EnMatch: Matchmaking for Better Player Engagement via Neural Combinatorial Optimization

Anonymous submission

Abstract

Matchmaking is a core task in e-sports and online games, as it contributes to player engagement and further influences the game's lifecycle. Previous methods focus on creating fair games at all times. They divide players into different tiers based on skill levels and only select players from the same tier for each game. Though this strategy can ensure fair matchmaking, it is not always good for player engagement. In this paper, we propose a novel **En**gagement-oriented **Match**making (**EnMatch**) framework to ensure fair games and simultaneously enhance player engagement. Two main issues need to be addressed. First, it is unclear how to measure the impact of different team compositions and confrontations on player engagement during the game considering the variety of player characteristics. Second, such a detailed consideration on every single player during matchmaking will result in an NP-hard combinatorial optimization problem with non-linear objectives. In light of these challenges, we turn to real-world data analysis to reveal engagement-related factors. The resulting insights guide the development of engagement modeling, enabling the estimation of quantified engagement before a match is completed. To handle the combinatorial optimization problem, we formulate the problem into a reinforcement learning framework, in which a neural combinatorial optimization problem is built and solved. The performance of EnMatch is finally demonstrated through the comparison with other state-of-the-art methods based on several real-world datasets and online deployments on two games.

Introduction

Matchmaking is an essential part of e-sports and online games. It pairs players into different combat teams and helps maintain an enjoyable playing experience for all participants. Previous research focuses on creating balanced games, where closely skilled players are matched to create competitive gameplay, assuming that balanced teams are the most desired matchmaking outcome for players. They hereby design an effective and efficient strategy— first divide players into different tiers and then only select players from the same tier to form opposing teams (Graepel and Herbrich 2006; Herbrich, Minka, and Graepel 2007; Gong et al. 2020a). Players in the same tier are supposed to have similar gaming skills. Hence, through this approach, all the players in one combat have similar gaming skills so that the fairness of games could be well ensured.

However, is game fairness the only critical factor for player engagement? In most matchmaking scenes, the answer is no, which has been demonstrated in EOMM (Chen et al. 2017). Using churn rate as an indicator of player engagement, EOMM analyzes the impact of match win-loss outcomes on player retention in 1-vs-1 scenes and shows that fair games are not sufficient to ensure player engagement. However, it still remains unexplored in scenes that contain multiple players in one team, i.e., k-vs-k mode. Two key challenges exist for an engagement-oriented matchmaking method in k-vs-k scenes:

What are the key factors that influence user engagement? Estimating player engagement in advance is crucial for establishing the optimization objective of matchmaking. Previous works mostly emphasize win-loss outcomes, which may not be sufficient in scenes with larger player counts. As the number of players in a match increases, the impact of factors such as players' roles within a team and the interaction between players becomes gradually increases. When engagement prediction relies on combinations of other players, the feature space of the model can become exponentially large, making model learning based solely on low-resolution user satisfaction labels with limited data difficult (Adams and Zemel 2011; Cao et al. 2007). To accurately predict player engagement, it is essential to figure out what the key factors are and quantify these factors together. This is especially important in our k-vs-k scene.

How to conduct matchmaking with careful consideration on each individual player? Note that in our k-vs-k scene, rather than selecting players from the same tier or forming 1-vs-1 games as in previous works, each player needs to be well estimated under multiple possible team compositions. This results in an NP-hard combinatorial optimization problem with non-linear objectives, which can not be efficiently solved by tier-based heuristics or the graph matching methods (Chen et al. 2017) proposed for 1-vs-1 games. Furthermore, the complexity of player interactions necessitates cross-tier matching, resulting in a significantly larger exponential search space. In addition to considering the quality of the match for a single game, we also need to take into account the quality of the matches that can be formed with the remaining players. Thus, when the focus shifts from fair game to engagement, a new combinatorial optimization solution is required.

It is due to these difficulties that we propose a novel engagement-oriented matchmaking framework, named EnMatch. EnMatch includes a new engagement modeling method that introduces players' granular gameplay performance (including player interactions, players' roles within a team, etc.) as additional supervision information, and a team formation algorithm based on recently developed Neural Combinatorial Optimization (NCO). Overall, the main contributions of this paper are summarized as follows:

- We address the common yet unresolved challenge of optimizing engagement-orient matchmaking in the context of k-vs-k online games, which involves a combinatorial optimization problem with nonlinear objectives.
- We propose EnMatch, which utilizes the latest work in the intersection of neural combinatorial optimization and reinforcement learning. Through this method, the development of matchmaking systems can evolve from fair game to player engagement.
- Exploratory data analysis is conducted to investigate the factors that affect player engagement. Simulation experiments on the industrial game datasets and live experiments are conducted to verify the effectiveness of the proposed method.

Related Work

Skill Modeling and Matchmaking

Many existing matchmaking algorithms are designed to create fair games and are rooted in skill modeling, such as Bradley-Terry (Bradley and Terry 1952a), ELO (Elo 1978), and Glicko (Glickman 1999) for 1-vs-1 scenes, and TrueSkill (Herbrich, Minka, and Graepel 2007; Minka, Cleven, and Zaykov 2018) for k-vs-k scenes, along with their extensions (DeLong et al. 2011; Makhijani and Ugander 2019). The goal of skill modeling is to represent each player's strength as a scalar random variable and then assign players heuristically to form comparable teams with minimal differences in strength. Beyond representing player strength as a real number, recent research (Semenov et al. 2016; Li et al. 2018; Sapienza, Goyal, and Ferrara 2018; Gong et al. 2020a) resort to predicting the game outcome using a neural network, thus taking player profiles into account. Despite the efforts made in skill modeling and game outcome prediction, the question of how to match players into two competing teams is still largely under-explored. While EOMM (Chen et al. 2017) is the first to address this issue, the graph matching is only suitable for 1-vs-1 scenes. GloMatch (Deng et al. 2021) utilizes self-attention to model team effects and employs reinforcement learning (RL) to generate matchmaking results player by player, but fails to beat heuristic methods in performance. Our paper incorporates the recent advancements in neural combinatorial optimization to propose EnMatch. It successfully extends the engagement optimization problem from 1-vs-1 to k-vs-k scenes in industrial practices.

