OptMatch: Optimized Matchmaking via Modeling the High-Order Interactions on the Arena

Linxia Gong[1], Xiaochuan Feng[1], Dezhi Ye[1], Hao Li[1], Runze Wu[1],
Jianrong Tao[1], Changjie Fan[1], Peng Cui[2]
[1]Fuxi AI Lab, NetEase Inc., Hangzhou, China
[2]Tsinghua University, China
{gonglinxia,fengxiaochuan,yedezhi,lihao01,wurunze1,hztaojianrong,fanchangjie}@corp.netease.com,
cuip@tsinghua.edu.cn

ABSTRACT

Matchmaking is a core problem for the e-sports and online games, which determines the player satisfaction and further influences the life cycle of the gaming products. Most of matchmaking systems take the form of grouping the queuing players into two opposing teams by following certain rules. The design and implementation of matchmaking systems are usually product-specific and labor-intensive.

This paper proposes a two-stage data-driven matchmaking framework (namely OptMatch), which is applicable to most of gaming products and has the minimal product knowledge required. OptMatch contains an offline learning stage and an online planning stage. The offline learning stage includes (1) relationship mining modules to learn the low-dimensional representations of individuals by capturing the high-order inter-personal interactions, and (2) a neural network to incorporate the team-up effect and predict the match outcomes. The online planning stage optimizes the gross player utilities (i.e., satisfaction) during the matchmaking process, by leveraging the learned representations and predictive model.

Quantitative evaluations on four real-world datasets and an online experiment on Fever Basketball[1] game are conducted to empirically demonstrate the effectiveness of OptMatch.

CCS CONCEPTS

• **Information systems → Data mining**; **Massively multiplayer online games**; • **Computing methodologies → Learning latent representations**.

KEYWORDS

matchmaking; graph embedding; player satisfaction; user modeling

ACM Reference Format:

Linxia Gong, Xiaochuan Feng, Dezhi Ye, Hao Li, Runze Wu, and Jianrong Tao, Changjie Fan, Peng Cui. 2020. OptMatch: Optimized Matchmaking via Modeling the High-Order Interactions on the Arena. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, NY, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403279

[1]https://chao.163.com/index.html

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403279

1 INTRODUCTION

Matchmaking arranges multiple players into player-versus-player (PvP) competitions. With large number of players requesting the gaming service and indicating their availability, the matchmaking system forms teams for queuing players and brings up every two teams into one combat.

Matchmaking is ubiquitous in the sporting events, especially in e-sports and online games. Most matchmaking systems in the market are skill-based[1, 7, 8, 12, 19] or feature-based[3, 4] and require profound industrial knowledge for the product designers.

To facilitate the matchmaking mechanism design, this paper investigates the high-order inter-personal interactions on the arena, and proposes a generalized data-driven matchmaking framework **OptMatch** that gives optimized matchmaking results based on the captured interactions.

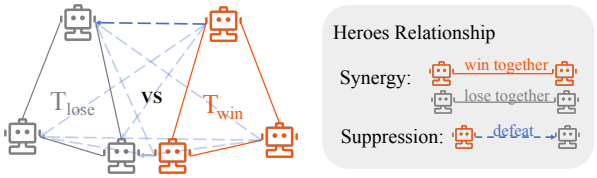


Figure 1: Relationships between individuals in one match.

In terms of high-order interactions on the arena, intuitively, an individual can be complementary, incompatible or even suppressive to others because of their abilities difference, competing styles or social effects. These mutual impacts are revealed by match outcome as shown in Figure 1, and the interpersonal interactions can be then identified throughout the numerous collected matches.

In order to acquire the high-order interactions among individuals, OptMatch concludes two types of relations from the match records to represent the tacit interplay of players, i.e., (1) synergy relation built from the win/lose together relationship, and (2) suppression relation indicating the advantage of one over another. In addition, OptMatch adopts an attention-based model that explicitly incorporates the intra-team and inter-team interactions to predict the match results and players' utilities. Finally, during the online

service, OptMatch uses a heuristic planning method combined with the interactions and patterns learned from past matches, to optimize the future matchmaking results.

In summary, the main contributions of this paper include:

- a generalized data-driven matchmaking system that optimizes the match qualities and player utilities;
- two interpersonal relations for representing and understanding tacit coordination interactions among players;
- corresponding graph embedding methods to incorporate the high-order interactions, together with the comparison of various graph embedding algorithms;
- a model to encode team-up effect and predict the match outcome.
- a github page[2] that contains the datasets and models mentioned in the paper.

The empirical results show that OptMatch gains an improvement of 6% ∼ 17% in the offline prediction tasks for the match outcomes on four real-world datasets, and an improvement of around 40% in the match quality during an online experiment.

2 RELATED WORK

2.1 Team Formation and Matchmaking

Extensive prior research has studied the matchup strategies to form comparable teams for each competition. The mainstream matchmaking strategies are skill-based and concentrate on building accurate skill rating systems. In this case, one player is assigned with one score that describes her absolute strength. This ability score is usually derived from the probabilistic algorithms like Bradley-Terry, ELO, Glicko, TrueSkill [1, 7, 8, 12, 19], and their extensions [5, 13, 17, 18, 24, 31]. The ability of the team is then modeled as the summation of the team members' scores.

Considering that using a single scalar is an oversimplification, subsequent strategies [4] extend the skill-based models to incorporate additional player information by inputting players' feature vectors and some [3] propose new model structures to take into account the domain-specific concerns.

Meanwhile, some strategies [16, 25, 26] look into the cooperative effect in the team composition. They aim to study the interplay between a player's performance improvement (resp., decline) throughout matches in the presence of beneficial (resp., disadvantageous) teammates by computing the pairwise strength weights between individuals.

The existing work mostly emphasizes the tracking of individual skills, either by inferring the synthetic ability score, or by constructing feature vectors based on expert knowledge. Some other research about the interpersonal effects limits to the pairwise influence and analyzes only players or heroes in each experiment. Despite the efforts of depicting individual abilities and mutual interplay, the exploitation of the high-order interactions among the heroes and players is inadequately studied.

2.2 Graph Embedding

Representation learning on graphs has been proposed to transform instances in topological space into fixed-size vectors in Euclidean space, in which geometric distance reflects their structural similarity. The widely applied graph embedding approaches include FM [22], DeepFM [11], DeepWalk [6, 21], Node2vec [10], LINE [27], SDNE [29], Struc2Vec [23]. Factorization based methods [11, 22] are computationally expensive and cannot be implemented in parallel. The other methods, learn the representation vectors of vertices based on the comparison of the neighborhoods of them.

In the use cases of this paper, the graphs are usually densely-connected, but the homophily hypothesis (i.e., nodes that are highly interconnected and belong to similar network clusters or communities should be embedded closely together, adopted by [6, 10, 21, 29]) on the graph is not always valid. In addition, the local structure of the vertices [23] is not the focus of this paper. Therefore, we adjust the graph embedding method of LINE [27] to learn the latent representation of vertices in this paper.

