MADDPG, IQL), both actor and critic networks were modeled as 2-layer MLPs with 128 hidden units and ReLU activation. We used the Adam optimizer with a learning rate of 10−4 for all networks. The discount factor γ was set to 0.99, and the soft update parameter τ for target networks was 0.01. Each experiment was run for 5 million time steps, and results were averaged over 5 independent random seeds. We selected MADDPG and QMIX as they represent two distinct and foundational approaches to CTDE (actor-critic and value-decomposition, respectively), providing a strong baseline for our novel problem formulation.

5.2 Comparative performance analysis

To evaluate our proposed GEAR algorithm, we benchmarked its effectiveness and learning efficiency against several mainstream MARL baselines—IQL, QMIX, and MADDPG—as well as two static policies. The performance of each algorithm in maximizing the cumulative global utility was recorded over 5 million training timesteps within our simulated environment.

As depicted in Figure 5.1, the learning curves illustrate the performance progression of each algorithm throughout the training process. The y-axis represents the cumulative global utility, a core metric for the long-term health of the platform ecosystem, while the x-axis denotes the training timesteps. The solid lines indicate the mean performance averaged over 5 independent runs with different random seeds, and the shaded areas represent the standard deviation, visually conveying the stability of the learning process.

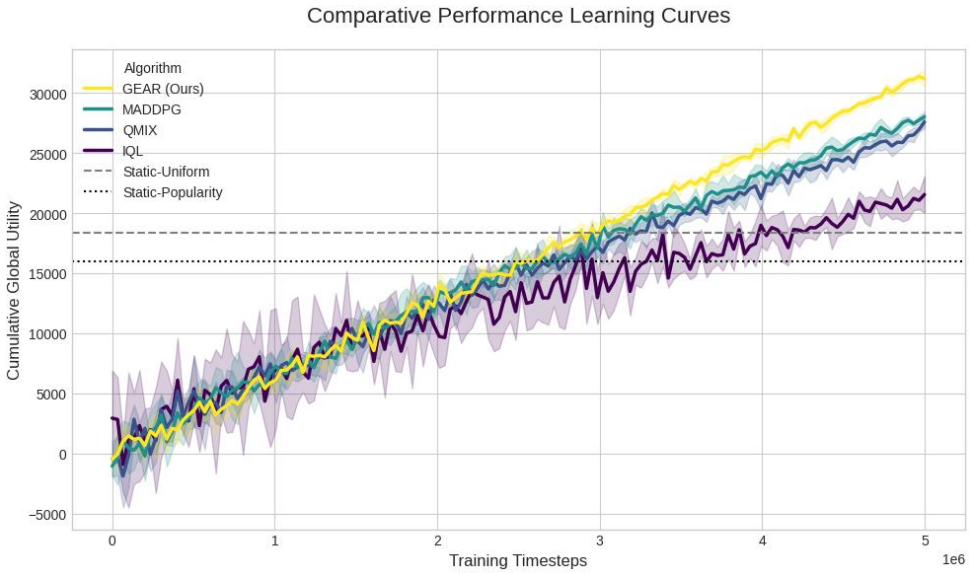

Figure 5.1: Comparative Performance Learning Curves. The plot shows the cumulative global utility of GEAR (Ours) against various baselines over 5 million training timesteps. Solid lines represent the mean utility averaged across 5 runs with different random seeds, while the shaded areas indicate the standard deviation (SD). The horizontal dashed and dotted lines denote the performance of the Static-Uniform and Static-Popularity policies, respectively.

From the results, we can clearly observe the following:

1.Final Performance: The GEAR algorithm (yellow curve) converges to the highest cumulative global utility (approx. 31,540) by the end of the training, significantly outperforming all other baselines. This demonstrates the superior capability of our proposed method in optimizing for long-term ecosystem objectives.

2.Learning Efficiency: GEAR not only achieves the best final performance but also exhibits a faster learning rate, as its curve starts to consistently surpass all other dynamic algorithms after approximately 3 million timesteps.