Neural Combinatorial Optimization

Recently, there has been a surge of interest in utilizing deep learning, especially deep reinforcement learning, to learn effective solvers for CO (combinatorial optimization) problems from historical instances, which is termed Neural Combinatorial Optimization (NCO). Successful techniques include (1) GNN-based variable space representation (Khalil et al. 2017; Joshi, Laurent, and Bresson 2019; Ma et al. 2019); (2) combining traditional generic combinatorial optimization techniques, such as searching with pruning (Kwon et al. 2020) and LNS (Wu et al. 2021a); (3) integrating learning models into problem-specific designed operators, such as 2-opt (Chen and Tian 2019; Wu et al. 2021b) and Lin-Kernighan-Helsgaun (LKH) (Xin et al. 2021); (4) modeling the problem as a sequential decision-making problem (constructive heuristics that sequentially extend a partial solution) and optimizing it using deep reinforcement learning (Vinyals, Fortunato, and Jaitly 2015a; Kwon et al. 2020; Song et al. 2022). Leveraging the integration of the aforementioned NCO techniques, EnMatch capably addresses the intricate combinatorial optimization challenges that emerge amidst the transition from a fair game model towards an engagement-centric one. To the best of our knowledge, there are no established NCO algorithms specifically designed for resolving CO problems that employ a neural network as an objective function. EnMatch presents a partial advancement in the matchmaking scene.

Preliminaries

Problem Definition

Let $\mathcal{P} = \{p_i | i = 1, 2, ..., N\}$ be the matching pool, each p_i denotes a player who are waiting to start a k-vs-k match. Players are associated with feature vectors $\mathcal{F} = \{x_i \in \mathbb{R}^d | p_i \in \mathcal{P}\}$. A Match M consists of two opposing teams, T_a and T_b. Each team T is an unordered subset selected from the matching pool $\mathcal{P}$, where $|T| = K$ denotes the number of players in the team. Supposing that N players can form L matches (i.e., $N = 2KL$), our goal is to generate a matching sequence $\mathcal{L} = \{T_a^m, T_b^m\}_{m=1}^L$, with the greatest gross player engagement:

$$\mathcal{U}(\mathcal{L}) = \sum_m \mathcal{U}_m(T_a, T_b) \tag{1}$$

where $\mathcal{U}_m(T_a, T_b)$ denotes the general engagement function for team T_a and T_b in a match $m \in \{1, 2, .., L\}$.

Matchmaking as a CO Problem

Although it has not been explicitly stated in previous work, the matchmaking problem based on skill modeling is actually a variant of the well-known CO problem, the Balanced Number Partitioning Problem (Michiels et al. 2003), which involves partitioning N numbers, each with a weight w_i, into L sets of size K as balanced as possible. The difference is that matchmaking only requires a local balance between two teams. Let $A \in \{0, 1\}^{N \times 2L}$ denotes the decision matrix, and the binary decision variable a_{ij} is 1 if player i is assigned to team j, and 0 otherwise, The CO model for this problem can be expressed as:

Figure 1: The framework of tier-based heuristic matchmaking system.

$$\min \quad \sum_{k=0}^{L-1} \left| \sum_{i=1}^{N} w_i \cdot a_{i,2k} - \sum_{i=1}^{N} w_i \cdot a_{i,2k+1} \right| \tag{2a}$$

$$s.t. \quad a_{ij} \in \{0,1\}, \quad \forall 1 \le i \le N, \forall 1 \le j \le 2L \tag{2b}$$

$$\sum_{i} a_{ij} = 1, \quad \forall 1 \le i \le N \tag{2c}$$

$$\sum_{j} a_{ij} = K, \quad \forall 1 \le j \le 2L \tag{2d}$$

Here, w_i could be the ability score of players, and the objective function means we want to minimize the differences between two teams within a match. Constraint (2b) ensures that each player is either included in a team or not). Constraint (2c) ensures that each player is exactly assigned to one team. Finally, constraints (2d) ensure that each team consists of exactly K players.

Clearly, when the matchmaking objective shifts from fair game to engagement, the optimization problem involves a non-linear objective function (i.e., an engagement model represented by a neural network in this paper), which exceeds the capabilities of integer programming and modern operations research tools such as *Gurobi*[1] and *OR-Tools*[2].

Tier-based Heuristic Matchmaking

The most widely used matchmaking strategy, which we called tier-based heuristic matchmaking, mainly consists of three stages: split, select, and enumeration, as shown in Figure 1. In the split phase, players in the matching pool are divided into different tiers based on their historical abilities. To create fair games, players from different tiers are not allowed to team up. Within the same tier, assuming a k-vs-k scene, $2k$ players will be selected based on handcrafted rules in the select phase. During the enumeration phase, various team candidates are generated from these $2k$ players. For example, in a 3-vs-3 scene, 6 players can form $\binom{6}{3} = 20$ different matching team combinations. Based on expert heuristics or game outcome prediction, the fairest team-ups are selected and the chosen players will be removed from the matching pool. The tier-based heuristics is a good heuristic approach for solving combinatorial optimization problems with fair game objectives by introducing additional tier constraints and neighborhood search.

[1]https://www.gurobi.com/

[2]https://developers.google.com/optimization

Exploratory Data Analysis

To investigate the factors that influence player engagement in k-vs-k scenes, we conduct exploratory data analysis on an anonymized dataset from a popular 3-vs-3 game. As the game does not have a sophisticated matchmaking system, we can collect a diverse range of match data where players from different tiers and with different attributes are matched together. This ensures that both "fair teams" and "diversity teams" exist. Fair teams are composed of players with the same tier, while diversity teams are composed of one high-tier player and two low-tier players. A total of 10,000 matches are collected for each type of matchmaking team. We analyze player interactions, including chat, upvotes from teammates, and downvotes from teammates, to assess collaboration engagement. Additionally, we investigate long-term engagement by analyzing the churn rate of different types of players, following the analysis made in EOMM (Chen et al. 2017). For each game, we divide players into "main players" and "support players". Game win-loss results are also included in our analysis.

Collaboration Engagement. We present player interaction results in Table 1. It shows that diversity teams have a 12.3% decrease in downvotes when they win the game. Meanwhile, regardless of game win-loss results, diversity teams have a higher chat frequency (by 19.2% and 7.1%, respectively) and a higher upvotes rate (by 15.8% and 10.3%, respectively). These statistics provide a strong indication that players of diverse teams are more likely to establish collaboration relationships, thereby improving in-game engagement. On the other hand, the results also show that ensuring fair games alone is not sufficient to maintain a good user experience, and cross-tier matchmaking is necessary to optimize player engagement.

Long-term Engagement. After every three games, we inspect whether the player will quit the game. We calculate averaged churn rate and present results in Table 2. Firstly, for the players winning three consecutive games, i.e., the players in "WWW" rows, the results show that the main players have a lower churn rate when compared with support players (4.9% and 5.9%, respectively). Conversely, for the "LLL" games where players lose three consecutive games, the main players have an obviously larger churn rate than the support players (8.4% and 6.2%, respectively). These reflect that, besides game results, the player role in the game is also an important factor for long-term engagement. Moreover, we also find an interesting phenomenon that main players with three consecutive winning games ("WWW" + "MMM") do not have significantly lower churn rates than all other scenes: the

Table 1: Player engagement-related behavior statistics for different kinds of teams under win/loss situations.