3 PROBLEM DEFINITION

3.1 Online Games and E-sports Preliminaries

- `Hero`: the graphical representation of the user in the virtual worlds. It usually takes a three-dimensional form of character in games.
- `PVP`: Player(s)-versus-player(s), is a type of multiplayer interactive conflict within a game between two or more live participants. It is used to describe any game, or aspect of a game, where players compete against each other.

3.2 Matchmaking

We define the matchmaking into two stages: the offline learning stage to learn the patterns from past matches; an online planning stage to optimize the players' satisfaction for future matches.

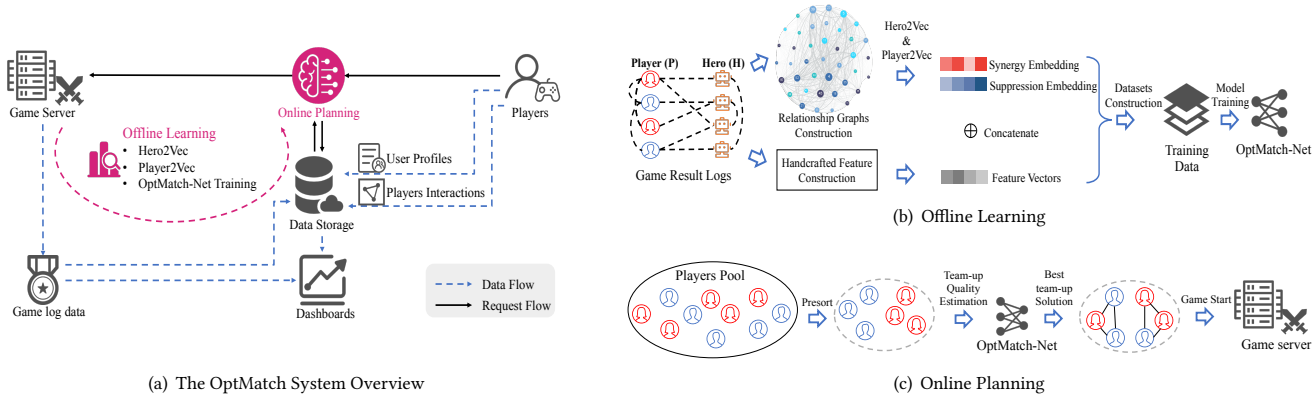
3.2.1 ***Offline Learning.*** Given a dataset with M observed matches, a population of heroes $\mathcal{H} = \{h_1, ..., h_H\}$ and players $\mathcal{P} = \{p_1, ..., p_N\}$ are involved. In each game, two disjoint teams competed with each other, which are denoted as T_{win}, T_{lose} by the outcome. For simplicity, we assume each team has k members, where $k = |T_{win}| = |T_{lose}|$. Offline learning aims to grasp the relations among the individuals (heroes/players) from the T_{win}, T_{lose} records, and identify the competition patterns throughout the matches.

3.2.2 ***Online Planning.*** In practice, matchmaking is applied to a pool of players $\mathcal{P}_t = \{p_1, \ldots, p_n\}$, who are waiting to start the k-vs-k matches at the moment t. The goal of matchmaking is to find the optimal assignments for players to maximize the total utility (i.e., the gross players' satisfaction). With $\mathcal{U}_m$ denoting the utility gained by the match m, the system makes the match-making decision to maximizing $\sum_{m=0}^{M} \mathcal{U}_m$ for the matches it creates. Mostly, the utility $\mathcal{U}_m$ of one single game is regarded as the accumulation of its players' utilities

$$\mathcal{U}_m = \sum_{i=0}^{n_m} \mathcal{U}_{m,i} \tag{1}$$

The *player utility* refers to a quantitative measure of a player's satisfaction gained from a match, for example, the reverse of churn risk for this player after a certain match. This requires the matchmaking system to pick right teammates and good opponents for as many players as possible.

[2] https://fuxiailab.github.io/OptMatch/



(a) The OptMatch System Overview

(b) Offline Learning

(c) Online Planning

Figure 2: The OptMatch System framework. (a) shows the structure of the two-stage iterative matchmaking system, with arrows to indicate its data flows and request flows. (b) shows the iterative offline learning phase, which contains the feature engineering (handcrafted features building and auto feature mining from relationships), and the model updating. (c) shows the online planning phase that matches up the players to start games.

3.3 Game Outcome

In each game, a team's outcome o_t can be observed in two forms: the binary indicator of win/loss ($o_t \in \{+1, -1\}$), or the value of score ($o_t \in \mathbb{R}$).

With richer information about the game, the o_t of score record is usually preferred when it's available. Conventional methods for utilizing the score information are to predict the score difference values. This leads to two limitations: firstly, it constrains the data aggregation among the datasets with scores and the datasets with only binary results; and secondly, the fitting values are greatly influenced by the magnitude of the score measure designs and the competitive intensity of each game. For example, a game with a score of 0vs10 should be different from a game with a score of 90vs100 intuitively but they are treated the same if only score difference is concerned.

For the aforementioned considerations, we convert the outcome into its relative form: (the $\hat{}$ symbol denotes the ground truth values, for distinguishing them from the model prediction values in the latter sections)

$$\hat{o}_{rela} = \pm \frac{|o_{win} - o_{lose}|}{|o_{win}| + |o_{lose}|} \tag{2}$$

where the sign of the relative outcome is determined by which team it refers to (+1 for the winning team and −1 for the losing one). This relative outcome unifies the two forms of observed game results, rescales the team score values to the range $[-1, +1]$, and keeps the binary results in $\{+1, -1\}$ that can be interpreted as a special scenario of the former form.

3.4 Graph Embedding in Matchmaking

Graph Embedding. Similar to [27], in matchmaking problem, a graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices, each representing an individual (hero or player) and $\mathcal{E}$ is the set of edges between the vertices, each representing a relationship between two individuals. Each edge $e \in \mathcal{E}$ is an ordered pair $e = (a, b)$ and is associated with a weight that indicates the strength of the relation.

Given a graph, graph embedding aims to represent each vertex $v \in \mathcal{V}$ into a low-dimensional space $\mathbb{R}^d$, i.e., learning a function $f_G : \mathcal{V} \to \mathbb{R}^d$, where $d \ll |\mathcal{V}|$. In the space $\mathbb{R}^d$, the first-order proximity and the second-order proximity between the vertices are preserved.

First-order Proximity. The first-order proximity in a graph is the local pairwise proximity between two vertices. For each pair of vertices linked by an edge (a, b), the weight on that edge $w_{a,b}$ indicates the first-order proximity between a and b [27].

In the matchmaking and competition context, the first-order proximity implies the compatibility of two individuals, i.e., $w_{a,b} > 0$ implies the beneficial interplay between the two heroes/players, while $w_{a,b} < 0$ implies the disadvantageous interplay.

