A Case Against Using Elo or Glicko Algorithms for Rating Players in n vs n Games

Quentin Fruytier
UT Austin
qdf76@my.utexas.edu

Abstract

This course project investigates the intricate task of rating players in n vs n zero-sum games, particularly prevalent in the context of online esports titles where large player bases engage in diverse team-based matchups. We explore and evaluate three well-known rating algorithms: the Elo system, Glicko, and Glicko-2, each building upon its predecessor to account for additional variables. A simulation study is conducted to empirically demonstrate the inadequacies of these algorithms in n vs n scenarios where $n > 1$, highlighting the urgency for tailored solutions. Our findings reveal significant performance drawbacks in current rating systems, especially concerning the failure to consider the performance gap between players during a game. The necessity for a revamped system becomes evident, with a proposed introduction of a rating scale that incorporates game closeness to expedite accurate player rating. The Glicko rating's assumption of a Normal distribution is also challenged by empirical evidence, emphasizing the need for algorithmic adjustments. This project advocates for the development of a novel rating system tailored to n vs n games, emphasizing the incorporation of individual player performance metrics. Despite the challenges, existing performance rating systems in popular esports titles showcase promising avenues for future improvements in the dynamic landscape of player rating and matchmaking.

1 Introduction

Estimating player ratings in x vs x games poses the intricate challenge of gauging the true skill levels of N players, relying solely on game outcomes, performance metrics, and the timestamp t of each game. The primary objective is to accurately deduce these skill ratings while minimizing the number of games needed for the estimation process. This challenge is particularly pronounced in sports contexts, where the assessment of individual skill levels hinges solely on observable game outcomes and performance metrics.

In recent years, sophisticated rating systems have emerged in popular 5 vs 5 online competitive games such as Counterstrike 2, Dota 2, Chess, League of Legends, and Overwatch. In these systems, players initiate with a base rating and subsequently gain or lose rating points based on game outcomes. While the Glicko/Glicko 2 algorithm, developed by Mark Glickman (1) and the Elo Algorithm (2) for rating chess players, are widely acknowledged, many games opt for heavily inspired tailored rating algorithms. Although reports suggest the implementation of some variant of Glicko for their rating systems, it has faced considerable criticism for its suboptimal performance across these popular online esports titles.

The significance of these player ratings cannot be overstated, as they underpin matchmaking algorithms that dictate the composition of teams, ensuring fair and competitive games. The precision of the rating system is paramount for constructing balanced matchups; its absence can lead to one-sided

Preprint. Under review.

games, thereby diminishing overall enjoyment and player engagement. Consequently, the player rating problem assumes considerable importance, especially in the realm of online competitive games.

1.1 Problem Description

Let $\theta_i \sim \Gamma$ be a random variable representing the skill rating of player i, sampled from the player skill distribution Γ. During each match, player i performs with skill $X_i \sim \mathcal{N}(\theta_i, \sigma_i)$, where σ_i is the inconsistency parameter of player i. For a given match, players $1, \ldots, n$ are observed to have performance levels $x_1, \ldots, x_n$, while their opponents, $n+1, \ldots, 2n$, are observed to have performances $x_{n+1}, \ldots, x_{2n}$.

Consider player $k \in \{1, \ldots, N\}$ and match $t \in \mathbb{N}$. Player k is matched with $n-1$ teammates with skill parameters $\theta_{k_2^t}, \ldots, \theta_{k_n^t}$ and n opponents with skill parameters $\theta_{k_n^t}, \ldots, \theta_{k_{2n}^t}$. The outcome of the game is represented by the variable $s_k^t \in \{0, 1\}$, where 1 indicates k's team winning and 0 indicates the opposing team winning. We make the assumption that the probability of player k winning their t-th game given their teammates and opponents is given by

$$\mathbb{P}(s_{i,j}^t = 1 | \theta_k, \theta_{k_2^t}, \ldots, \theta_{k_{2n}^t}) = \frac{1}{1 + \exp\left(\frac{\sum_{j=n+1}^{2n} x_{k_j^t} - \sum_{i=1}^{n} x_{k_i^t}}{173.29}\right)}. \tag{1}$$

Given these modeling choices and observed game outcomes, we can devise an algorithm to update player ratings after every game.

2 Rating Algorithms

The majority of publicly known algorithms for rating players in zero-sum games originated from the world of chess player rankings. The journey began in the 1970s with the development of the Elo rating system (2). Over the years, this system underwent refinement, notably by Glicko in 1995 and its subsequent iteration, Glicko-2, in 2012. Remarkably, these algorithms have found applications in today's most popular competitive e-sports titles, including Counter Strike, Dota 2, League of Legends, and Valorant.

In the discussions that follow, let r_i represent the current estimated rating for player i. We also adopt the assumption that the player skill distribution, denoted as Γ, follows a normal distribution $\mathcal{N}(1500, 350)$. However, this choice itself warrants scrutiny, as player skill distributions are commonly observed to be right-skewed (see Figure 3). It was even suggested in(2) that a Maxwell-Boltzmann distribution is a more accurate representation of the player skill distribution.