Team Type	#Teams	#Chats	#Upvotes	#Downvotes
Fair_win	10,000	31,824	6,185	970
Diversity_win	10,000	37,934	7,162	851
Diver.%−Fair%	/	19.2%	15.8%	-12.3%
Fair_lose	10,000	56,828	3,510	1,423
Diversity_lose	10,000	60,863	3,872	1,467
Diver.%−Fair%	/	7.1%	10.3%	3.1%

Table 2: The impact of player states on their engagement. Data is from a popular PvP game. Average churn risks vary drastically upon players' recent three match outcomes ((W)in or (L)ose) and player role ((M)ain or (S)upport). "2A+B" refers to all possible sequences of three matches that consist of two A and one B.

ID	Last 3 Outcomes	Last 3 Roles	Churn Rate
1	2W+L | W+2L	MMM	$4.0\% - 5.1\%$
2	2W+L | W+2L	2M+S | M+2S	$4.2\% - 5.2\%$
3	2W+L | W+2L	SSS	$4.9\% - 5.7\%$
4	WWW	MMM	4.9%
5	WWW	2M+S | M+2S	$3.9\% - 5.3\%$
6	WWW	SSS	5.9%
7	LLL	MMM	8.4%
8	LLL	2M+S | M+2S	$6.9\% - 7.7\%$
9	LLL	SSS	6.2%

lowest churn rates in ID 1, 2, and 5 are similar or even better than the churn rate of ID 4. This suggests that diverse win-loss experiences and diverse game roles can lead to higher game engagement of players. Therefore, maximizing player engagement through matchmaking is not a trivial task.

Method

In this section, we introduce our EnMatch framework, which aims to match players across different tiers and attributes to make diverse game experiences and maximize player engagement in k-vs-k scenes.

Overview

The overall framework is presented in Figure 2(a) and there are five major components:

- **Matching Pool** refers to the set of all players who have not been selected. In each step of matching, the matching pool removes the selected 2K players.
- **Encoder** extracts representations for players in the matching pool, considering the potential interactions between players with diverse characteristics.
- **Masked Decoder** generates 2K players autoregressively based on the inputted player representations from the encoder. The generated 2K players can be directly divided into two teams, which are odd-index and even-index teams.
- **Heuristic Operator** is a specially designed CO operator aimed at further enhancing the matching results obtained through decoder output.
- **Engagement Model** provides engagement prediction for each selected player with the players' features and team-up information as input.

As a typical reinforcement learning loop, in EnMatch, the engagement model (i.e., the environment) provides a prediction of user engagement (i.e., the reward signal) and updates the matching pool (i.e., the state) which is subsequently used to generate 2K players (i.e., the action). The decoder's matchmaking results and those obtained by the heuristic operator are jointly optimized to encourage the decoder to output good results from the beginning and be more heuristic operator friendly.

MDP formulation

In one episode of solving a user request (i.e., the matching pool), the discrete-time MDP can be described as follows:

- **State** $s_t \in \mathcal{S}$ represents the matching pool $\mathcal{P}_t$ at time step t. The matching pool is dynamic and each matchmaking process removes 2K matched players.
- **Action** $a_t \in \mathcal{A}_{2k} = \{p_i \,|p_i \in \mathcal{P}_t, |\, p_i \,| = 2K\}$ of the agent is to generate 2K players from all valid players.
- **Transition** P is deterministic here. Next state s_{t+1} is obtained by removing the selected players a_t from the current matching pool s_t, i.e., $s_{t+1} = (\mathcal{P}_t \setminus a_t)$.
- **Reward** r_t is directly related to the estimated engagement in the environment. Specifically, the reward is defined as the sum of individual engagement of the two competing teams T_a and T_b of match M: $r_t = \sum_{p_i \in \mathcal{M}_t} e\left(\boldsymbol{s}_i, \boldsymbol{s}_{M_t}\right)$, where $e(\cdot,\cdot)$ is the engagement model, $\boldsymbol{s}_i$ represents the features of player p_i, while $\boldsymbol{s}_M$ represents the features of all players in a k-vs-k match. Given the step limit T of the interaction between the agent and environment, the return (i.e., cumulative rewards) from step t of the episode is $R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$, with the discount factor $\gamma \in [0, 1]$.
- **Policy** π represents a conditional probability distribution over all possible team compositions given a state.

Engagement Model

The engagement model of EnMatch, depicted in Figure 2(c), predicts each player's engagement by taking into account their granular in-game performance. To aid in representation learning, an auxiliary task is integrated into the initial stages of model training, following the principles of multi-task learning (Caruana 1997) and curriculum learning (Pentina, Sharmanska, and Lampert 2015), where multiple related tasks are learned simultaneously to enhance the model's generalization (Ruder 2017). Please refer to the Appendix for a comprehensive description of the network architecture.

Policy Learning

In this subsection, we introduce the policy network used to select an action given the state.

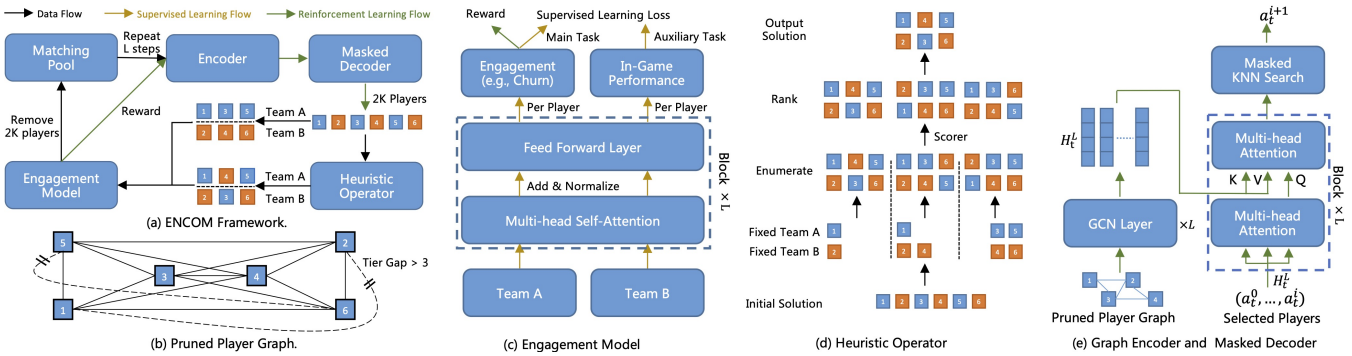

Figure 2: Overall framework (a) and instantiations of each component (b-e). The black lines represent data flow, the yellow lines represent flow for supervised learning, and the green lines represent flow for reinforcement learning.