Second-order Proximity. The second-order proximity between a pair of vertices (a, b) in a graph is the similarity between their neighborhood graph structures. Let $p_a = (w_{a,1}, \ldots, w_{a,|\mathcal{V}|})$ denote the first-order proximity of a with all the other vertices, then the second-order proximity between a and b is determined by the similarity between p_a and p_b (if no vertex is linked from/to both a and b, the second-order proximity is 0) [27].

Likewise, in the matchmaking context, the second-order proximity implies that individuals who share similar compatible/incompatible teammates, or share similar suppressive opponents, tend to be similar to each other.

4 MATCHMAKING SYSTEM

PvP competitions usually involve two opposing teams in many sporting events and online games. In the context of e-sports and online games, players compete by controlling selected heroes, each of which is designed with different strengths and weaknesses. While in traditional sports, in one sense, each player can be considered as one hero to fit in the OptMatch system.

4.1 System Overview

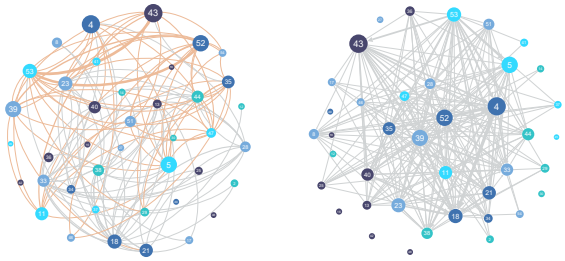
The OptMatch system consists of two phases: offline learning and online planning. The offline learning phase contains:

- *Hero2Vec* learns the low-dimensional representations of heroes that incorporates their relationship in the competitions. The embedding vectors that represent different relations can be trained separately and parallelly.
- *Player2Vec* constructs the representation vectors of players based on the hero representations learned in Hero2Vec, as well as the features built from players' records.
- *OptMatch-Net* model takes the players' representations and team-up information to predict the match results. The model incorporates the inter-player interactions and learns the competition patterns.

The online planning service, afterwards, leverages the representation vectors of players and OptMatch-Net model to maximize the (predicted) gross utilities for the queuing players.

4.2 Hero2Vec

4.2.1 ***Graph Construction.*** To grasp the interplay among heroes in Figure 1, we first need to extract and construct the relationship graphs that reflect their mutual impact. In the intense and fast-paced competitions, tacit coordination plays a larger role than verbal communication [15]. Also, the complementary data of chatting or friendship is usually hard to acquire. Considering these reasons, we focus only on the relationships that can be refined from the match records. After the examination of the match characteristics, we identify two types of relations out of the two opposing k-individual teams T_{win}, T_{lose}, as shown in Figure 3:



(a) Synergy Graph (b) Suppression Graph

Figure 3: The relationship graphs of heroes in Fever Basketball dataset, where the node size is proportional to its occurrence frequency and link width to the relation strength. (a) Synergy graph represents the compatibility of heroes, wherein the orange links are associated with positive weights and the gray with negative weights; (b) Suppression graph represents the advantage/disadvantage that a hero has over another, with the arrows indicating the suppressing relation.

a) Synergy Graph Construction. Synergy describes the compatibility between heroes, which is defined as the beneficial (resp., disadvantageous) relation between two heroes when they both showed up in the winning (resp., losing) side of the match. To construct the synergy graph, we go through the winning and losing teams respectively and build an edge between each pair of heroes. The edge weight is defined as:

$$w_{a,b}^{syn} = f_{a,b}^{win} - f_{a,b}^{lose} \tag{3}$$

where $f_{a,b}^{win}$ refers to the times when the heroes (h_a, h_b) co-occurred in the winning teams throughout the collected matches, while $f_{a,b}^{lose}$ refers to the co-occurrence of the heroes (h_a, h_b) in the losing teams. Therefore, the synergy graph is a signed undirected graph.

b) Suppression Graph Construction. Different from the synergy relation above, suppression is a directional relation, which indicates that one hero h_a defeated another hero h_b. The weight for an order pair (a, b) is defined as:

$$w_{(a,b)}^{suppr} = f_{(a \text{ defeat } b)} \tag{4}$$

where $f_{(a \text{ defeat } b)}$ is the times when the hero a defeated the hero b, and vice versa for the link weight $w_{(b,a)}^{suppr}$.

4.2.2 ***Graph Embedding.*** After building the graphs that represent the structural relationships among heroes, we start to train the function to project the heroes into low-dimensional vectors while maintaining the necessary information in the meanwhile.

a) Graph Embedding on Synergy Graph (a signed undirected graph): As the Synergy Graph has both positive and negative connections, the graph embedding method in [27] and the signed proximity term proposed in [14] is adopted to learn the compact low-dimensional representation of the heroes.

The first-order proximity in the Synergy Graph. We formulate the joint probability between hero a and hero b as:

$$p_1(a, b) = \sigma(\vec{e}_a^T \cdot \vec{e}_b) = \frac{1}{1 + \exp(-\text{sign}(a, b) \cdot \vec{e}_a^T \cdot \vec{e}_b)} \tag{5}$$

where $\vec{e}_a \in \mathbb{R}^d$ is the low-dimensional embedding vector of vertex a. Meanwhile, its empirical probability can be defined as

$$\hat{p}_1(a, b) = \frac{|w_{a,b}|}{\sum_{(a,b)\in\mathcal{E}} |w_{a,b}|} \tag{6}$$

To preserve the first-order proximity, we need to minimize the distance between the two distributions p_1 and $\hat{p}_1$. By choosing the KL-divergence as the distance measure and omitting the constants, we get the objective function:

$$O_1 = -\sum_{(a,b)\in\mathcal{E}} |w_{a,b}| \cdot \log p_1(a, b) \tag{7}$$

The second-order proximity in the Synergy Graph. The second-order proximity in [27] assumes that vertices sharing many neighborhoods are similar to each other. That is, each vertex is treated as a specific "context" and vertices with similar distributions over the "contexts" are assumed to be similar. We use the vector $\vec{e}_a$ to represent vertex a, and $\vec{e}'_a$ to represent the "context" of vertex a. For each edge (a, b), the probability of "context" $\vec{e}'_b$ generated by vertex $\vec{e}_a$ is formulated as:

$$p_2(b|a) = \frac{\exp(\text{sign}(a, b) \cdot \vec{e}_b'^T \cdot \vec{e}_a)}{\sum_{k=1}^{|\mathcal{V}|} \exp(\text{sign}(a, k) \cdot \vec{e}_k'^T \cdot \vec{e}_a)} \tag{8}$$

By combining the empirical distribution $\hat{p}_2(b|a) = \frac{|w_{a,b}|}{\sum_{k\in N(a)}|w_{a,k}|}$ with $N(i)$ denoting the out-neighbors of vertex a, and the KL-divergence function, we get the objective function:

$$O_2 = -\sum_{(a,b)\in\mathcal{E}} |w_{a,b}| \cdot \log p_2(b|a) \tag{9}$$

Then the first-order proximity and second-order proximity in the Synergy Graph can be preserved during the graph embedding, by jointly training the objective function (7) and (9).

