Forming Competitive Teams in Video Games

Patrick Boutet
University of British Columbia
pboutet@cs.ubc.ca

Sarah Elhammadi
University of British Columbia
shammadi@cs.ubc.ca

Alexandra Kim
University of British Columbia
kimalser@cs.ubc.ca

ABSTRACT

Creating competing teams in online multiplayer games is called matchmaking. The goal of a matchmaking system is to form teams from a pool of players and match the teams against each other, such that the chances of winning the game are roughly the same for two opposing teams. Current matchmaking systems mainly consider players' ranks, and while they do form balanced teams from the rank point of view, the team formation might not always be fair if one of the teams has a more complete group skill set, compared to the other team. In this paper, we are making an attempt to improve over the existing matchmaking systems by taking into account both ranks of the players as well as their experiences in the game. We show that the described problem of forming teams is NP-hard and present a greedy algorithm as one of the possible solutions. We also evaluate our findings using open data provided by the Riot Games API that contains statistics on players and their matches in the online multiplayer game League of Legends (LoL).

1. INTRODUCTION

Designing a fair matchmaking system for multiplayer video games is important as it may impact players' satisfaction and their loyalty to the video game, which in turn affects the company's revenue. To the best of our knowledge, there has not been much published work in this area, with most works focusing on the analysis of current matchmaking systems but offering no formal formulation or solution. In this paper, we provide the problem definition and design our solution for matchmaking in the online game League of Legends [1].

League of Legends is a team-based strategy game. There are two teams playing against each other in a combative arena, with each team having five players. The goal of the game is to take down the opponents' nexus located in the centre of their base before they destroy one's own team nexus. The purpose of matchmaking is to form two teams from a pool of players so that both teams have approximately the same chance to win the game, that is each team should have a 50% chance of winning [2]. The teams are formed by considering each player's ranking, which is calculated by a method based on the Elo rating. This rating system was originally created for reevaluating ratings of chess players based on their ratings before a match and the match outcome [3; 4].

However, there is a bit more nuance to League of Legends. At the start of the game, each player chooses a character to play, also known as champions. There are multiple positions on the map where the players need to place their champions, which we will refer to as lanes. The role that a player chooses for their character almost certainly defines what lane the player is going to be at most of the game. Every champion is most effective only in a limited number of roles in the game, which is in most cases one role, sometimes two and very rarely more than two roles.

In this paper, we will show that in ranked games, players on average have a stronger preference for a single champion role. Moreover, there is a positive correlation between having a more diverse team and winning the game, which is why when choosing champions, players should try to pick champions of varying roles. For the above two reasons, it is reasonable to suggest an enhancement to the current matchmaking system to not only consider players' ranks, but their playing habits as well to allow a fairer game. Therefore, we formally define the problem of matchmaking with role coverage (MMRC), we show that the MMRC problem is NP-hard, and propose a greedy competing team formation algorithm to solve the problem. We then evaluate our solution against existing League of Legends match data to show that we can form more diverse teams without sacrificing the homogeneity of ranks, within and between teams, that the current matchmaking system provides.

The rest of the paper is structured as follows. Section 2 outlines our motivations for this paper. Section 3 follows, providing background on related work. We cover the process of League of Legends data gathering in section 4. Then in section 5, we analyze the League of Legends game data to show why it is reasonable to make changes to current matchmaking systems. Section 6 formalizes the problem and notation. In section 7, we study the complexity of the problem and propose a greedy algorithm. Section 8 presents an evaluation of the algorithm against real game data. Section 9 discusses some key points of our paper. Then in section 10, we cover challenges and weaknesses of our approach. Sections 11 and 12 discuss the direction of future work and provide conclusions to the paper.

2. MOTIVATION

Motivation for proposing an enhancement to competitive team matchmaking systems came from observing a very unique application of team formation to the very popular domain of online multiplayer competitive team based video games. Competitive team based games such as League of Legends, DOTA2, Heroes of the Storm, Overwatch, Smite and others are the cornerstone of the currently exploding eSports industry. Early 2017 forecasts predicted 2017 to hit \$696 million in revenue and forecast it to grow to \$1.5 billion by 2020 [5]. Actual number for 2017 showed revenue of \$1.5 billion with new estimates of growth nearing \$2.3 billion by 2020. [6]. Making balanced teams to ensure fair competition and player satisfaction in game can have substantial impacts on a game developers' bottom line. With this in mind, we explored existing research in the field of competing team matchmaking in online games and found that there was a definite lack in any formal formulation or solution of the problem, which we explore further in section 3. Furthermore, we explored game developer media releases and discussions regarding the matchmaking systems in their games, all of which provided limited details on the actual implementations or how their systems functioned. We suspect this is to limit the exposure of such systems to cheating or exploitation. Next, we explored existing works in team formation and found that there were none that extended team formation to forming competing teams, which we also discuss further in section 3. The problem also proved itself challenging and interesting from a theoretical point of view: a multi-objective optimization problem, NP-hardness, real world time complexity implications, and potential greedy algorithm solution. With each of these aspects building upon the next, we felt a strong motivation to propose our work herein as meaningful contributions to an area little studied but ripe with potential.