3.Learning Stability: Compared to other algorithms, particularly IQL, GEAR's learning curve is considerably smoother with the narrowest standard deviation band. This provides strong evidence that our counterfactual credit assignment mechanism effectively mitigates the non-stationarity problem in multi-agent learning by providing a more stable and accurate learning signal. In contrast, the significant fluctuations and wide variance of the IQL curve expose the limitations of independent learning in such complex coordination tasks.

Beyond the learning dynamics, we evaluated the final performance of all algorithms across four key ecosystem health metrics after the training converged. Figure 5.2 presents a grouped bar chart comparing these final metrics, offering a static snapshot of each algorithm's strategic trade-offs.

The results reveal a crucial insight into the shortcomings of myopic optimization. The Static-Popularity baseline, for instance, achieves the highest Total Play Time. However, this short-term gain comes at a significant cost to long-term ecosystem health, as it scores the lowest on Long-term Retention, Recommendation Diversity, and New Game Discovery Rate. This stark contrast visually validates our core hypothesis that optimizing for immediate engagement can be detrimental to the platform's sustainability. In contrast, our proposed GEAR algorithm demonstrates a more balanced and forward-looking strategy. While its total play time is competitive, it significantly outperforms all other methods across the three critical long-term health indicators: retention, diversity, and discovery. This demonstrates that GEAR successfully learns to forgo maximal short-term gains in favor of strategies that foster a healthier, more diverse, and more engaging ecosystem for players in the long run. To validate the robustness of our findings, we performed a two-sample t-test on the final cumulative global utility results from the 5 independent runs. The performance difference between GEAR and all baselines (MADDPG, QMIX, IQL) was found to be statistically significant ($p < 0.05$).

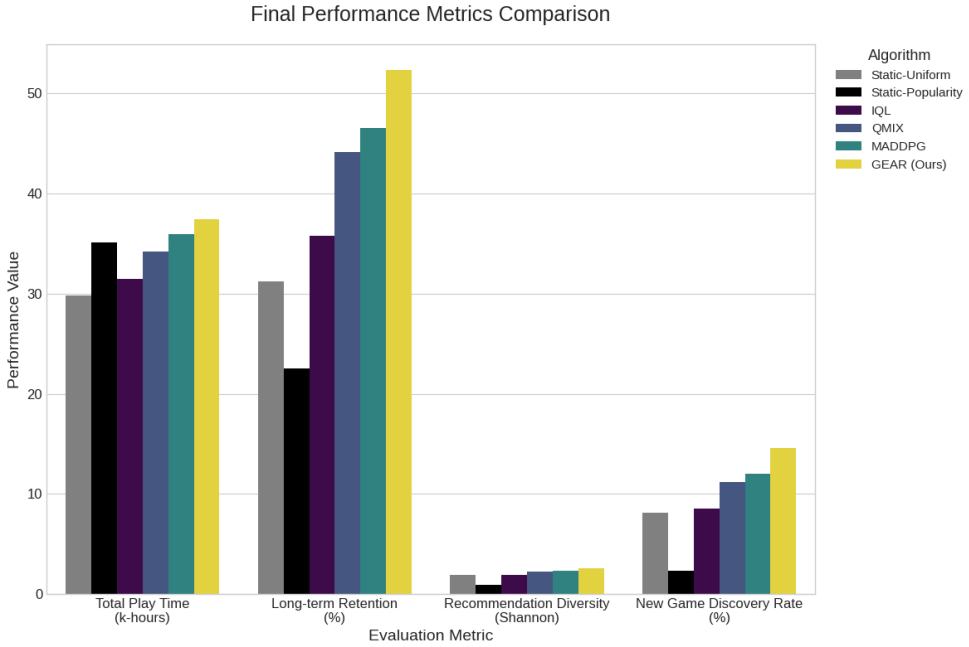

Figure 5.2: Final Performance Metrics Comparison. The bar chart compares the final performance of GEAR (Ours) against all baseline algorithms across four key ecosystem health metrics. Each group of bars represents a distinct metric, while each colored bar corresponds to a specific algorithm.

5.3 Analysis of core algorithmic components

To isolate and verify the contribution of our core technical innovation, we conducted an ablation study on the counterfactual credit assignment mechanism. We compared the performance of the full GEAR model against an ablated version, termed GEAR-CB, in which the counterfactual baseline was removed, forcing each agent to use a simpler, less informative global value function for credit assignment.