Pruned Player Graph. The matching pool is described by a pruned graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \cdots, v_n\}$ denotes players $\mathcal{P}$ with features $\mathbf{V} \in \mathbb{R}^{n \times d_v}$ and edges $\mathcal{E}$ indicate whether two players can be matched together. Expert knowledge can be included here to prune this graph. For example, we may not want a very large tier gap within a game and specify a maximum acceptable tier gap as 3. As shown in Figure 2(b), in this scene, edges among players with tier gaps larger than 3 will be removed. These prior rules can to some extent ensure game fairness and directly prevent extremely poor gaming experiences, and are taken into consideration again in the masked decoder.

Graph Encoder. The Graph Neural Network (GNN) is widely employed in the NCO field for variable space representation due to its potential to process instances of any size by sharing parameters across all variables (Khalil et al. 2017; Joshi, Laurent, and Bresson 2019; Ma et al. 2019). We parametrize the encoder part of policy $\pi_\theta\left(a_t \mid s_t\right)$ by a graph convolutional network (GCN), a GNN variant broadly used in various tasks (Kipf and Welling 2016; Duvenaud et al. 2015; Yao, Mao, and Luo 2019), with the graph convolution layer expressed as below:

$$h_i^{(0)} = W_0 x_i + b_0 \tag{3}$$

$$h_i^{(l+1)} = \sigma\left(\mathrm{LN}\left(\sum_{j=1}^{n} \frac{1}{\sqrt{d_i d_j}} A_{ij} h_j^{(l)} W^{(l)} + b^{(l)}\right)\right) \tag{4}$$

Where the W_0 and b_0 are linear projection parameters for the input feature vector x_i, $\mathrm{LN}(\cdot)$ represents the layer normalization operation, $h_i^{(l+1)}$ is the output feature vector of the i-th node in the $(l+1)$-th layer, $\sigma(\cdot)$ is the Tanh activation function, A_{ij} is the adjacency matrix, $h_j^{(l)}$ is the feature vector of the j-th node in the l-th layer, $W^{(l)}$ is the weight matrix, $b^{(l)}$ is the bias vector, d_i and d_j are the degrees of the i-th and j-th nodes, respectively.

Masked Decoder. Recently, RNNs (Gong et al. 2019), Transformer Decoder (Vaswani et al. 2017a), and GPT Decoder (Radford et al. 2019) have been widely used to map the encoder embeddings to a correlated output sequence in an autoregressive manner, so does in our proposed framework. As shown in Figure 2(e), we employ Transformer Decoder to decode the action $a_t = \left\{a_t^0, \ldots, a_t^{2K-1}\right\}$, where a_t^i refers to a specific player. The structure of our masked decoder is similar to the pointer network (Vinyals, Fortunato, and Jaitly 2015b; Bello et al. 2016) and well simplified for faster online predicting. Instead of working on large discrete action spaces like the pointer network, we address it with a continuous action space and combine policy gradients with a KNN (K-Nearest Neighbor) search, i.e., finding the K closest vectors from a set of vectors to a query vector h. The cosine similarity metric is used to measure the similarity between the vectors. In order to take into account the constraints between players, at each decoding step, we apply a masked softmax on the cosine similarity scores o_i to output the action probabilities corresponding to unselected player p_i, which can be done by simple masking:

$$\mathrm{Mask}(x) = \begin{cases} x_i, & \text{if } p_i \text{ is a valid player,} \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

$$\mathrm{MaskedSoftmax}(x) = \mathrm{softmax}(\mathrm{Mask}(x)) \tag{6}$$

$$\hat{h}_i = \begin{cases} \mathrm{cell}(H^L, \hat{h}_0), & \text{if } i = 0, \\ \mathrm{cell}(H^L, \hat{h}_{i-1}, \ldots, \hat{h}_0), & \text{if } i > 0. \end{cases} \tag{7}$$

$$o_i = \mathrm{TopK}(\mathrm{cosine}(\hat{h}_i, H^L)) \tag{8}$$

$$a^i \sim \mathrm{Categorical}(\mathrm{MaskedSoftmax}(o_i)) \tag{9}$$

$$a = (a^0, a^1, \ldots, a^{2K-1}) \tag{10}$$

where H^L is the player embeddings of the last GCN layer, the cell is the Transformer Decoder which consists of two MHA layers, $\hat{h}_i$ is the hidden state of the decoder at step i, cosine is the cosine similarity operator which is followed by TopK operation to truncate the top K optimal results, and a^i is the selected player at step i obtained by weighted sampling or an argmax operation. The Mask function is used to set the mask for invalid players to 0, which ensures that the MaskedSoftmax only considers valid players. The action $a = (a^0, a^1, \ldots, a^{2K-1})$ is then enhanced by the heuristic

operator, and the engagement model provides the reward r.

Heuristic Operator. One of the main reasons for NCO's success is integrating learning models into problem-specific heuristic operators. As shown in Figure 2(d), inspired by the tier-based heuristic method, we design a heuristic operator for matchmaking based on Large Neighborhood Search (LNS) (Wu et al. 2021a), which iteratively destroys and repairs an initial solution. Unlike reinforcement learning, which optimizes player allocation across multiple matches, the heuristic operator focuses on enhancing the search capability of EnMatch within a single match. The operator takes the initial assignment as input and iteratively destroys and repairs the solution while searching neighborhood solutions with a destroy probability p. Expert rules or the engagement model then sorts these candidate solutions to obtain the optimal one. The algorithm is elaborately described in Algorithm 1 located in the Appendix.

Policy Learning. Proximal Policy Optimization (Schulman et al. 2017) is an algorithm that integrates an estimator of the policy gradient into a stochastic gradient ascent algorithm for an agent following a stochastic policy π_θ. The commonly used gradient estimator, denoted as $\hat{g}$, takes the form:

$$\hat{g} = \hat{\mathbb{E}}_t \left[\hat{A}_t \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \tag{11}$$

where $\hat{A}_t(s_t, a_t)$ is an estimator of the advantage function at timestep t (Schulman et al. 2017; Wang et al. 2016), and is utilized to reduce the variance of policy gradient algorithms. The expectation $\hat{\mathbb{E}}_t[\cdot]$ represents the empirical average over a finite batch of samples.

Discussion of Time Performance

After the model is trained, it must be deployed online while satisfying certain latency requirements. A typical matchmaking service request involves between 10^2 to 10^3 players and must be responded to in 300 milliseconds or less. Some RL-based methods, such as GloMatch, define the action as selecting one player, which results in high latency. A commonly used time-saving approach is to divide the player pool into several groups based on player tier, each having n_m players. However, as we have analyzed, engagement-oriented matchmaking requires a larger and more diverse player pool than a fair game one. Compared to methods like GloMatch, EnMatch supports cross-tier matchmaking and two to three times the number of players, and achieves a three to four times speedup. During the online planning phase, the optimal candidate generated by the heuristic operator will be used as the final matchmaking assignment.