3. RELATED WORK

With the incredible popularity of League of Legends - its 2016 monthly users numbered 100 million [7] - there is a surprising lack of research on the subject of matchmaking within these types of competitive team based games, more commonly known as MOBAs (Multiplayer Online Battle Arenas). The work that does exist typically falls into two camps. In the first camp, we have player skill analysis or how to more accurately evaluate and rate a players skill, which is the primary determinant in matchmaking. In the second, we see modifications of matchmaking systems themselves in order to form more balanced teams.

Recent work by Chen et al. looked at League of Legends using a model-based analysis approach and found that base skill of a player, base skill of champions, and champion-specific skills of a player were the largest attributes contributing to a player's skill. They suggest that these should be incorporated into determining a player's skill rating instead of purely basing it on match win/loss records [8]. Suznjevic, Matijasevic and Konfic proposed an **A**pplication **C**ontext **A**ware **R**ating algor**I**thm (ACARI) for determining a player's skill rating over a more traditional Elo approach, which is typically based solely on match wins/losses [9]. interestingly, ACARI takes into account players' role; an important trend seen throughout other works as well [10; 11; 12]. Prakannoppakun and Sinthupinyo propose a novel neural network method to determine the skill rating of a player [13]. They compare this against the Elo rating method and show that their method is more accurate in predicting match outcome and much better when it considers members of the team as part of the analysis.

In the second camp, focusing on modifying the matchmaking system, we find several proposals on ways to modify these systems - many of which recommend incorporating a role based approach, yet most lack a formal formulation of the problem or solution of a matchmaking algorithm. Myślak and Deja are one of the first to analyze League of Legends match data and suggest that the existing matchmaking system does not use the internal structure of gameplay [10]. They focus particularly on player role and show that teams with a proper role distribution win more matches. This leads them to suggest a matchmaking system that should implicitly take into account forming teams that have full role coverage by utilizing player in-game preferences for a particular role based on their match history [10]. Jiménez-Rodrguez et al. suggest a role based approach for matchmaking in online multiplayer games that can easily be extended to incorporate other user defined preferences or past experiences to improve player satisfaction [11]. Claypool et al. perform an in-depth analysis of the effectiveness of League of Legends matchmaking system through match history data analysis and a user study. Their work focuses primarily on the disconnect between game balance from the matchmaking system and player opinions of game balance. Interestingly, even if a game is balanced by the matchmaking system, if players win a match they perceive it as slightly unbalanced, but when they lose they find the teams to be more unbalanced. Players find games they win more enjoyable, and surprisingly, games unbalanced in their favour are found to be most enjoyable. This has lead to some interesting suggestions on how matchmaking systems should take into account player losing streaks and other in-game attributes [14]. Pobiedina et al. looked at the cooperation and success of teams in online games [12]. Through an in-depth analysis of match results of a different yet very similar MOBA, DOTA2, they found that a proper role distribution in teams increases the team's chances of winning [12]. Lastly, for works that formalized the problem and proposed a solution, we have Delalleau et al. with their novel neural network evaluation system for matchmaking [15]. Their BalanceNet is a step in the right direction and given better data may prove to be an effective way to form more balanced teams. Chen also continues his work in this area, recently proposing a novel matchmaking framework (EOMM) [16] with the objective of optimizing player engagement. It has been evaluated on 1vs1 games, but it is proposed that it can be extended to multiplayer games involving teams.

We investigate work from Lappas, Liu and Terzi, which examines the idea of forming teams to solve a task requiring specific skill coverage [17]. In particular, their 2009 paper *Finding a team of experts in social networks* formulates the problem of team formation: Given the set of n individuals $X = \{1, ..., n\}$, a graph $G(X, E)$, and task T, find $X' \subseteq X$, so that $C(X', T) = T$, and the communication cost $Cc(X')$ is minimized. They propose several algorithms to solving variations of the problem and we draw upon them as inspiration for our solution to forming competing teams. To the best of our knowledge, no team formation works have been applied to the context of forming competing teams. We feel that this opens up an interesting opportunity to apply a considerable amount of existing research and theory to a new domain with massive amounts of data and real world relevance.