The signed proximity term $sign(a,b)\cdot\vec{e}_a^T\cdot\vec{e}_b$ and $sign(a,b)\cdot\vec{e}_b'^T\cdot\vec{e}_a$ are the signed inner product of two vectors, introduced by [14]. For positive connected nodes, likelihood value increases as the inner product increases. Negative and noise pairs, on the other hand, have higher likelihood value when the inner product similarity is lower.

b) Graph Embedding on Suppression Graph (an unsigned directed graph): Note that the first-order proximity is only applicable for undirected graphs, not for directed graphs. This fits the intuition that a pair of heroes (h_a, h_b) who has pairwise suppressive relation doesn't show the local proximity. Therefore, only the *second-order proximity* is preserved in the Suppression Graph.

The graph embedding on the Suppression Graph defines the probability distributions and minimizes the objective function as follows:

$$p_2(b|a) = \frac{\exp(\vec{e}_b'^T\cdot\vec{e}_a)}{\sum_{k=1}^{|\mathcal{V}|}\exp(\vec{e}_k'^T\cdot\vec{e}_a)} \qquad \hat{p}_2(b|a) = \frac{w_{a,b}}{\sum_{k\in N(a)} w_{a,k}} \tag{10}$$

$$O_2 = -\sum_{(a,b)\in\mathcal{E}} w_{a,b}\cdot\log p_2(b|a) \tag{11}$$

By learning $\{\vec{e}_i\}_{i=1,\ldots,|\mathcal{V}|}$ and $\{\vec{e}_i'\}_{i=1,\ldots,|\mathcal{V}|}$ that minimize the objective (11), we get the d-dimensional vector $\vec{e}_i$ to represent each vertex i, $i\in\mathcal{V}$.

Note: Apart from the graph construction and embedding method above, many other ways to count the link weights and learn the vertex embedding are available in the matchmaking scenario. The comparison of various network construction and embedding methods is provided in Appendix B.

4.3 Player2Vec

Player2Vec constructs the representation vectors for players, which includes the handcrafted features that identified and calculated from the players' profiles, as well as the relational representation vectors that are learned from relationship mining.

4.3.1 Player Feature Vector. As the traditional machine learning methods do, we calculate several indexes that imply the ability of players and make an effect on the competition outcome. The indexes are usually designed by the experts in the domain and are greatly case-specific. An example of feature design that we implemented on the Fever Basketball dataset is given in the Appendix B.1.

4.3.2 Player Representations in the Hero Spaces. In the other way, we project the players into the subspaces that encode the hero relationships respectively. For the use cases of sports and the game modes where players compete with fixed heroes, we can simply employ the vectors of heroes to represent the players.

For the game modes with selectable heroes for players, the projection can be performed by building and learning a heterogeneous graph that contains the several relationships among players, among heroes and inbetween. Whereas, for the concern of deployment efficiency, we adopt a straightforward solution, which projects the players through average-pooling over their hero-selection to encode the players' preference, and the hero-mastery to encode their proficiency. For each player, we have the representation vectors as

$$\vec{u} = \text{Concat}(\vec{u}^{pick}, \vec{u}^{win})$$
$$\vec{u}^{pick} = \sum_{h=1}^{H} p_h^{pick}\cdot\vec{e}_h, \quad \vec{u}^{win} = \sum_{h=1}^{H} p_h^{win}\cdot\vec{e}_h \tag{12}$$

in both synergy subspace and suppression subspace (denoted as $\vec{u}^{syn}$ and $\vec{u}^{suppr}$ for latter reference). In (12), p_h^{pick} is the pick rate of hero h, and p_h^{win} is the pick rate of hero h when we investigate only the winning matches of this player.

4.4 OptMatch-Net

OptMatch-Net model is to predict the utility of the match m, defined as (1). Work like [4] computed utility $\mathcal{U}_m$ by predicting the utility of every player $\mathcal{U}_{m,i}$ in the match one by one. In our work, we simplify the computation by adopting the assumption that fair games lead to better player experience. Thus we reformulate the utility of a match $\mathcal{U}_m$ in equation (1) into the drawn probability of the match:

$$\mathcal{U}_m = P_m(drawn) = 1 - |o_{rela}| \tag{13}$$

With (13), we design the OptMatch-Net as in Figure 4. Note that OptMatch-Net is able to compute the match utility $\mathcal{U}_m$ in (1) without the simplification, by slightly changing the structure as [4].

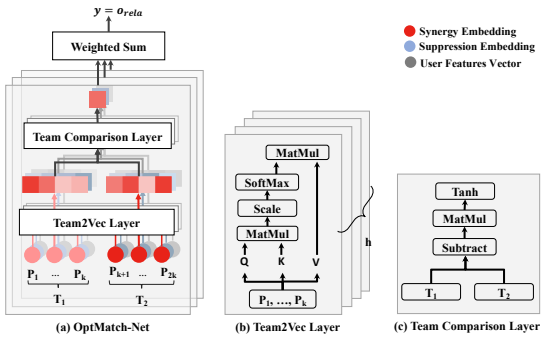


Figure 4: The OptMatch-Net model to evaluate the matchmaking quality.

4.4.1 Input Layer. With the players' representation vectors learned in 4.3, we start to learn the collective representations of each team and compare the two teams by their team representation vectors.

As the input of OptMatch-Net model, two teams of players $T_1 = \{p_1, p_2, \ldots, p_k\}, T_2 = \{p_{k+1}, p_{k+2}, \ldots, p_{2k}\}$ are given, and each player has several embedding vectors (that is, relational embedding vectors $\vec{u}^{syn}$, $\vec{u}^{suppr}$ and player feature vector $\vec{u}^{ftr}$).

To facilitate the model training efficiency and interpretability, the players' embedding vectors sets(i.e., U^{syn}, U^{suppr} and U^{ftr}) are fed into the model separately to get a prediction value respectively. The final output is the weighted sum of those prediction values.

4.4.2 Team2Vec Layer. Precedent literature [4, 12] treated the team as the simple summation or concatenation of its members. Instead, we utilize a multi-head self-attentive mechanism proposed by [28] to incorporate the *intra-team interactions* to construct the representations of a team.

Particularly, given a team of players with their embedding vectors U where $U \in \{U^{syn}, U^{suppr}, U^{ftr}\}$, we compute the output after the multi-head attention function as:

$$\begin{aligned} x &= [\vec{u}_1, \cdots, \vec{u}_k] \\ \text{Attention}(Q,K,V) &= \text{softmax}(\frac{QK^T}{\sqrt{d_K}})V, \quad K \in \mathbb{R}^{d_K} \\ \text{MultiHead}(Q,K,V) &= \text{Concat}(head_1, \cdots, head_h)W^o \\ \text{where } head_i &= \text{Attention}(xW_i^Q, xW_i^K, xW_i^V) \end{aligned} \tag{14}$$

And finally with the multi-head attention output, we feed it into a neural network with a Relu activation function[9], to generate the d_{team}-dimensional representation vector of the team:

$$\vec{T} = \text{Relu}(W \cdot \text{MultiHead}(Q,K,V) + b), \quad \vec{T} \in \mathbb{R}^{d_{team}} \tag{15}$$

where W is the weight matrix, b is the bias vector, and $\vec{T}$ refers to the team representation.