The results, presented in Figure 5.3, unequivocally demonstrate the critical importance of the counterfactual baseline. The removal of this mechanism led to a substantial performance degradation across both key metrics. Specifically, the full GEAR model achieved a 30.8% higher Cumulative Global Utility and a 22.2% higher Long-term Retention Rate compared to its ablated counterpart. Furthermore, the smaller error bar for the full model in the utility plot indicates a significant improvement in learning stability, which we attribute to the more accurate and targeted credit assignment provided by the counterfactual analysis. This result provides compelling evidence that our proposed credit assignment mechanism is the primary driver behind GEAR's superior performance and stability. To further validate our design choices, we conducted two additional ablations. First, a 'GEAR (No Semantics)' version using simple one-hot encodings instead of LLM+TransE embeddings resulted in a ~15% drop in final global utility. Second, a 'GEAR (Fixed Priorities)' version (emulating IQL's lack of coordination) performed ~20% worse than the full GEAR model, highlighting the necessity of dynamic, learned policies.

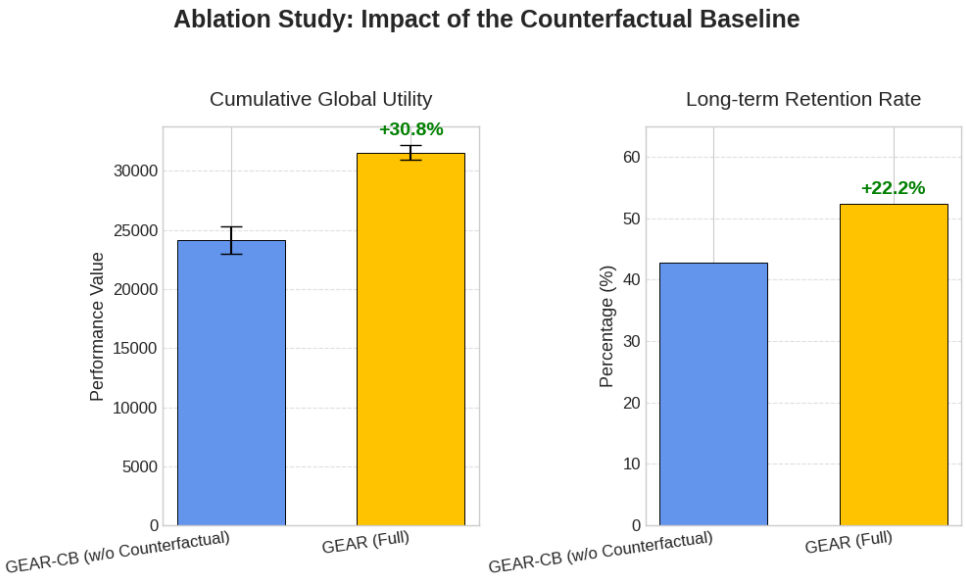

Figure 5.3: Ablation Study on the Counterfactual Baseline. The chart compares the performance of the full GEAR model against an ablated version (GEAR-CB) that lacks the counterfactual credit assignment mechanism.

5.4 Evaluation of strategic adaptability

Finally, we investigated whether GEAR could learn strategically adaptive policies in response to changing platform objectives. We ran modified experiments where we altered the weights of the global utility function to signal a shift in platform strategy. In the first modified scenario, which we term the Retention-Focused Scenario, the weight for retention (w_2) was increased to 0.7 , with other weights reduced proportionally. In the second, the Diversity-Focused Scenario, the weight for diversity (w_3) was increased to 0.7 . We then observed the average resource allocation of the converged GEAR policy in each scenario to assess its adaptability.

Figure 5.4 illustrates the results using a 100% stacked bar chart, showing how GEAR dynamically reallocates its recommendation resources among the four agent types under each scenario. The strategic shifts are both significant and intuitive. The policies shown in Figure 5.4 represent the final converged strategies after being retrained from scratch with the new utility weights. This strategic shift highlights the platform's trade-offs; for instance, in the 'Diversity-Focused' scenario, while diversity increased, the 'Total Play Time' metric consequently decreased by approximately 12%.