4. DATA GATHERING

4.1 API Access

To evaluate our algorithm we gathered data from League of Legends, the most popular and accessible MOBA. The first step was to gain access to their game data API [18], which after creating an account to evaluate the contents of their API, we determined the level of player game, match and rank data was sufficient for our work. All accounts receive an initial development API key that is severely rate limited; after consulting their research portal page [19], we submitted a personal project application to receive a production API key with very large rate limits. With wait times over a month on project approval, we were able to expedite the application by discussing the project with one of Riots API developers, Riot Tuxedo, on their RiotAPIDevCommunity discord server [20]. In order to gather match data, a crawling process is required that involves first getting a small number of player ids, getting those players match histories, obtaining match details, extracting the other 9 player ids from those matches, and repeating the crawling process to continue gathering data. This crawling processed proved to be too slow and inefficient to gather a large number of matches between a specific date range across all game server regions[1].

4.2 Data Partner

To speed up the data gathering process we reached out to several League of Legends community statistic sites as potential data providers. Fortunately, one responded - Team Solo Mid - and after discussing our project with one of their software engineers, they shared with us over 7.5 million match ids and their corresponding region code that covered matches played out over a timeframe of June 2017 to November 2017.

4.3 API Endpoints

With this match id dataset and our production API key, we wrote python scripts to call several game API endpoint. First was the `/lol/match/v3/matches/{matchId}` endpoint that gave us a response json containing the match outcome details. This included the champion each player chose, the role they played during the match and which team won. As match outcome details were retrieved the player ids were parsed out and another script called the `/lol/league/v3/positions/by-summoner/{summonerId}` endpoint to gather a player's current rank. Players are divided across seven ranked tiers (Bronze, Silver, Gold, Platinum, Diamond, Master, Challenger). With five of these tiers (Bronze, Silver, Gold, Platinum, Diamond) having sub-tiers (V, IV, III, II, I). These tiers are Elo based, but the true Elo is hidden by the game designers. To solve this issue we did a rough conversion based on the game's earlier version when the game still displayed the Elo rank [21]. Lastly the `/lol/champion-mastery/v3/champion-masteries/by-summoner/{summonerId}` endpoint was used to gather the frequency at which players play a specific champion. Whenever a player plays any of the 139 champions in the game, they accumulate champion mastery points for that champion [22]. Procuring champion mastery points is fairly linear: the more a player plays a given champion, the higher their mastery points for that champion. This endpoint returned a player's ordered list of champions and their mastery score. We used this as a proxy to infer the preferred role of a player instead of doing a historic match outcome analysis on each player, as it would have been too time consuming.

[1]A server region is a local geographic zone where League of Legends is played (http://leagueoflegends.wikia.com/wiki/Servers).

4.4 Data Storage

These scripts were deployed to an AWS EC2 server, to ensure a stable connection to the API, and the retrieved data was stored to a MySQL server for easy access. Upon request, our data is available as a MySQL dump file for anyone wishing to utilize it for future analysis.

5. GAME ANALYSIS

As mentioned in section 3, there were several works that suggest that solely Elo based matchmaking might be inferior to those that take into account more user variables. In this section, we explain why we propose to include players' preferred roles into the current matchmaking systems. We analyzed over 700 000 ranked games[2] in League of Legends to discover general preferences of the players, as well as how team diversity affects match outcomes.

First, we will find out whether or not players generally have role preferences in League of Legends. To do that we will consider mastery scores of each player [22]. A mastery score of a champion reflects how much a player has played that champion in the past. While considering the entire game history might be an oversimplification, it serves as a good enough approximation of players' preferences. Using a sliding window (e.g. considering last n matches or matches in the past n months) would be more accurate, but unfortunately due to time constraints we could not obtain such detailed data, and therefore we use mastery scores that were earned by each player starting from their very fist game. We discuss the implications of this further in section 10, where we review challenges and limitations.

Fig 1 shows role mastery score distribution, averaged for all players in Bronze/Silver/Gold, Platinum/Diamond and Master/Challenger tiers[3]. As can be seen from the distributions, across all tiers, players generally have preferred roles; it is most visible in the highest tiers (i.e. Master and Challenger, fig. 1c). Therefore, to provide higher player satisfaction, it is reasonable to include these unique player preferences into account when forming teams.

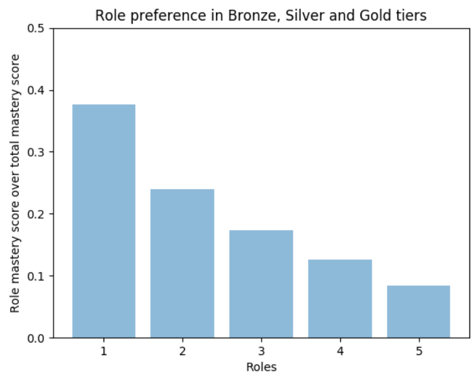
To find out how team diversity is related to match outcomes, we filter the matches to only consider the ones in which the two teams have an unequal number of roles. Then we calculate a point biserial correlation coefficient (r_{pb}) between the number of roles per team and a Boolean variable that indicates whether each team has won or lost the match. As expected, there is a positive correlation between the two variables, with $r_{pb} \approx 0.611$, meaning that more diverse teams are more likely to win a game when matched against a less diverse team.