In the competition circumstances that we studied, there are no different restrictions between the two teams. Hence, the parameters of the Team2Vec layer are shared between T_1 and T_2. In other cases, however, when the two teams have different goals such as attacking and defensing, the layer parameters shall not be shared.

4.4.3 Team Comparison Layer. This layer aims to compare the abilities of the two teams and predict the competition result. Classical win prediction models, such as ELO and Bradley-Terry[1], estimate the probability of the pairwise comparison result based on their pre-evaluated capabilities:

$$\text{P}(i \text{ beats } j) = \frac{c_i}{c_j + c_i} \tag{16}$$

where $c_i, c_j \in \mathbb{R}^+$ are the capability values of i and j, and (i, j) can be two competing individuals or groups. According to Blade-Chest[2], the formula (16) can be transformed as:

$$\begin{aligned} \text{P}(i \text{ beats } j) &= \frac{\exp(\alpha_i)}{\exp(\alpha_i) + \exp(\alpha_j)} = \frac{1}{1 + \exp(-(\alpha_i - \alpha_j))} \\ &= \sigma(\Delta(i,j)) \end{aligned} \tag{17}$$

where α_i represents the strength of player i, $\sigma(\cdot)$ is the sigmoid function, and $\Delta(i, j)$ is the matchup function that measures the edge given to player i when matched up against player j. In the Bradley-Terry algorithm, it is modeled as $\Delta(i, j) = \alpha_i - \alpha_j$, the difference of capability values between two players. While the Blade-Chest algorithm extends (16) to incorporate additional player information, by modeling each player's strength as a weighted sum $\alpha_i = W^T \vec{u}_i$, the matchup function turns to be

$$\Delta(i,j) = W^T(\vec{u}_i - \vec{u}_j) \tag{18}$$

On the other hand, in our task, the competition results take the form of relative score difference between the two teams, which from another perspective can be seen as the empirical probability distribution of the *advantage* of i over j:

$$\hat{\text{A}}(i \text{ beats } j) = \hat{o}_{rela} = \frac{s_i - s_j}{s_i + s_j} \quad \hat{\text{A}} \in \mathbb{R}, -1 \le \hat{\text{A}} \le 1 \tag{19}$$

where s_i and s_j are the scores of two teams.

To fit the empirical probability distribution form in (19), we adjust the win probability definition in (16) and (17) into the advantage probability as

$$\begin{aligned} \text{A}(i \text{ beats } j) &= \frac{c_i - c_j}{c_j + c_i} = \frac{\exp(\alpha_i) - \exp(\alpha_j)}{\exp(\alpha_i) + \exp(\alpha_j)} \\ &= \frac{\exp(\alpha_i - \alpha_j) - 1}{\exp(\alpha_i - \alpha_j) + 1} = \tanh(\Delta(i,j)) \\ \text{and } \tanh(x) &= \frac{\exp(2x) - 1}{\exp(2x) + 1} \end{aligned} \tag{20}$$

where the matchup function $\Delta(i, j)$ should have the properties:

- when $\Delta(i, j) \to +\infty$, $\text{A}(i \text{ beats } j) \to 1$
- when $\Delta(i, j) \to -\infty$, $\text{A}(i \text{ beats } j) \to -1$
- when $\Delta(i, j) = 0$, $\text{A}(i \text{ beats } j) = 0$

As the matchup function $\Delta(i, j)$ in (18) satisfies the aforementioned requirements, we simply build the team comparison layer with a neural network that gives the prediction:

$$o_{rela} = \tanh(W^T(\vec{T}_1 - \vec{T}_2) + b) \tag{21}$$

where the $\vec{T}_1$ and $\vec{T}_2$ are the representation of two competing teams, yielded by the Team2Vec layer.

4.4.4 Output Layer. On each embedding vectors set U, where $U \in \{U^{syn}, U^{suppr}, U^{ftr}\}$, the corresponding team representations $(\vec{T}_1, \vec{T}_2)$ and comparison prediction o_{rela} can be computed.

Each prediction value can be viewed as a team strength difference measure from one perspective. To learn a comprehensive team comparison result, we leverage the weighted summation of all the predictions. The output of OptMatch-Net is then

$$y = w^{syn} * o_{rela}^{syn} + w^{suppr} * o_{rela}^{suppr} + w^{ftr} * o_{rela}^{ftr} \tag{22}$$

where the weights w^{syn}, w^{suppr} and w^{ftr} are learned during the training as well.

4.4.5 Objective Function. With the different forms of match results available in the dataset, OptMatch-Net fulfills different tasks and is trained with different objective functions.

a) For relative score difference task, the match result $\hat{y}_i \in (-1, 1)$ is a continuous value, therefore OptMatch-Net takes the mean square error as the loss function, i.e., $\mathcal{L}(o_{rela}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$.

b) For win prediction task, the match result $\hat{y}_i \in \{-1, 1\}$ is a binary result, therefore OptMatch-Net takes the hinge loss as the loss function, i.e., $\mathcal{L}(win) = \sum_i^N \max(0, 1 - y_i \cdot \hat{y}_i)$.

4.5 Online Planning

The goal of online planning is to ensure high-quality matchups in every game, or at least as many games as possible.

Additionally, as one matchmaking service request usually involves 10^2 to 10^3 players and is required to be responded in $\le$ 100ms,

the exhaustive search is infeasible. In this case, we apply a heuristic method to split players queue into several units each of which has n_m players, by the players' ability ranking tiers. On a unit of n_m players, we acquire several match-up candidates based on the team formation methods in [30], which are proved to be *ex post Pareto efficient*. Then we predict the gross utility of those n_m players for each match-up candidate with OptMatch-Net, and keep the maximum value as the utility $\mathcal{U}_m$ together with the corresponding match-up candidate as the matchmaking assignment.

5 DATASETS

5.1 Dataset Statistics

Public and industrial datasets are used to measure the performance of OptMatch. The statistics of datasets are given in Table 1.

Table 1: Statistics of the datasets

Dataset	Matches	Heroes	Players
Dota2 (5v5)	50,000	113	10,815
LOL (5v5)	187,588	139	43,706
NBA	3,342	/	949
Fever Basketball (3v3)	851,648	40	33,873

Public Datasets:

- *Dota2*, *LOL(League of Legends)* are datasets of the famous 5v5 Multiplayer Online Battle Arena (MOBA) games, in which players control heroes of different abilities to fight. *Dota2* contains 50,000 matches, 113 heroes and 10,815 players. *LOL(League of Legends)* contains 187,588 matches, 139 heroes and 43,706 players.
- *NBA* contains 3,342 matches[3] of 2018-2019, with the top points, top rebounds and top assists players of each game (949 basketball players in total).