Simulation Experiments On Datasets

As existing public datasets lack player identity information necessary to measure individual engagement, we first conduct an artificial experiment to address the balanced matchmaking problem described in the preliminaries, which is postponed to the appendix due to its length and complexity. EnMatch is effective in handling large combination spaces, indicating its potential to support the modification of the matchmaking objective from fair game to user engagement. We further conduct several simulation experiments on two industrial datasets to evaluate our framework.

Table 3: Summary Statistics of the Datasets

Dataset	Matches	Players	Features	Match Mode
SPG	851,648	33,873	22	3-vs-3
RPGPVP	400,985	185,232	36	15-vs-15

Experimental Settings

The statistics of two industrial datasets are given in Table 3, which are SPG dataset, collected from a popular online basketball game, and RPGPVP dataset (Deng et al. 2021), collected from a well-known MMORPG game[3]. Following the setting of EOMM (Chen et al. 2017) and GloMatch (Deng et al. 2021), the engagement is measured by the probability of players remaining active within 30 minutes, and we assume a matching pool of 120 candidate players, i.e., building 20 and 4 matches for these two datasets, respectively. The Tier-based Heuristics and EnMatch limit the number of scoring candidate sets to 100 and 1,000 for these two datasets, respectively. For all the experiments, we are using a single GeForce GTX 2080 Ti GPU, 8CPU, and 64GB memory. More experiment details, including data process, data split, feature design, hyperparameters, and simulation process are listed in the Appendix.

Baselines

We consider the following baselines, including match outcome predictors (EOMM, OptMatch (Gong et al. 2020a), and MBNet (Deng et al. 2021)), heuristic matchmaking approaches (Tier-based Heuristics), general NCO methods (PointerNetwork (Bello et al. 2016)), and deep learning based matchmaking approaches (GloMatch (Deng et al. 2021)). A detailed description of baselines is listed in the Appendix.

Results and Analysis

The results of the engagement modeling on the two datasets are shown in Table 4. We can find that EnMatchNet without auxiliary achieves similar results compared to MBNet (i.e., GloMatch) and OptMatchNet, while EnMatchNet achieves the best performance in both accuracy and AUC metrics with an improvement of 3.23%/2.57% and 2.86%/2.92%, respectively, over the second-best method. It suggests that the use of high-resolution intermediate labels as auxiliary tasks can enhance the accuracy and efficiency of the engagement prediction, which is consistent with our data analysis results. Using the learned engagement model as a reward function, we report the matchmaking performance on the average reward per episode (120 players) on the simulation settings, which is summarized and shown in Table 5. EnMatch achieves higher rewards in both datasets, with improvements

[3]Both games are real-world production games. Their names are kept anonymous during the review stage.

Table 4: Classification metrics of the player engagement prediction model for two datasets, where * denotes the second best result.

	Fever Basketball		*RPGPVP*	
	ACC	AUC	ACC	AUC
EOMM	0.679	0.606	0.725	0.681
OptMatchNet	0.703	0.657	0.755	0.744
MBNet	0.711*	0.663	0.769	0.752*
EnMatchNet w/o aux.	0.708	0.664*	0.778*	0.751
EnMatchNet	**0.734**	**0.683**	**0.798**	**0.774**
Improvement	3.23%	2.86%	2.57%	2.92%

Table 5: Performance comparison for different matchmaking methods in the two simulation environments in terms of the cumulative reward per episode (a larger value means a higher player retention).

Reward	*SPG*	*RPGPVP*
Random	49.3 (±4.4)	40.1 (±7.6)
Tier-based Heuristics	64.7 (±0.3)	51.9 (±0.6)
PointNetwork	62.3 (±1.2)	47.6 (±2.7)
GloMatch	61.9 (±1.6)	52.7* (±3.1)
OptMatch	65.1* (±0.4)	51.2 (±0.8)
EnMatch	**66.5 (±0.9)**	**58.9 (±1.5)**
Improvement	2.15%	11.76%

of 2.15% and 11.76%, respectively. The performance gain is particularly significant in 15-vs-15 scenes where the search space is larger. The results prove the applicability of EnMatch to act as an engagement optimizer. RL-based GloMatch and search-based methods, including OptMatch and tier-based heuristics, show different performance strengths. Search-based methods are better for local search within a single game, making it effective in SPG, while GloMatch is better for making decisions for multiple games simultaneously, which determines its effectiveness in the RPGPVP dataset. EnMatch combines the strengths of both RL and search heuristics by incorporating NCO for optimal performance in both scenes.

Online Deployment

We applied our proposed engagement-oriented matchmaking system to industrial matchmaking services in *SPG* and *RPGPVP*, with the system primarily running three fair game algorithms: tier-based heuristic method, OptMatch-fair, and GloMatch-fair. In a two-week online A/B testing, we compare EnMatch's performance with OptMatch-fair and GloMatch-fair, as well as the three baselines adjusted to optimize for player engagement.

Results

To account for the impact of different algorithms on players' daily retention, we use the total number of matches played under the same A/B testing traffic as an engagement metric, instead of the average number of matches per player. This metric is equivalent to the sum of player retention events, where a retention event is defined as a player participating in the next match within 30 minutes. As expected, engagement-oriented algorithms, such as OptMatch and GloMatch, resulted in more match sessions for players compared to their fair counterparts, OptMatch-fair and GloMatch-fair. Tier-based heuristics and OptMatch perform well in the SPG scene for 3-vs-3 matches with a small combination space. However, in the large search space RPGPVP scene (15-vs-15), GloMatch outperforms heuristics and OptMatch, but still lagged behind EnMatch, which is consistent with our simulation experiments. EnMatch achieves larger numbers of matches in both games, indicating higher player retention. Although our optimization goal is the retention of the next match, the metric which has considered daily player retention still achieved significant growth. It shows that engagement matchmaking is also meaningful in the long term.

Table 6: Online performance comparison on two games.

Num. of matches	*SPG*	*RPGPVP*
OptMatch-fair	197,862	50,637
GloMatch-fair	195,993	50,795
Tier-based Heuristics	216,720	53,671
OptMatch	219,284*	54,924
GloMatch	214,973	55,133*
EnMatch	**222,880**	**59,097**
Improvement	1.64%	7.19%

System Performance

Our policy network's single inference time is 20-30 milliseconds with a GPU device and 200-300 milliseconds with a CPU device. Based on online service logs, matchmaking requests typically involve less than 400 players, and the EnMatch based system takes less than 200 milliseconds on average to respond to one matchmaking request using a GPU. Our method can handle more players at once than state-of-the-art methods like OptMatch and GloMatch, without requiring grouping, due to its powerful capacity for solving combinatorial optimization.

Conclusion

We propose EnMatch, a novel framework for engagement-optimized matchmaking in more general k-vs-k scenes from the view of neural combinatorial optimization. EnMatch consists of three components: an engagement predictive model, a heuristic operator, and an encoder-decoder network, which are independent and can be tuned or replaced for specific applications. To evaluate the framework, we conduct simulations based on real data from online games and live experiments on two games. Moreover, not limited to matchmaking in games, the formulation and optimization techniques of EnMatch are general and can be further applied to combinatorial optimization (CO) problems, particularly assignment problems, with non-linear objectives, which are left as future work.