6. PROBLEM DEFINITION

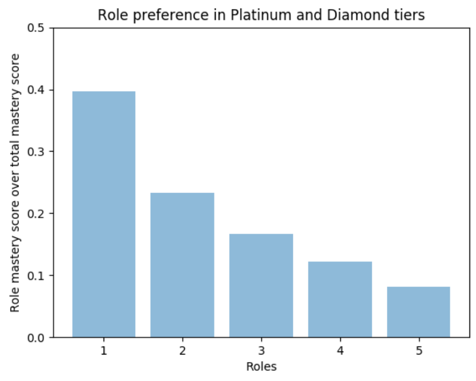
From the point of view of a matchmaking system, there is a pool of players, each with their unique champion/role preferences and ranks, and the goal is to form two teams

[2]Ranked games are competitive games where match outcomes affect players' rankings.

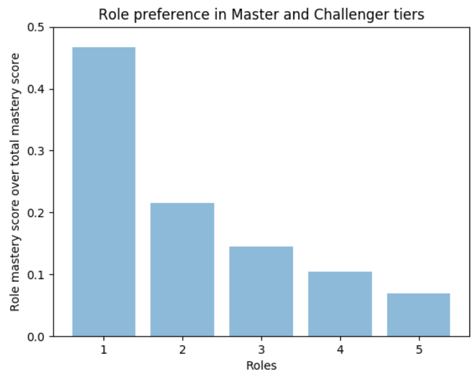
[3]A tier that a player is in is defined by the player's ranking. The tiers in ascending order are Bronze, Silver, Gold, Platinum, Diamond, Master, Challenger.



(a) Mastery score distribution in Bronze, Silver and Gold tiers



(b) Mastery score distribution in Platinum and Diamond tiers



(c) Mastery score distribution in Master and Challenger tiers

Figure 1: Role preferences approximated by mastery score distributions, sorted in descending order. The leftmost bar indicates the most preferred role and the rightmost bar indicates the least preferred role.

that would have equal chances to win the game. Formally, the problem is defined as follows.

Problem 1. *Given a set of players $P = \{1, ..., n\}$, a set of roles $R = \{1, ..., m\}$, and a set of champions $C = \{1, ..., l\}$, form two teams $T_k \subset P$, $k \in \{1, 2\}$ of m players each (i.e., $|T_k| = m$), such that all m roles are covered in each team, and the standard deviation of ranks within one team and the absolute difference between average ranks of the two teams, are minimized. Each champion j assumes a set of roles $C_j \subset R$ and each player i has a rank e_i and a mastery score s_j for each champion they plays, i.e., $p_i = \{r_i, (j, s_j)|j \subset C\}$. We denote the most preferred role for a player i as r_i. The most preferred role for a player is the role of the champion for which they have the highest mastery score. We denote the roles covered by a team T_k as $F(T_k)$*

We call the matchmaking problem defined above as Matchmaking with Role Coverage (MMRC) problem.

7. GREEDY APPROACH

In this section, we study the complexity of the MMRC problem and propose a greedy algorithm to solve the MMRC.

7.1 Complexity

We consider the MMRC problem as outlined in section 6.

Proposition 1. *The MMRC problem is NP-hard.*

Proof. We prove the proposition by a reduction from the Number Partitioning Problem (NPP). An instance of the NPP consists of a list $a_1, a_2, ..., a_N$ of positive integers. The NPP seeks to find a partition, i.e. a subset $A \subset \{1, ..., N\}$ such that the discrepancy $E(A) = |\sum_{i\in A} a_i - \sum_{i\notin A} a_i|$ is minimized. We transform an instance of the NPP problem into MMRC problem by forming a team of $N/2$ players each and the number of roles to one. Then there exists a partition A of N integers such that the discrepancy $E(A)$ is minimized if and only if there exists a partition of the N players such that the difference in the average rank is minimized. $\square$

7.2 Greedy Algorithm

We propose a greedy approach for the MMRC problem. The pseudo code for this algorithm is shown in Algorithm 1. Initially, both teams start with no players and then from the pool of available players, the algorithm greedily selects a player who minimizes the objective as defined in section 6. The average rank of the team T_k after adding a player i is computed as follows:

$$e_{avg}(T_k \cup i) = \frac{e_i + \sum_{s\in T_k} e_s}{1 + |T_k|}$$

while the standard deviation is computed as:

$$e_{sd}(T_k \cup i) = \sqrt{\frac{1}{1+|T_k|}((e_i - e_{avg}(T_k \cup i))^2 + \sum_{s\in S}(e_s - e_{avg}(T_k \cup i))^2)}$$

The algorithm starts with choosing some player p^* from a pool of players based on some heuristics (e.g. random, or player who has been waiting for the longest time, etc.). It then iterates through the pool of players to find the player who optimizes the objective function (i.e. the difference between average team ranks and standard deviation of ranks within each team are minimized). An extra condition is that every player chosen for any of the teams should "cover" a role that has not yet been covered by other players in the team. This is done by looking at the value r_i that reflects which role is preferred by the player i and the Boolean vector $F(T_k)$ that shows which roles a team k covers. If there is no such player who can cover the needed role, we choose the player who only minimizes the objective, without covering the specified role. α is a weight parameter, indicating how important homogeneity of ranks within a team is, compared to that between two teams. The algorithm iterates until both teams are filled. The time complexity of the greedy algorithm is $O(m^2n)$.