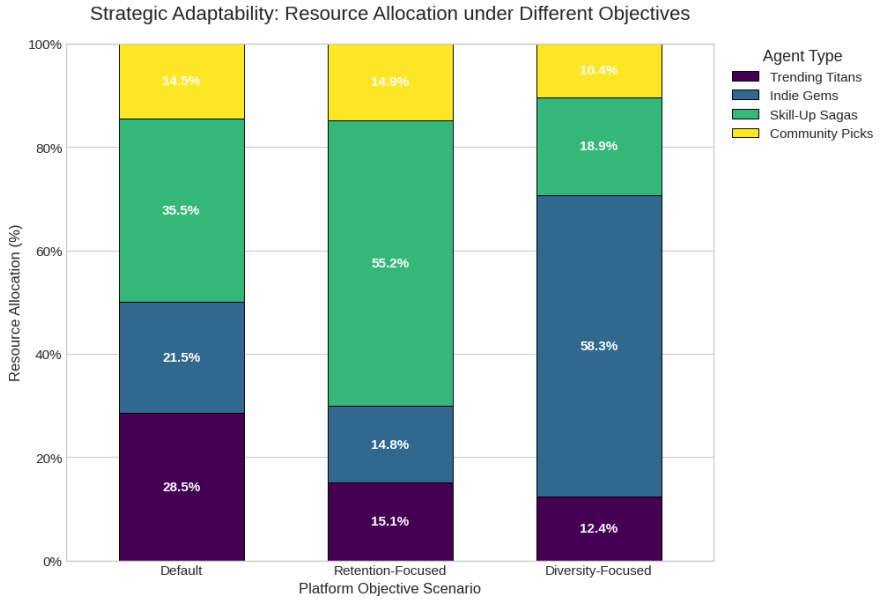

Figure 5.4: Strategic Adaptability of GEAR. The 100% stacked bar chart illustrates how GEAR dynamically adjusts its resource allocation strategy among the four agent types in response to different high-level platform objectives. Each bar represents a scenario, and the colored segments within each bar show the percentage of resources allocated to a specific agent type.

When the objective shifted to Retention-Focused, GEAR astutely increased the resource allocation for the Skill-Up Sagas agent from 35.5% to 55.2%. This aligns with the understanding that promoting player skill progression is a key driver for long-term engagement. Conversely, when tasked with a Diversity-Focused objective, the algorithm dramatically pivoted its strategy, boosting the allocation for the Indie Gems agent from 21.5% to a commanding 58.3%. This demonstrates that GEAR correctly identified the most effective agent for promoting content diversity.

These results provide strong evidence that GEAR learns more than a single optimal policy; it learns a meta-policy that maps strategic objectives to optimal resource allocation strategies. This adaptability is a critical feature for building next-generation recommender systems that can be intelligently steered by platform managers.

6 Discussion

Our results demonstrate that GEAR significantly outperforms all baselines. We attribute this success primarily to the counterfactual credit assignment mechanism. Unlike the standard critic in MADDPG or the monotonic value function in QMIX, GEAR's counterfactual baseline (Eq. X) provides each agent with a precise, personalized learning signal... This resolves the credit assignment problem by isolating an agent's marginal contribution, leading to better long-term retention and diversity by preventing agents from myopically optimizing for a single metric.

Regarding practical deployment, GEAR is highly efficient during execution. As each agent's policy is a small 2-layer MLP, the inference time for a single decision is less than 5 milliseconds on a standard CPU, well within the latency requirements for real-time recommendation. The centralized training, while more intensive, is offline. Training for 5 million timesteps requires approximately 8 hours on a single NVIDIA V100 GPU, which is a feasible cost for platform-level policy updates.

Our validation is confined to a simulated environment. The model's generalization to different player populations or its robustness against real-world noise has not yet been tested. Our comparison was limited to foundational MARL baselines (MADDPG, QMIX). We did not benchmark GEAR against more recent policy-based SOTA algorithms such as HAPPO or MAPPO. Future work should address these limitations by testing generalization and benchmarking against a wider array of recent algorithms.