References

Adams, R. P.; and Zemel, R. S. 2011. Ranking via sinkhorn propagation. *arXiv preprint arXiv:1106.1925*.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bello, I.; Pham, H.; Le, Q. V.; Norouzi, M.; and Bengio, S. 2016. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*.

Bradley, R. A.; and Terry, M. E. 1952a. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4): 324–345.

Bradley, R. A.; and Terry, M. E. 1952b. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4): 324–345.

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; and Li, H. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, 129–136.

Caruana, R. 1997. Multitask learning. *Machine learning*, 28: 41–75.

Chen, X.; and Tian, Y. 2019. Learning to perform local rewriting for combinatorial optimization. *Advances in Neural Information Processing Systems*, 32.

Chen, Z.; Xue, S.; Kolen, J.; Aghdaie, N.; Zaman, K. A.; Sun, Y.; and Seif El-Nasr, M. 2017. Eomm: An engagement optimized matchmaking framework. In *Proceedings of the 26th International Conference on World Wide Web*, 1143–1150. International World Wide Web Conferences Steering Committee.

Delalleau, O.; Contal, E.; Thibodeau-Laufer, E.; Ferrari, R. C.; Bengio, Y.; and Zhang, F. 2012. Beyond skill rating: Advanced matchmaking in ghost recon online. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(3): 167–177.

DeLong, C.; Pathak, N.; Erickson, K.; Perrino, E.; Shim, K.; and Srivastava, J. 2011. TeamSkill: modeling team chemistry in online multi-player games. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 519–531. Springer.

Deng, Q.; Li, H.; Wang, K.; Hu, Z.; Wu, R.; Gong, L.; Tao, J.; Fan, C.; and Cui, P. 2021. Globally optimized matchmaking in online games. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2753–2763.

Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28.

Elo, A. E. 1978. *The rating of chessplayers, past and present*. Arco Pub.

Glickman, M. E. 1999. Parameter estimation in large dynamic paired comparison experiments. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 48(3): 377–394.

Gong, L.; Feng, X.; Ye, D.; Li, H.; Wu, R.; Tao, J.; Fan, C.; and Cui, P. 2020a. OptMatch: Optimized Matchmaking via Modeling the High-Order Interactions on the Arena. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2300–2310.

Gong, L.; Feng, X.; Ye, D.; Li, H.; Wu, R.; Tao, J.; Fan, C.; and Cui, P. 2020b. Optmatch: Optimized matchmaking via modeling the high-order interactions on the arena. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2300–2310.

Gong, Y.; Zhu, Y.; Duan, L.; Liu, Q.; Guan, Z.; Sun, F.; Ou, W.; and Zhu, K. Q. 2019. Exact-k recommendation via maximal clique optimization. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 617–626.

Graepel, T.; and Herbrich, R. 2006. Ranking and matchmaking. *Game Developer Magazine*, 25: 34.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Herbrich, R.; Minka, T.; and Graepel, T. 2006. TrueSkill™: a Bayesian skill rating system. *Advances in neural information processing systems*, 19.

Herbrich, R.; Minka, T.; and Graepel, T. 2007. TrueSkill™: a Bayesian skill rating system. In *Advances in neural information processing systems*, 569–576.

Joshi, C. K.; Laurent, T.; and Bresson, X. 2019. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*.

Khalil, E.; Dai, H.; Zhang, Y.; Dilkina, B.; and Song, L. 2017. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kwon, Y.-D.; Choo, J.; Kim, B.; Yoon, I.; Gwon, Y.; and Min, S. 2020. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 21188–21198.

Li, Y.; Cheng, M.; Fujii, K.; Hsieh, F.; and Hsieh, C.-J. 2018. Learning from group comparisons: exploiting higher order interactions. In *Advances in Neural Information Processing Systems*, 4981–4990.

Ma, Q.; Ge, S.; He, D.; Thaker, D.; and Drori, I. 2019. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. *arXiv preprint arXiv:1911.04936*.

Makhijani, R.; and Ugander, J. 2019. Parametric Models for Intransitivity in Pairwise Rankings. In *The World Wide Web Conference*, 3056–3062. ACM.

Michiels, W.; Korst, J.; Aarts, E.; and Van Leeuwen, J. 2003. Performance ratios for the differencing method applied to the balanced number partitioning problem. In *STACS 2003: 20th Annual Symposium on Theoretical Aspects of Computer Science Berlin, Germany, February 27–March 1, 2003 Proceedings 20*, 583–595. Springer.

Minka, T.; Cleven, R.; and Zaykov, Y. 2018. Trueskill 2: An improved bayesian skill rating system. *Tech. Rep.*

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, 8026–8037.

Pentina, A.; Sharmanska, V.; and Lampert, C. H. 2015. Curriculum learning of multiple tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5492–5500.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Ruder, S. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Sapienza, A.; Goyal, P.; and Ferrara, E. 2018. Deep Neural Networks for Optimal Team Composition. *arXiv preprint arXiv:1805.03285*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Semenov, A.; Romov, P.; Korolev, S.; Yashkov, D.; and Neklyudov, K. 2016. Performance of machine learning algorithms in predicting game outcome from drafts in Dota 2. In *International Conference on Analysis of Images, Social Networks and Texts*, 26–37. Springer.

Song, W.; Cao, Z.; Zhang, J.; Xu, C.; and Lim, A. 2022. Learning variable ordering heuristics for solving constraint satisfaction problems. *Engineering Applications of Artificial Intelligence*, 109: 104603.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017a. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017b. Attention is all you need. *Advances in neural information processing systems*, 30.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015a. Pointer networks. *Advances in neural information processing systems*, 28.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015b. Pointer networks. In *Advances in neural information processing systems*, 2692–2700.

Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, 1995–2003. PMLR.

Wu, Y.; Song, W.; Cao, Z.; and Zhang, J. 2021a. Learning large neighborhood search policy for integer programming. *Advances in Neural Information Processing Systems*, 34: 30075–30087.

Wu, Y.; Song, W.; Cao, Z.; Zhang, J.; and Lim, A. 2021b. Learning improvement heuristics for solving routing problems. *IEEE transactions on neural networks and learning systems*, 33(9): 5057–5069.

Xin, L.; Song, W.; Cao, Z.; and Zhang, J. 2021. NeuroLKH: Combining deep learning model with Lin-Kernighan-Helsgaun heuristic for solving the traveling salesman problem. *Advances in Neural Information Processing Systems*, 34: 7472–7483.

Yao, L.; Mao, C.; and Luo, Y. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 7370–7377.