Industrial Dataset:

- *Fever basketball* is an online basketball game[4] released by Netease. We work on the competitive 3v3 ranking mode and collect a large-scale dataset of recent matches. The dataset contains 851,648 matches, 40 heroes and 33,873 players.

5.2 Dataset 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. The feature vectors and network embeddings of players are learned from the records in the training set. This ensures no leak of result information from the test set.

6 EXPERIMENTS

Extensive experiments are conducted to compare our algorithm with state-of-art algorithms and demonstrate the effectiveness of the proposed framework.

Details of the experiments setting are given in appendix B.

[3]https://rapidapi.com/api-sports/api/api-nba
[4]https://chao.163.com/index.html

6.1 Performance of OptMatch-Net

6.1.1 ***Baselines.*** OptMatch-Net is compared with five methods:

- LR [20]: A linear model with logistic loss.
- XGBoost [26]: A scalable tree-based model for feature learning and game outcome prediction.
- BalanceNet [4]: A multi-layer neural network method widely deployed in online games.
- Blade-Chest [2]: A model that incorporates the intransitivity in competition and predicts the match results.
- HOI [16]: A factorization based model which takes pair-interaction of players into consideration.

6.1.2 ***Model Variants.*** As the team comparison layer adopts a simple form in OptMatch-Net, we investigate only the contribution of the team2vec layer, which employs the attention mechanism.

- OptMatch-Net-no-attention: A variant that uses average-pooling to replace the self-attention mechanism.

6.1.3 Evaluation Metrics. For datasets(*Dota2*, *LOL* and *NBA*) with only binary result marks (win/loss) available, Accuracy(Acc) is used to measure the models.

For dataset(*Fever Basketball*) with both team scores and win/loss flags available, we perform two prediction tasks: win prediction and relative score difference prediction. Accuracy (Acc) is used as the evaluation metric for the former task, and Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) for the latter.

Additionally, we test the model trained with the score difference on the win prediction task, which verifies the robustness and portability of our method. We define a Transfer Accuracy ($\text{Acc}_{\text{trans}}$) as the evaluation metric:

$$\text{Acc}_{\text{trans}} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(y_i = H(\hat{y_i})), \text{ with } H(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases} \quad (23)$$

6.1.4 Prediction Performance Comparison (Table 2). We can observe that OptMatch-Net significantly outperforms all the compared baselines across all the datasets and prediction tasks. Besides, the comparison between OptMatch-Net variants verifies the effect of Team2Vec layer for capturing the intra-team interactions. It manifests that our model adopts a more principled way to leverage statistic features and interaction relations. Moreover, the improvement on the metric $\text{Acc}_{\text{trans}}$ indicates the reusability and mobility of the models.

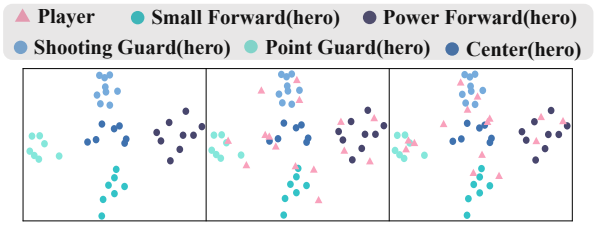
6.2 Effectiveness of Graph Embedding

Figure 5 visualizes the representations of heroes and players.

The leftmost subfigures in 5(a) and 5(b) are the representation vectors of heroes in the synergy and suppression space. We can see that heroes of the same category congregate with each other while heroes of different categories stay distant, which fits our intuitions that those heroes of the same category are designed to be similar, and different categories are designed to suppress one another to some degree. If, in the application cases, a hero doesn't behave as its expected roles in the games, its representation vectors will appear in the spots away from its category, with a big Euclidean distance value. This feature helps the diagnosis of the game design.

Table 2: Performance comparison for different methods. (The best results of baselines are indicated with *).

Algorithm	Fever Basketball				NBA	Dota2	LOL
	MAE	RMSE	Acc	Acc_{trans}	Acc	Acc	Acc
LR	0.178	0.227	0.613	0.642*	0.602	0.596	0.599
XGBoost	0.184	0.233	0.639*	0.631	0.635*	0.602	0.613
BalanceNet	0.175*	0.224*	0.628	0.639	0.619	0.614	0.624
Blade-Chest	0.177	0.231	0.629	0.634	0.589	0.623	0.636
HOI	0.208	0.243	0.639*	0.640	0.597	0.628*	0.639*
OptMatch-Net-no-attention	0.172	0.222	0.648	0.653	0.637	0.645	0.655
OptMatch-Net	**0.169**	**0.218**	**0.660**	**0.670**	**0.646**	**0.652**	**0.661**



(a) Visualization of graph embedding results on Synergy graph

(b) Visualization of graph embedding results on Suppression graph

Figure 5: Visualization of the graph embedding results on *Fever Basketball* dataset.

The two subfigures on the right of Figure 5(a) and 5(b), encode the players into the vector space of heroes, showing that players have diverse preference of picking heroes and proficiency in controlling.

Table 3: Model performance with only feature vectors

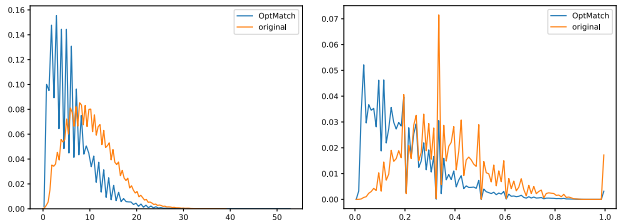
Algorithm	Fever Basketball	
	MAE	RMSE
LR	0.209 (↓ 17.4%)	0.254 (↓ 11.8%)
XGBoost	0.220 (↓ 19.5%)	0.253 (↓ 13.4%)
BalanceNet	0.185 (↓ 5.7%)	0.236 (↓ 5.3%)
Blade-Chest	0.204 (↓ 15.2%)	0.251 (↓ 8.6%)
OptMatch-Net	0.209 (↓ 23.6%)	0.251 (↓ 15.1%)

To verify the effectiveness of the relational representations, we test the model performance with only players' feature vectors. The results in Table 3 show a performance decrease of 5.6% to 23.6%.

6.3 Online Matchmaking System Experiments

In order to assess the performance of the OptMatch framework, we conduct online A/B tests on Fever Basketball, where we distribute the matchmaking requests randomly into OptMatch service and its original matchmaking service (designed with sophisticated skill system and matchmaking rules).

Match Quality. The perceived quality of the matchmaking system is in terms of the score difference in the matches. To evaluate the ability of the two matchmaking systems, we compare the *absolute score difference* and *relative score difference (defined in equation (2))* of the games assigned by the two systems respectively in Figure 6. The matches assigned by OptMatch have generally smaller score difference, which refer to fairer games and better match quality. With the original matchmaking system yielding matches with an average absolute score difference of 10.108 and average relative score difference of 0.370, OptMatch yields these indexes of 6.123 and 0.203, giving an improvement of around 40%.



(a) Distribution of *Absolute* Score Difference (b) Distribution of *Relative* Score Difference

Figure 6: The distribution of score difference throughout online matches. (Smaller value in x-axis refers to better match quality.)