It is important to mention here that we take an implicit approach with our solution to the competing team matchmaking problem. This means that the teams that our algorithm forms do not force a player to play the role that the algorithm assumed they would when forming the teams. Instead we form teams with the intent that if the players choose their preferred role, each player will get the role they want, and not have to compete with another teammate for it. This would lead to a better player experience in regards to them getting to play the role they want and having better odds at winning the game because they have an optimal team role distribution.

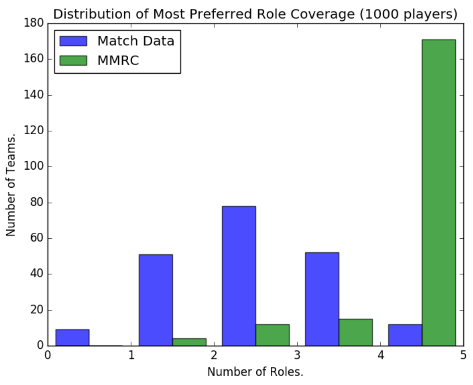
Algorithm 1 Greedy competing team formation algorithm

```
Input: Set of players P
Output: Teams T1 and T2
1: $T_1 \leftarrow \{p^*\}$
2: $T_2 \leftarrow \emptyset$
3: while $(|T_1| < m)$ or $(|T_2| < m)$ do
4:     $k \leftarrow \text{argmin}_{k \in \{1,2\}} |T_k|$          $\triangleright$ if equal, return 1
5:     $p \leftarrow \text{argmin}_{p \in P, r_p \notin F(T_k)} \{|e_{avg}(T_k \cup p) -$
       $- e_{avg}(T_{k' \neq k})| + \alpha \cdot e_{sd}(T_k \cup p)\}$
6:     $T_k \leftarrow T_k \cup p$
7:     $F(T_k) \leftarrow F(T_k) \cup r_p$
8: return $T_1, T_2$
```

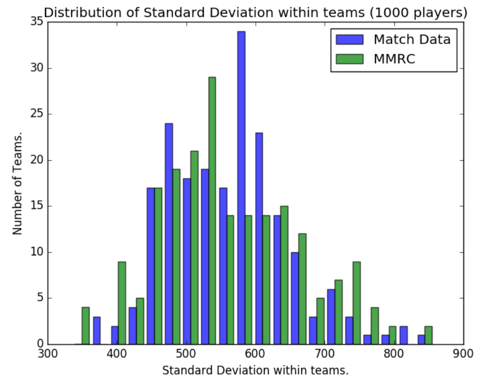
8. EVALUATION

For the evaluation of our greedy algorithm, we compare the teams formed by our approach with those retrieved from the actual League of Legends match data. We simulate the player pool by sampling matches from the data and use players that participated in those matches as our player pool. We then compare the teams from the sampled matches to the teams that our algorithm was able to form from the simulated player pool.

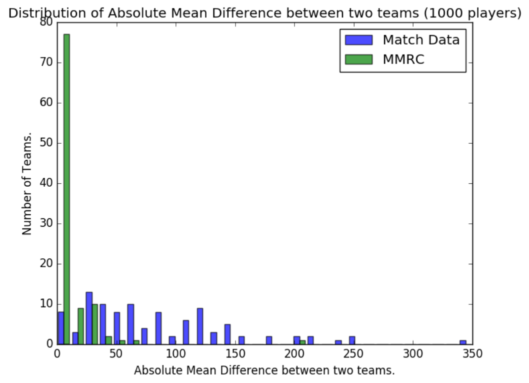
We evaluate the performance of our algorithm in terms of the most preferred role coverage per team, the homogeneity of ranks within one team and between the two teams. Concretely, we compute the most preferred role coverage and rank homogeneity for unique teams across a player pool of 1000 from the League of Legends match data. And, as described above, using the same pool of players, we form teams using our algorithm with $\alpha = 1$, and compute the same measures. We perform identical evaluation for a player pool of 10000 and 50000 as well. The results for all three evaluations are shown in figures 2, 3 and 4. As expected, teams formed by our algorithm have better most preferred role coverage without sacrificing the rank homogeneity within each team and between the teams. In fact, our algorithm performs even better in terms of absolute difference of mean ranks between teams using different pools of players.



(a) Distribution of the most preferred role Coverage.