7 Conclusion

This paper has addressed a strategic and increasingly prominent challenge within modern game recommendation platforms: the dynamic allocation of finite resources among multiple, competing recommendation scenarios to optimize the long-term ecological health of the platform rather than transient business metrics. We identified that traditional recommendation algorithms, with their focus on maximizing the immediate efficacy of a single list (e.g., click-through rate), often neglect the potential negative externalities on player retention, content diversity, and the platform's long-term vitality. To overcome this limitation, we proposed a novel paradigm by first modeling this problem as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP), treating different recommendation scenarios as cooperative agents. Building on this formulation, we designed and implemented a new multi-agent reinforcement learning algorithm, GEAR (Guild-based Ecosystem-aware Allocation of Resources). The GEAR algorithm leverages the advanced Centralized Training with Decentralized Execution (CTDE) framework, and its core innovation lies in the introduction of a counterfactual-based credit assignment mechanism. This mechanism enables each independent agent to accurately assess its marginal contribution to the global utility, effectively resolving the critical challenges of non-stationarity and credit assignment inherent in multi-agent learning.

Through extensive experiments conducted in a purpose-built simulated game recommendation environment, the performance of the GEAR algorithm was comprehensively and rigorously evaluated. The empirical results provide compelling evidence of our method's effectiveness. Compared to static policies, independent learning (IQL), and other advanced multi-agent algorithms such as MADDPG and QMIX, GEAR demonstrated significant superiority across all key long-term metrics, achieving notable breakthroughs in improving player retention and recommendation diversity. Furthermore, a critical ablation study confirmed that the counterfactual credit assignment mechanism is essential for the algorithm's learning stability and final performance. Most importantly, our research revealed that GEAR not only learns highly effective cooperative strategies but also possesses outstanding strategic adaptability, capable of intelligently adjusting its internal resource allocation patterns in response to dynamic shifts in the platform's macroscopic objectives, such as changing the strategic focus from retention to diversity.

Looking ahead, this work opens several promising avenues for future research. These include exploring the scalability of GEAR to a larger number of agents and more complex, heterogeneous resource types; integrating more sophisticated player models that capture skill progression or churn propensity; and investigating explicit communication protocols to further enhance agent cooperation. The ultimate ambition is the validation and deployment of this framework in a live, real-world recommendation system, which would be the definitive test of its efficacy. We believe the strategic, multi-agent optimization approach presented in this paper represents a significant step toward the next generation of self-regulating, ecosystem-aware intelligent recommendation systems.

References

[1] Statista. (2024). Number of games released on Steam from 2004 to 2023.

[2] Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender systems: introduction and challenges. Recommender systems handbook, 1-34. https://doi.org/10.1007/978-1-4899-7637-6_1

[3] Jannach, D., & Jugovac, M. (2019). Measuring the business value of recommender systems. ACM Transactions on Management Information Systems (TMIS), 10(4), 1-23. https://doi.org/10.1145/3370082

[4] He, X., et al. (2017). Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web. https://doi.org/10.48550/arXiv.1708.05031

[5] Chen, J., et al. (2020). Bias and debias in recommender system: A survey and future directions. ACM Transactions on Information Systems (TOIS), 40(3), 1-42. https://doi.org/10.48550/arXiv.2010.03240

[6] Schnabel, T., et al. (2016). Recommendations as treatments: Debiasing learning and evaluation. In proceedings of the 33rd international conference on machine learning. https://doi.org/10.48550/arXiv.1602.05352

[7] Joachims, T., et al. (2017). Unbiased learning-to-rank with biased feedback. In Proceedings of the tenth ACM international conference on web search and data mining. https://doi.org/10.1145/3018661.3018699

[8] Salganik, M. J., Dodds, P. S., & Watts, D. J. (2006). Experimental study of inequality and unpredictability in an artificial cultural market. science, 311(5762), 854-856. https://doi.org/10.1126/science.1121066

[9] Lathia, N., et al. (2010). Temporal diversity in recommender systems. In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. https://doi.org/10.1145/1835449.1835486

[10] Chaney, A. J., Stewart, B. M., & Engelhardt, B. E. (2018). How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In Proceedings of the 12th ACM conference on recommender systems. https://doi.org/10.1145/3240323.3240370