System Performance. According to the online service records, OptMatch system takes ~100ms in average with a setting of 4 CPU and 16GB memory to respond to one matchmaking query.

7 CONCLUSION

This paper introduces a generalized data-driven matchmaking framework (OptMatch) that learns the high-order interpersonal interactions and optimize the player satisfaction.

The advantages of OptMatch over most prior work are that (1) it is applicable to most of use cases, including sports, e-sports and online games with minimal requirements for product-specific knowledge, as the interactions OptMatch incorporates exist in most competitions; (2) it has minimal requirements of data to work, as only the basic team-up information and match outcome are necessary; (3) it is sensitive and efficient to trace the data changes by iteratively updating the players' representation vectors as well as the prediction model; (4) it gives the matchmaking results that lead to optimized gross player utilities.

REFERENCES

[1] Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika* 39, 3/4 (1952), 324–345.

[2] Shuo Chen and Thorsten Joachims. 2016. Modeling intransitivity in matchup and comparison data. In *Proceedings of the ninth acm international conference on web search and data mining*. ACM, 227–236.

[3] Shuo Chen and Thorsten Joachims. 2016. Predicting matchups and preferences in context. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 775–784.

[4] Olivier Delalleau, Emile Contal, Eric Thibodeau-Laufer, Raul Chandias Ferrari, Yoshua Bengio, and Frank Zhang. 2012. Beyond skill rating: Advanced matchmaking in ghost recon online. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 3 (2012), 167–177.

[5] Colin DeLong, Nishith Pathak, Kendrick Erickson, Eric Perrino, Kyong Shim, and Jaideep Srivastava. 2011. TeamSkill: modeling team chemistry in online multi-player games. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 519–531.

[6] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 135–144.

[7] Arpad E Elo. 1978. *The rating of chessplayers, past and present*. Arco Pub.

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

[9] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 315–323.

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[12] Ralf Herbrich, Tom Minka, and Thore Graepel. 2007. TrueSkill™: a Bayesian skill rating system. In *Advances in neural information processing systems*. 569–576.

[13] Tzu-Kuo Huang, Chih-Jen Lin, and Ruby C Weng. 2008. Ranking individuals by group comparisons. *Journal of Machine Learning Research* 9, Oct (2008), 2187–2216.

[14] Junghwan Kim, Haekyu Park, Ji-Eun Lee, and U Kang. 2018. Side: Representation learning in signed directed networks. In *Proceedings of the 2018 World Wide Web Conference*. 509–518.

[15] Young Ji Kim, David Engel, Anita Williams Woolley, Jeffrey Yu-Ting Lin, Naomi McArthur, and Thomas W Malone. 2017. What makes a strong team?: Using collective intelligence to predict team performance in League of Legends. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. ACM, 2316–2329.

[16] Yao Li, Minhao Cheng, Kevin Fujii, Fushing Hsieh, and Cho-Jui Hsieh. 2018. Learning from group comparisons: exploiting higher order interactions. In *Advances in Neural Information Processing Systems*. 4981–4990.

[17] Rahul Makhijani and Johan Ugander. 2019. Parametric Models for Intransitivity in Pairwise Rankings. In *The World Wide Web Conference*. ACM, 3056–3062.

[18] Joshua E Menke, C Shane Reese, and Tony R Martinez. 2007. Hierarchical models for estimating individual ratings from group competitions. In *American Statistical Association*. Citeseer.

[19] Tom Minka, Ryan Cleven, and Yordan Zaykov. 2018. Trueskill 2: An improved bayesian skill rating system. *Tech. Rep.* (2018).

[20] Andrew Y Ng and Michael I Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*. 841–848.

[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[22] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[23] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 385–394.

[24] Ilya O Ryzhov, Awais Tariq, and Warren B Powell. 2011. May the best man win: simulation optimization for match-making in e-sports. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*. IEEE, 4234–4245.

[25] Anna Sapienza, Palash Goyal, and Emilio Ferrara. 2018. Deep Neural Networks for Optimal Team Composition. *arXiv preprint arXiv:1805.03285* (2018).

[26] Aleksandr Semenov, Peter Romov, Sergey Korolev, Daniil Yashkov, and Kirill Neklyudov. 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*. Springer, 26–37.

[27] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 1067–1077.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

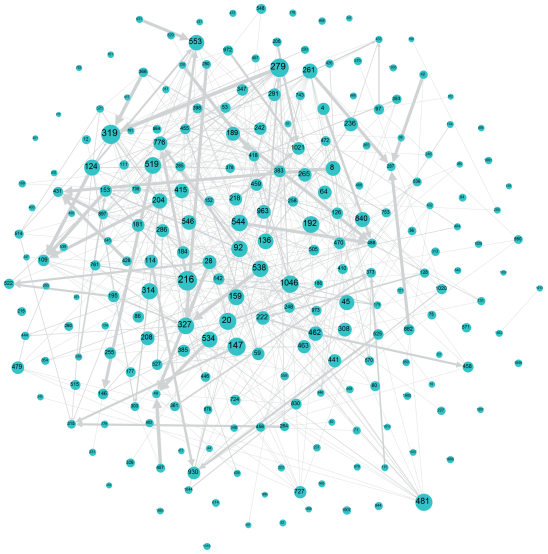
[29] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1225–1234.

[30] Mason Wright and Yevgeniy Vorobeychik. 2015. Mechanism design for team formation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

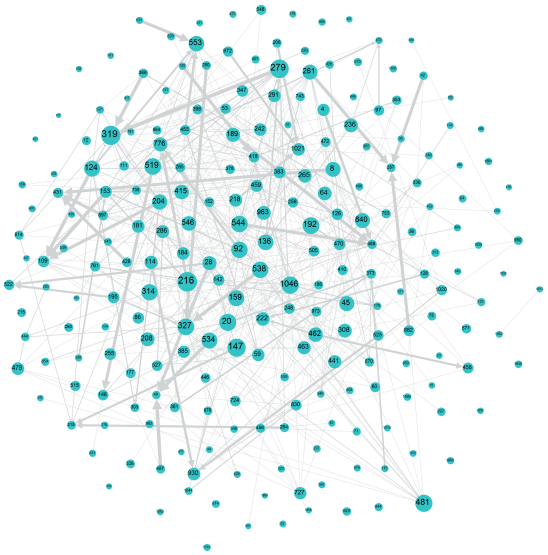
[31] Lei Zhang, Jun Wu, Zhong-Cun Wang, and Chong-Jun Wang. 2010. A factor-based model for context-sensitive skill rating systems. In *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, Vol. 2. IEEE, 249–255.

A DATA ANALYSIS ON NBA DATASET

Figure 7 shows the Synergy Graph and Suppression Graph built from the records in NBA dataset, alongside the description of typical players in Table 4.