(b) Distribution of the standard deviation of the ranks within teams.



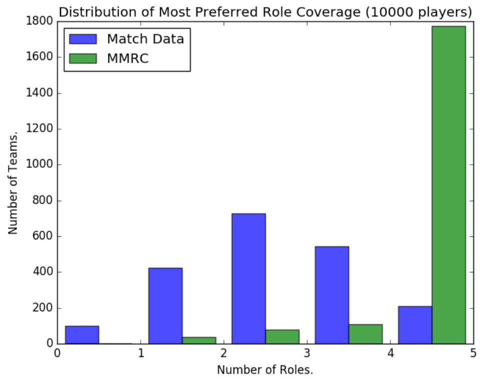
(c) Distribution of the absolute difference of the Mean Ranks between teams.

Figure 2: Distribution of the most preferred role Coverage, the standard deviation of the ranks within teams and the absolute difference of the Mean Ranks between teams for a player pool of 1000.

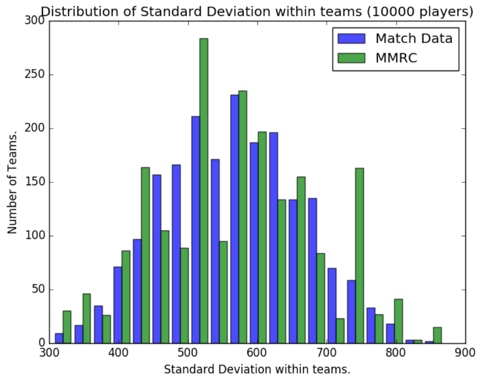
9. DISCUSSION

In the previous section, we found that our algorithm performed better in regards to optimizing for role coverage than was found in the real match data. We believe this method would help in forming competitive teams and result in fairer game play.

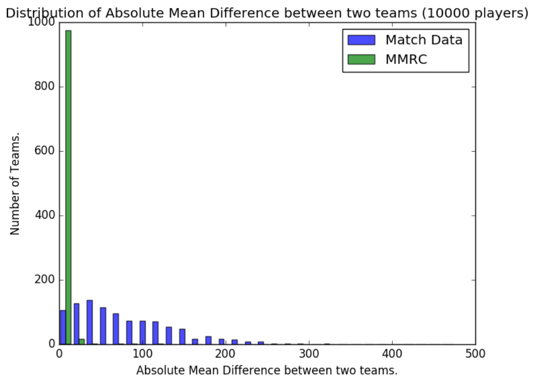
Based on comparison of the actually composed teams and the teams formed by our algorithm, our method outper-



(a) Distribution of the most preferred role Coverage.

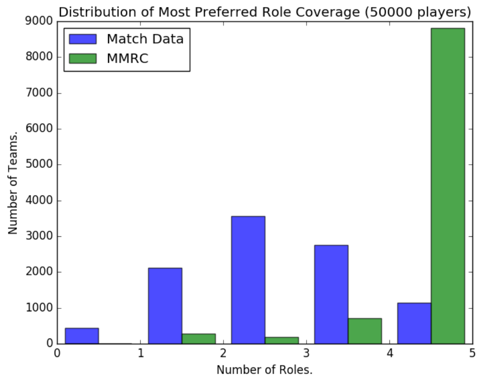


(b) Distribution of the standard deviation of the ranks within teams.

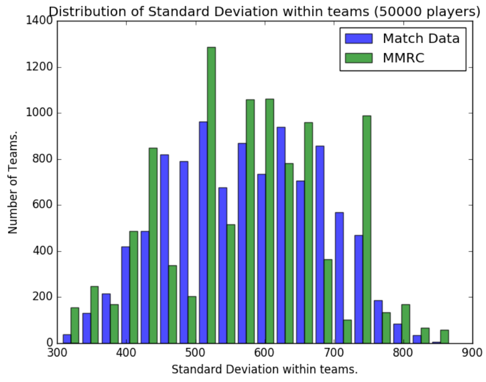


(c) Distribution of the absolute difference of the Mean Ranks between teams.

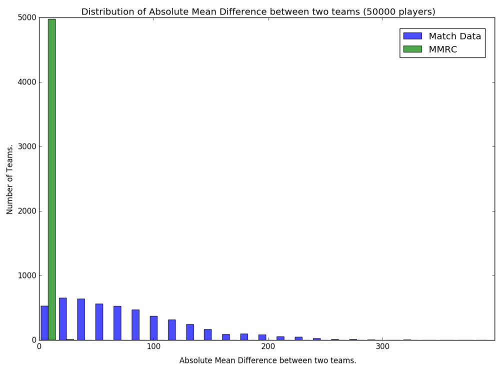
Figure 3: Distribution of the most preferred role Coverage, the standard deviation of the ranks within teams and the absolute difference of the Mean Ranks between teams for a player pool of 10000.



(a) Distribution of the most preferred role Coverage.