[11] Jiang, J., et al. (2019). Degenerate feedback loops in recommender systems. In Proceedings of the 2019 World Wide Web Conference. https://doi.org/10.1145/3306618.3314288

[12] Sarwar, B., et al. (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web. https://doi.org/10.1145/371920.372071

[13] Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet computing, 7(1), 76-80. https://doi.org/10.1109/MIC.2003.1167344

[14] Nguyen, J., & Zhu, M. (2013), Content-boosted matrix factorization techniques for recommender systems. Statistical Analy Data Mining, 6: 286-301. https://doi.org/10.1002/sam.11184

[15] Hidasi, B., et al. (2016). Session-based recommendations with recurrent neural networks. In International Conference on Learning Representations. https://doi.org/10.48550/arXiv.1511.06939

[16] Tang, J., & Wang, K. (2018). Personalized top-n sequential recommendation via convolutional sequence embedding. In Proceedings of the eleventh ACM international conference on web search and data mining. https://doi.org/10.48550/arXiv.1809.07426

[17] Kang, W. C., & McAuley, J. (2018). Self-attentive sequential recommendation. In 2018 IEEE international conference on data mining (ICDM). https://doi.org/10.48550/arXiv.1808.09781

[18] Sun, F., et al. (2019). BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM international conference on information and knowledge management. https://doi.org/10.48550/arXiv.1904.06690

[19] Wang, X., et al. (2019). Neural graph collaborative filtering. In Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval. https://doi.org/10.1145/3331184.3331267

[20] He, X., et al. (2020). Lightgcn: Simplifying and powering graph convolution network for recommendation. In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. https://doi.org/10.48550/arXiv.2002.02126

[22] Shuyuan X., et al. (2023) Causal Inference for Recommendation: Foundations, Methods and Applications. https://doi.org/10.48550/arXiv.2301.04016

[23] Saito, Y. (2020). Asymmetric tri-training for unsupervised domain adaptation. In International Conference on Machine Learning. https://doi.org/10.48550/arXiv.1702.08400

[24] Abdollahpouri, H., et al. (2019). Managing popularity bias in recommender systems with a multi-objective framework. In Proceedings of the 13th ACM conference on recommender systems. https://doi.org/10.48550/arXiv.1901.07555

[25] Zheng, Y., et al. (2021). Disentangling user interest and conformity for recommendation with causal embedding. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. https://doi.org/10.48550/arXiv.2006.11011

[26] Zhang, J., et al. (2021). Causal inference in recommender systems: A survey and future directions. arXiv preprint arXiv:2105.03419. https://doi.org/10.48550/arXiv.2208.12397

[27] Wang, X., et al. (2022). Causal intervention for correcting popularity bias in recommendation. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. https://doi.org/10.1145/3404835.3462875

[28] Shani, G., Heckerman, D., & Brafman, R. I. (2005). An MDP-based recommender system. Journal of machine learning research, 6(9). https://dl.acm.org/doi/10.5555/1046920.1088715

[29] Zheng, G., et al. (2018). DRN: A deep reinforcement learning framework for news recommendation. In Proceedings of the 2018 world wide web conference. https://doi.org/10.1145/3178876.3185994

[30] Chen, X., et al. (2019). Top-K off-policy correction for a REINFORCE recommender system. In Proceedings of the twelfth ACM international conference on web search and data mining. https://doi.org/10.48550/arXiv.1812.02353

[31] Afsar, M. M., Crump, T., & Far, B. (2021). Reinforcement learning based recommender systems: A survey. ACM Computing Surveys (CSUR), 54(2), 1-38. https://doi.org/10.1145/3543846

[32] Levine, S., et al. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. https://doi.org/10.48550/arXiv.2005.01643

[33] Xin, J., et al. (2020). Self-supervised reinforcement learning for recommender systems. In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. https://doi.org/10.48550/arXiv.2006.05779

[34] Chen, X., Yao, L., McAuley, J., Zhou, G., & Wang, X. (2023). Deep reinforcement learning in recommender systems: A survey and new perspectives. Knowledge-Based Systems, 264, 110335. https://doi.org/10.1016/j.knosys.2023.110335

[35] Ma, J., et al. (2019). Disentangled graph collaborative filtering. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. https://doi.org/10.1145/3397271.3401137