(a) Synergy Graph



(b) Suppression Graph

Figure 7: Relationship Graphs in NBA dataset

Several interesting observations are found here:

Synergy Graph: In online games, usually from the synergy graph we can conclude which pairs of nodes have beneficial interactions (from the orange links) when teamed up. In NBA games, players usually stay in only one team during a match season. Therefore, the synergy graph shows a bit different characteristics:

Table 4: List of players in NBA dataset.

player id	player name	team
20	Giannis Antetokounmpo	Milwaukee Bucks
327	Kyle Lowry	Raptors
314	Kawhi Leonard	Raptors
255	Serge Ibaka	Raptors
479	Pascal Siakam	Raptors
527	Fred VanVleet	Raptors
153	Kevin Durant	Warriors
204	Draymond Green	Warriors
124	Stephen Curry	Warriors
514	Klay Thompson	Warriors
216	James Harden	Rockets
92	Clint Capela	Rockets
415	Chris Paul	Rockets
279	Nikola Jokic	Nuggets
366	Jamal Murray	Nuggets
383	Paul Millsap	Nuggets
319	Damian Lillard	Trail Blazers
544	Russell Westbrook	Thunder
488	Ish Smith	Pistons
⋮	⋮	⋮

(1) It tends to have local communities on the graph, as the cooperation usually occur among a group of certain players.
(2) The size of vertices is proportional to the appearance frequency of the players, and this dataset collects only the top points/rebounces/assists players of each team in a match, so we can speculate that players like James Harden(216), Giannis Antetokounmpo(20) and Damian Lillard(319) had good performance within their teams.
(3) The remarkable combination of players, i.e., the orange triangle of James Harden(261), Chris Paul(415), Clint Capela(92), is related to the fact that the team *Rockets* had seldom lost the games when these three players participated.
(4) The local community of Kyle Lowry(327), Kawhi Leonard(314), Serge Ibaka(255), Pascal Siakam(479) and Fred VanVleet(527), corresponds to the NBA Champion of the 2018-2019 season. It's commented that *Raptors* won the champion under the leadership of Kawhi Leonard(314) and Kyle Lowry(327), and another three teammates are believed to have made big contribution.

Suppression Graph: Unlike the suppression graph that indicates the advantage of one individual over another in online games, the suppressive relation in NBA dataset is more related to the relation that one team defeated another.

(1) The links pointing from Nikola Jokic(279) and Jamal Murray(366) to Damian Lillard(319) imply an advantage of the team *Nuggets* over the team *Trail Blazers*.
(2) Similarly, the team *Thunder* had a perfect record against the team *Pistons*, which is reflected by the link from Russell Westbrook(544) to Ish Smith(488).

B EXPERIMENT DETAILS ON FEVER BASKETBALL DATASET

B.1 Feature Design

Table 5 lists the handcrafted features we built for players in Fever Basketball dataset alongside their descriptions.

Table 5: Handcrafted features for prediction on the Fever Backetball dataset. (The *average* operation is performed at match-wise)

Feature	Description
role_level	role level of the player account
ass_num_avg	average assistance
ballblk_num_avg	average times of being blocked
ballpass_num_avg	average passes
ballstolen_num_avg	average times of being stolen
balltouch_num_avg	average times of ball touching
blk_num_avg	average blocks
game_times	total game times
win_game_times	total win game times
lose_game_times	total lose game times
mentor_id_times_avg	average times of mentor guidance
mvp_rate	the mvp rate
overtime_rate	the overtime rate
reb_num_avg	average rebounces
role_score_avg	average scores by player
skill_use_times_avg	statistics of skill uses
steal_num_avg	average steal times
three_point_fieldgoal_avg	average 3-point fieldgoals
three_point_shoot_avg	average 3-point shoots
two_point_fieldgoal_avg	average 2-point fieldgoals
two_point_shoot_avg	average 2-point shoots
wideopen_num_avg	average wideopen scores

B.2 Experimental Setup

The offline/online experiments are performed in Python.

- *Logistic Regression* is implemented with scikit-learn Python package[5]. We used SAGA as the solver for the optimization.
- *XGBoost* is implemented with its official Python package[6]. We trained 1000 trees of maximum depth 4 with a learning rate of 0.1 using 8 parallel threads.
- *BalanceNet* implements a multilayer neural network with Keras Python package whose input is players' feature vectors and output is the probability/score difference that team A wins over team B.
- *Blade-Chest* employs a blade layer and a chest layer in the neural network, through which each team gets a blade vector and a chest vector. The model predicts match outcome based on these two vectors.
- *HOI* implements a factorization-based model to describe pairwise interactions between players by tensorflow.

[5]https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model
[6]https://xgboost.ai

B.3 Comparison of Hero2Vec Methods

B.3.1 Unsupervised Hero2Vec. Various definitions of link weights to construct the relationship graphs are available. We tried different definitions to build the graphs and testify their effects with the downstream prediction task. For the win/lose prediction task on Fever Basketball dataset, Table 6 and Table 7 show the prediction performance evaluated with prediction accuracy.

Table 6: Comparison of network construction methods for synergy graph. f_{win} is the times of winning together; p_{win} is the win rate of the pair of heroes; p'_{win} is the win rate counted only on the matches where the hero pair appeared at one side.

Link Weight Definition	Prediction Algorithms		
	LR	XGBOOST	OptMatch
$f_{win} - f_{lose}$	0.619	0.615	0.639
$p_{win} - p_{lose}$	0.594	0.617	0.625
$p'_{win} - p'_{lose}$	0.613	0.609	0.630

Table 7: Comparison of network construction methods for suppression graph. f refers to the times and p refers to the defeat rate.

Link Weight Definition	Prediction Algorithms		
	LR	XGBOOST	OptMatch
$f_{(a \text{ defeat } b)}$	0.621	0.617	0.640
$f_{(a \text{ defeat } b)} - f_{(b \text{ defeat } a)}$	0.601	0.607	0.619
$p_{(a \text{ defeat } b)}$	0.618	0.619	0.630

It's important to note that no definition of link weights is promised to outperform the others. Depending on the dataset, different definitions encode different information to some degree. It's worth trying different construction methods and pick the most suitable one for a given case.

B.3.2 Supervised Hero2Vec. Another question arises naturally after looking into the offline learn phase of OptMatch: What is the advantage of building the relationship graphs and learning the latent vectors, comparing to adding one embedding layer in the Neural Network? To investigate this problem, we test the prediction performance of the OptMatch-Net model, by adopting pre-trained embedding like claimed in the paper, and by adding an embedding layer before OptMatch-Net. Results are provided in Table 8.

Table 8: Comparison of network embedding methods.

Embedding Category	Accuracy	Time Efficiency	
		Train	Predict
Pre-trained Embedding	0.639	24.85s	5.52×10^{-7}s
Embedding Layer	0.637	259.59s	0.33s

With the research in existing literature and the experiment results, we are able to conclude that pre-trained embedding vectors lead to comparable performance with adding the embedding layer in the neural network. Pre-trained embedding vectors can be considered somehow incorporating the expert knowledge. Besides, with much fewer parameters to employ, methods with pre-trained embedding vectors gain a great improvement in time efficiency.