(b) Distribution of the standard deviation of the ranks within teams.



(c) Distribution of the absolute difference of the Mean Ranks between teams.

Figure 4: Distribution of the most preferred role Coverage, the standard deviation of the ranks within teams and the absolute difference of the Mean Ranks between teams for a player pool of 50000.

formed the original matchmaking system in terms of minimizing rank differences. We observe that the original matchmaking system optimizes for rank homogeneity as well, and we expected it to perform at least as well as our method. One possible reason why it did not is because the pool of players available at the time could be different from the simulated pool that we consider. The original matchmaking system could probably also optimize for other metrics that we did not consider in our algorithm (e.g. queue times).

In section 7, we discussed that our greedy algorithm solution took an implicit approach when forming teams. However, League of Legends began developing a new mechanism of explicit role preference in their matchmaking system in recent years. Upon further investigation, it appears that League of Legends realized the importance of role distribution on the success of a team to win a match. In 2013, they introduced a new prototype matchmaking system called Team Builder [23] that let players explicitly define the role they wanted to play and the matchmaking system would form a team with a proper role distribution based on what players explicitly defined. The system was never fully adopted as the developers found it too inflexible, having teams unable to change roles before the game started, and reduced the ability of teams to form specific strategies or champion compositions. It was then replaced with a new system called Champ Select in 2016 [24], which was fully adopted by 2017. Champ Select has players explicitly define roles they wish to play and the matchmaking system then forms a team with a proper role distribution and players were permitted to change roles during a period before the game started. Many games such as DOTA2, Heroes of the Storm, Overwatch, Smite and others have not adopted such an explicit role based mechanism. Instead, from what we have observed,these other MOBAs still use a modified Elo based system. Champ Select's explicit method of role preference is an interesting contrast to our own implicit method and is something that game developers should consider when designing matchmaking systems for their specific game context.

10. CHALLENGES AND LIMITATIONS

Using a player's champion mastery scores as a proxy for their preferred role is not an ideal measure of role preference. As a result, our algorithm would not consider if a player decides to begin learning another role or if they play 2 or more roles very proficiently. This can lead to our algorithm forming teams where role conflicts may arise and that could result in a poor player experience and reduced player satisfaction. As mentioned in section 5, a sliding window approach may be more suitable or another game specific measure of player preference could be explored.

We rely on the game for the calculation of what role a player assumed after a match is completed. The accuracy of this in-game system calculation is worth future investigation - should this calculation be incorrect, it could reduce the accuracy of our analysis on player role preferences and if more balanced teams win more often.

Our evaluation is not as robust as it could be due to several factors that are difficult to account for. We form teams at the same time, whereas on a real game server the player pools would likely be fluctuating much more as players come and go and games may form at very different times from one another.

11. FUTURE WORK

One possible improvement over our approach is to consider multiple roles as preferred roles for each player. That could be done by either setting a fixed number of roles, or considering the entire role distribution to see which roles the player prefers most. In the latter case, the likelihood of being assigned a particular role would then depend on the distribution.

Another possibility to improve our method or matchmaking in general is to look at other game metrics. For example, in League of Legends that could be the number of kills and deaths, amount of damage dealt, or amount of gold [4] earned and spent, etc. These metrics would be helpful at giving a better description of each individual's and team's performance, rather than a binary win/loss variable. This could also count towards role coverage; if a player showed good performance at a particular role, perhaps the matchmaking system could suggest the player try that role again by matching them with appropriate teammates.

Looking at other datasets from other multiplayer competitive team based games, such as DOTA2 or Overwatch, could be beneficial in several ways. It would allow us to further potentially show role preferences for players and that roles matter when it comes to a team's chances of winning. It would also allow us to adapt our problem formulation and algorithm to be more flexible in role coverage. An example of this would be in the game Overwatch, where there is typically multiples of the same role on a team vs League of Legends where each role is assumed only once. Developing a generalized framework to optimize for different specific game contexts would most likely benefit new and existing game developers when they look to change or develop new games and/or matchmaking systems.

Lastly, future studies could involve partnering with actual game developers to potentially evaluate modification of existing methods or developing new matchmaking systems on public test servers involving real players and live matchmaking.

[4]In League of Legends gold is an in-game currency

12. CONCLUSIONS

In conclusion, we have shown that players do have preferences for roles in League of Legends and that there is a positive correlation between having a more diverse team and winning the game. Our work offers a possible improvement over current matchmaking systems by taking player preferences into account. We formulate the problem as a matchmaking with role coverage (MMRC) problem and have shown that it is NP-hard. We also offer a greedy solution to the MMRC problem that utilizes player in-game data to form teams that implicitly lead players to play their preferred role. Ideally, our solution should lead to proper role distribution within a team and therefore a more functional team to compete with. We then evaluate our algorithm to show that it forms more diverse teams without sacrificing homogeneity in ranks. We hope to draw more attention to the competing teams formation problem, since the popularity, demand and revenue from MOBAs and eSports creates a large opportunity on a little studied, challenging yet exciting problem.