Network Architecture of Engagement Model

Matchmaking methods focused on balance typically rely on players' skill ratings (Herbrich, Minka, and Graepel 2006; Bradley and Terry 1952b) or match outcome predictions (Gong et al. 2020b; Herbrich, Minka, and Graepel 2006; Delalleau et al. 2012). For instance, OptMatch (Gong et al. 2020a) and GloMatch (Deng et al. 2021) select matches with predicted win rates closest to 50%. For engagement-oriented matchmaking, a straightforward approach is substituting the match outcome prediction model with an engagement prediction model that takes two teams (2K players) as input and predicts each player's engagement. Following the EOMM method, a common engagement measure is retention probability, or the likelihood of a player continuing to play within a subsequent timeframe. In team-oriented game scenarios, previous studies typically use naive pooling operations (Herbrich, Minka, and Graepel 2006; Delalleau et al. 2012) or Multi-Head Attention (MHA) (Vaswani et al. 2017b; Deng et al. 2021) to learn aggregated representations for outcome prediction, capturing intra-team interactions. The proposed engagement model in EnMatch, depicted in Figure 2(c), predicts each player's engagement rather than the match-level outcomes. It employs L self-attention layers, composed of an MHA layer and a feed-forward network (FFN) layer.

$$h_i^{(0)} = W_0 x_i + b_0 \tag{12}$$

$$\text{SelfAttention}(Q) = \text{softmax}(\frac{QQ^T}{\sqrt{d}})Q \tag{13}$$

$$\widehat{H^{(l)}} = \text{Concat}(\text{Head}_1, .., \text{Head}_k)W^O \tag{14}$$

$$\text{where Head}_i = \text{SelfAttention}\left(H^{(l-1)}W_{hi}\right)$$

$$h_i^{(l)} = \text{FFN}(\widehat{h_i^{(l)}}) = \max(0, W_1^{(l)}\widehat{h_i^{(l)}} + b_1^{(l)})W_2^{(l)} + b_2^{(l)} \tag{15}$$

where the W_0 and b_0 are linear projection parameters for the input feature vector x_i, $H^{(l)} = \left\{h_i^{(l)}\right\}_{1 \le i \le N}$ denotes player embeddings generated by layer l. (k) represents the number of heads, $W_{hi} \in \mathbb{R}^{d_h \times d_k}$ is the parameter for each head, and $W^O \in \mathbb{R}^{(kd_k) \times d_h}$ are projection parameters for MHA. $W_1^{(l)}, W_2^{(l)}, b_1^{(l)}$, and $b_2^{(l)}$ are learnable parameters for the FFN. Moreover, each sub-layer incorporates a skip-connection (He et al. 2016) and layer normalization (Ba, Kiros, and Hinton 2016). feature

The hidden representation $h_i^{(L)}$ is then passed through a sigmoid activation function to obtain the predicted retention probability for binary classification (main task), and a linear function to obtain the predicted in-game performance for continuous regression tasks (auxiliary task) with an Adaptive Weighted Mean Squared Error (AWMSE). The auxiliary task is incorporated into the early stages of model training as a curriculum to assist in representation learning. This approach is in line with the idea of multi-task learning (Caruana 1997) and curriculum learning (Pentina, Sharmanska, and Lampert 2015), where multiple related tasks are learned to improve the generalization of the model (Ruder 2017). The ground-truth retention label indicates whether a player intends to continue playing in the next game within a certain time. In-game performance in EnMatch encompasses various metrics such as game outcome, performance ranking, score, number of key behaviors, and others reflecting a player's current match performance. A player's features include both portrait features like their tier and online time, and in-game performance features from their three most recent matches. The loss function is defined as:

$$\mathcal{L}_{main} = -\sum_{i=1}^{N} \left[y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)\right] \tag{16}$$

$$\mathcal{L}_{aux} = \frac{1}{N} \sum_{i=1}^{N} \frac{\sum_j (z_{ij} - \hat{z}_{ij})^2}{\sum_j (z_{ij} - \bar{z}_j)^2} \tag{17}$$

where y_i represents the ground-truth retention label for player i, $\hat{y}_i$ is the predicted retention probability, and z_i, $\hat{z}_i$, $\bar{z}_i$ correspond to the actual, predicted, and average in-game performance, respectively.

Algorithm of Heuristic Operator

Algorithm 1: LNS-based Heuristic Operator Algorithm

Input: Initial solution Team A and B, LNS steps M, search size k, scoring function f, destroy probability p
Output: The optimal Team A and Team B.

```
bestScore ← −∞  bestTeamsA, bestTeamsB ← teamA, teamB
for i = 1 to M do
    tempTeamA, tempTeamB ← bestTeamsA, bestTeamsB  for j = 1 to k do
        fixedPos ← Sample([tempTeamA, tempTeamB], 1-p)
          PartialShuffle([tempTeamA, tempTeamB], fixedPos)
        score ← f(tempTeamA, tempTeamB)  if score ¿ bestScore then
            bestScore ← score  bestTeamsA ← tempTeamA
              bestTeamsB ← tempTeamB
        end
    end
end
return bestTeamsA, bestTeamsB
```

Details of Simulation Experiments

Baselines

We consider the following baselines, including match outcome predictors, heuristic matchmaking approaches, general NCO methods, and deep learning based matchmaking approaches:

- **EOMM.** The logistic regression based match outcome prediction model proposed in EOMM (Chen et al. 2017).
- **OptMatchNet (Gong et al. 2020a).** The attention-based match outcome prediction model proposed in OptMatch (Gong et al. 2020a), which takes players' pre-trained node embedding as the input.
- **MBNet.** The match balance model proposed in GloMatch (Deng et al. 2021), which is basically a Transformer.
- **EnMatchNet.** Our proposed engagement model described in Section . We conduct an ablation study by comparing its performance with and without auxiliary tasks.
- **Random.** A strategy that randomly arranges players.
- **Tier-based Heuristics.** The heuristic strategy is described in Section . without the ranking phase.
- **PointNetwork (Bello et al. 2016).** A state-of-the-art general NCO method, which has been widely used for Traveling Salesman Problem (TSP) and Vehicle Routing Problem (VRP).
- **GloMatch (Deng et al. 2021).** A novel balanced matchmaking framework that generates the matchmaking result player by player using basic reinforcement learning methods.

Table 7: Handcrafted features for players in the SPG dataset.

Feature No.	Feature Description
1	role level of the player account
2	historical skill score
3	historical mvp rate
4	historical game frequency
5	average assistance in this match
6	average times of being blocked in this match
7	average passes in this match
8	average times of being stolen in this match
9	average times of ball touching in this match
10	average blocks in this match in this match
11	score rank in this match
12	rank of mvp rate in this match
13	match result in this match
14	average rebounces in this match
15	average scores in this match
16	statistics of skill uses in this match
17	average steal times in this match
18	average 3-point fieldgoals in this match
19	average 3-point shoots in this match
20	average 2-point fieldgoals in this match
21	average 2-point shoots in this match
22	average wideopen scores in this match

- **OptMatch (Gong et al. 2020a).** A two-stage matchmaking framework that combines a win-loss prediction model and heuristic matchmaking strategies based on single-match optimization.

