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:

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

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

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.