Acknowledgements

We would like to graciously thank Kyle Young (@TSM-Leaf) and the Solomid Network of websites (solomid.net) for providing 6 months of historic match id and region code data from their own sites: champion.gg and probuilds.net. This constituted 7.5 million matches across all server regions which greatly accelerated the data we were able to collect and analyze for our work.

13. REFERENCES

[1] Riot Games. League of legends. `https://na.leagueoflegends.com/en/`. Accessed: 2017-12-22.

Figure 5: TeamSoloMid logo

[2] VP of Game Design Zileas. Lol matchmaking explained. http://forums.na.leagueoflegends.com/board/showthread.php?t=12029. Accessed: 2017-12-22.

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

[4] Paul CH Albers and Han de Vries. Elo-rating as a tool in the sequential estimation of dominance strengths, 2001.

[5] Aaron Mickunas. Esports revenues will reach $696 million this year and grow to $1.5 billion by 2020 as brand investment doubles. https://newzoo.com/insights/articles/esports-revenues-will-reach-696-million-in-2017/. Accessed: 2018-01-03.

[6] Aaron Mickunas. Riot games reveals 'league of legends' has 100 million monthly players. Report:Theesportsindustrybroughtin$1.5billiontotalrevenuein2017,oncoursetohit$2.3billionby2022. Accessed: 2018-01-02.

[7] Paul Tassi. Riot games reveals 'league of legends' has 100 million monthly players. https://www.forbes.com/sites/insertcoin/2016/09/13/riot-games-reveals-league-of-legends-has-100-million-monthly-players/#583ab2365aa8. Accessed: 2017-12-30.

[8] Z. Chen, Y. Sun, M. Seif El-nasr, and T.-H. D. Nguyen. Player Skill Decomposition in Multiplayer Online Battle Arenas. *ArXiv e-prints*, February 2017.

[9] M. Suznjevic, M. Matijasevic, and J. Konfic. Application context based algorithm for player skill evaluation in moba games. In *2015 International Workshop on Network and Systems Support for Games (NetGames)*, pages 1–6, Dec 2015.

[10] Mateusz Myślak and Dominik Deja. *Developing Game-Structure Sensitive Matchmaking System for Massive-Multiplayer Online Games*, pages 200–208. Springer International Publishing, Cham, 2015.

[11] Jorge Jiménez-Rodrıguez, Guillermo Jiménez-Dıaz, and Belén Dıaz-Agudo. Matchmaking and case-based recommendations. In *19th International Conference on Case Based Reasoning*, 2011.

[12] N. Pobiedina, J. Neidhardt, M. d. C. Calatrava Moreno, L. Grad-Gyenge, and H. Werthner. On successful team formation: Statistical analysis of a multiplayer online game. In *2013 IEEE 15th Conference on Business Informatics*, pages 55–62, July 2013.

[13] N. Prakannoppakun and S. Sinthupinyo. Skill rating method in multiplayer online battle arena. In *2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–6, June 2016.

[14] M. Claypool, J. Decelle, G. Hall, and L. O'Donnell. Surrender at 20? matchmaking in league of legends. In *2015 IEEE Games Entertainment Media Conference (GEM)*, pages 1–4, Oct 2015.

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

[16] Zhengxing Chen, Su Xue, John Kolen, Navid Aghdaie, Kazi A Zaman, Yizhou Sun, and Magy Seif El-Nasr. Eomm: An engagement optimized matchmaking framework. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1143–1150. International World Wide Web Conferences Steering Committee, 2017.

[17] Theodoros Lappas, Kun Liu, and Evimaria Terzi. Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 467–476. ACM, 2009.

[18] Riot Games. Riot developer. https://developer.riotgames.com/. Accessed: 2018-01-01.

[19] Riot Games. Research with riot. https://www.riotgames.com/en/explore/research-with-riot. Accessed: 2018-01-01.

[20] Riot Games. Riot api developer community. https://discord.gg/uYW7qhP. Accessed: 2018-01-01.

[21] League of Legends Wiki. Elo rating system. http://leagueoflegends.wikia.com/wiki/Elo_rating_system. Accessed: 2018-01-01.

[22] Riot Games. Champion mastery. https://na.leagueoflegends.com/en/page/features/champion-mastery. Accessed: 2018-01-01.

[23] STATUS KWOH. Announcing team builder two-day live beta! https://na.leagueoflegends.com/en/news/game-updates/features/announcing-team-builder-two-day-live-beta. Accessed: 2018-01-01.

[24] RIOT LYTE. New champ select replaces team builder. https://na.leagueoflegends.com/en/news/game-updates/features/new-champ-select-replaces-team-builder. Accessed: 2018-01-01.