Feature Design

For the game datasets used in the simulation experiments, we designed several features related to the game characteristics. As an example, Table 7 lists the handcrafted features we built for players in SPG dataset alongside their descriptions.

Data Process

The original dataset contains matches sorted by time and identified by unique match ID. In order to concatenate player sessions, all match records are duplicated and identified by both match ID and unique player ID. We then identify all matches in which each player participated and sort them by time. After obtaining player sessions, we concatenate the in-game features from the 5 most recent matches as features for each match.

Data Split

Matches are sorted by the time for each dataset. Then we take the first 80% matches as the training set and the remaining 20% matches as the test set. This ensures no leak of label information from the test set.

Simulation Procedure

To compare different matchmaking algorithms, we create a realistic RL environment for simulation, following the setting of EOMM. To maintain fairness, we utilize the same population (matching pool) and simulator, based on the same learned user engagement model. During the simulation, we sample each player's state from a collection of real players' states to reflect the actual data distribution. For each matchmaking method, the simulation procedure within each round is as follows:

1. Create a matching pool of P players to simulate real-world requests.
2. Use a matchmaking strategy to obtain the matching results, i.e., L games of k-vs-k.
3. For each game, input the selected 2K players and obtain the engagement estimation as rewards for each player using the trained engagement model.
4. Repeat steps 1, 2, and 3 for 100 times, and calculate the averaged reward.

Here, the simulation experiments assume a matching pool of 120 candidate players for both the SPG and RPGPVP datasets, i.e., building 20 and 4 matches for these two datasets, respectively. For SPG dataset, we set L and K to be 20 and 3, respectively. For dataset B, we set L and K to be 4 and 15, respectively.

Hyper-parameters

We implement the match outcome prediction model and policy network by PyTorch (Paszke et al. 2019) with the Adam optimizer (Kingma and Ba 2014).

- We employ early stopping and dropout techniques to prevent over-fitting when possible.
- The embedding size is fixed to 128 for all models. The hidden sizes of linear layers are set to 512.
- In this work, the decoder of EnMatch is composed of a stack of $L = 2$ identical self-attention layers, and we employ $h = 4$ parallel attention heads.
- We apply grid search for tuning the hyper-parameters of the outcome prediction model: the learning rate is tuned amongst $\{0.0001, 0.0005, 0.001\}$, the coefficient of $L2$ regularization is searched in $\{10^{-5}, 10^{-4}\}$, and the dropout ratio in $\{0.1, 0.2, 0.3\}$. The batch size is 128 for all models.

Additional Artificial Experiments

As existing public datasets lack player identity information necessary to measure individual engagement, we first conduct an artificial experiment to address the balanced matchmaking problem described in Section 3.2. We simplify the team matching problem to a number matching problem: choose 2L groups of data with k players from a pool of N numbers to form L matches, such that the difference in the sum of numbers between each match is minimized. Experimental results showed that even in the task of fair matchmaking, EnMatch performs better than traditional heuristic methods and other baselines.

Baselines

We consider the following baselines, including heuristic approaches, deep learning based matchmaking approaches, general NCO methods, and the integer programming method:

- **Random.** A matchmaking strategy that randomly arranges players.
- **Tier-based Heuristics.** The heuristic strategy is described in Section 3.2.2.
- **PointNetwork (Vinyals, Fortunato, and Jaitly 2015b).** A state-of-the-art general NCO method, which has been widely used for Traveling Salesman Problem (TSP) and Vehicle Routing Problem (VRP).

Table 8: Performance comparison for different methods on synthetic datasets with the objective of balanced matchmaking, where * denotes the the best result except for OR-Tools.

Obj.	(N=6, K=3, L=1)	(N=18, K=3, L=3)	(N=20, K=10, L=1)	(N=60, K=10, L=3)
Random	5879.5	17348.0	~10^7	~10^7
Tier-based Heuristics-100	1599.8	1529.3	17630.6	16749.0
OptMatch-100	1578.5	1531.2	16520.2	51287.9
Tier-based Heuristics-1000	**1279.2***	1131.9	1901.3	1841.9
OptMatch-1000	**1279.2***	1949.0	1769.7	5242.3
PointNetwork	2910.0	3431.2	~10^5	~10^5
GloMatch	2209.4	1398.2	18751.4	10917.9
Ortools-2	1279.2	64.4	194.7	9411.1
Ortools-120	**1279.2**	**23.4**	**19.6**	**230.3**
EnMatch-100	1396.2	1097.1	4091.7	9186.2
EnMatch-1000	**1279.2***	**1023.7***	**861.3***	**1247.5***

- **GloMatch (Deng et al. 2021).** A novel balanced matchmaking framework that generates the matchmaking result player by player using basic reinforcement learning methods.
- **OptMatch (Gong et al. 2020a).** A two-stage matchmaking framework that combines a win-loss prediction model and heuristic matchmaking strategies based on single-match optimization.
- **OR-Tools**. An advanced open-source integer programming solver, which is acted as the upper bound of the optimal solution. We limit the solving time to 2 and 120 seconds, termed Ortools-2 and Ortools-120.

Experimental Results

We conduct four sets of experiments in increasing order of difficulty, with the following parameters: (N=6, K=3, L=1), (N=18, K=3, L=3), (N=20, K=10, L=1), and (N=60, K=10, L=3). The objective function is defined in equation 2. Tier-based Heuristics, OptMatch, and EnMatch, which are search-based methods, are limited to a maximum search count of 10, 10, 100, and 100 in four experimental scenarios, referred to as algo-100. Similarly, they are limited to a maximum search count of 100, 100, 1000, and 1000, referred to as algo-1000. It can be seen that ortools-120 achieves the best results, but it takes 2 minutes to solve, which is unacceptable for online services. Moreover, ortools cannot handle scenarios with nonlinear objective functions. In the $(N = 60, K = 10, L = 3)$ scenario, Ortools-2 also shows a significant performance drop due to the reduced solving time. Compared to problems such as TSP, where the optimal objective value increases with problem size, the difficulty of the Number Matchmaking scenario increases with the increase of K and N, but the optimal objective value decreases sharply. This results in the optimal solution's objective value being much better than the second-best EnMatch. Among all matchmaking baselines, EnMatch achieves the best results, especially in the $K = 10$ scenario. Tier-based Heuristics and OptMatch perform well in the $L = 1$ case, but due to the lack of consideration for multiple game matches, they perform poorly in the $L = 3$ case, as reflected in GloMatch's superiority over Tier-based Heuristics-100 and OptMatch-100 in some scenarios.