In the subsections below, we provide an overview of these rating algorithms, outlining their core concepts, and delve into a discussion of their respective strengths and weaknesses. It's important to note that all the algorithms discussed subsequently can be easily extended to accommodate team-based games with $n > 1$ by treating a team as the sum of its individual parts.

2.1 Elo Rating

The Elo rating system (see Algorithm 1), introduced by Arpad Elo in 1978 for chess (2), was designed to iteratively update a player's ranking after tournaments based on their performance. The fundamental concept involves determining the probability of player k winning a game, given their current Elo rating and that of their opponents. This probability is expressed as

$$\frac{1}{1 + \exp\left(\frac{\theta_{k_2^i} - \theta_k}{173.29}\right)}$$

where $\theta_{k_2^i}$ is player k's opponent's rating and θ_k is player k's rating.

The algorithm then assigns Elo rewards based on the match outcome, adjusted by the estimated probability that player k wins. The adjustment factor is calculated as $s_k^t - \hat{\mathbb{P}}(s_k^t = 1)$. Consequently, player k receives fewer rewards for games predicted in their favor and faces greater penalties for losing games they were expected to win.

However, a notable drawback of the Elo rating system is its reliance on the assumption that the current Elo ratings of opponents accurately represent their true skill ratings. Additionally, the system lacks a mechanism to scale rewards based on the closeness of the game, potentially leading to less nuanced evaluations.

```
Algorithm 1 Elo Rating
    Input: Initial $K \in \mathbb{R}, r_k, (s_k^1, ..., s_k^T)$, $(r_{k_2}^1, ..., r_{k_2}^T)$.
    Update $r_k$:
        $r_k^{'} = r_k + K \sum_{i=1}^{T} \left( s_k^i - \frac{1}{1 + \exp\left(\frac{\theta_{k_2^i} - \theta_k}{173.29}\right)} \right)$    (2)
    End Set $r_k \leftarrow r_k^{'}, RD_k \leftarrow RD_k^{'}$
```

2.2 Glicko Rating

The Glicko rating algorithm, introduced by Mark Glickman in 1995 (1), represents an advancement in estimating the ratings of players engaged in one-on-one zero-sum games. Initially proposed as an enhancement to the Elo rating system for ranking chess players, the Glicko algorithm gained popularity, particularly for its innovative use of the Rating Deviation as a measure of confidence in a player's current rating. This deviation allows for more precise adjustments to a player's rating after each win or loss. The Glicko algorithm has found widespread implementation across various online gaming platforms, including lichess, chess.com, Counterstrike: Global Offensive, Dota 2, Splatoon 2, and others. The algorithm begins by defining a rating period during which all games are assumed to have occurred simultaneously. This period can range from hours to weeks, depending on the specific game. In essence, the Rating Deviation (RD_k) of player k is assumed to remain unchanged over this period during which T games are played. The core idea of the Glicko algorithm is to leverage the rating deviation to calculate 95% confidence intervals for each player's estimated rating, updating their respective ratings based on the algorithm's confidence in the player's rating and the outcome of the games.

Step 1: Initialization of the estimated rating (r_k) and rating deviation (RD_k) of player k at the start of the new rating period. If the player is unrated and has played no games, the initialization is set to $r_k = 1500$ and $RD_k = 350$. If player k has played before, RD_k is set as $\min\{\sqrt{RD_k^2 + c^2}, 350\}$, where c is a constant, and r_k is the rating obtained from the last rating update.

Step 2: Observation of all game outcomes in the rating period and updating player k's rating and rating deviation according to equations (3) and (4). This process is subsequently repeated after each rating period.

It is noteworthy that the player's rating deviation only increases with the passage of time without games played, while it always decreases after any game played. Correctly bounding the rating deviation is crucial to avoid discouraging players from participating, as a small rating deviation leads to minimal rating changes. Although Glicko is designed for zero-sum games, the ratings awarded to players are not necessarily zero-sum and depend on the rating deviation of each player.

```
Algorithm 2 Glicko Algorithm
    Input: Initial $r_k, RD_k, (s_k^1, ..., s_k^T), (r_{k_2}^1, ..., r_{k_2}^T)$, and $(RD_{k_2^1}, ..., RD_{k_2^T})$.
    Update $r_k$:
        $r_k^{'} = r_k + \frac{1}{173.29(1/RD_k^2 + 1/d^2)} \sum_{i=1}^{T} g(RD_{k_2^i}) \left( s_k^t - \frac{1}{1 + e^{-g(RD_{k_2^i})(r_k - r_{k_2}^i)/173.29}} \right)$    (3)
    where $g(RD_i) = 1/\sqrt{1 + 3RD_i^2/(\pi^2 * 173.29)}$, and
    $d^2 = \left( (1/173.29)^2 \sum_{i=1}^{T} g(RD_{k_2^i})^2 \frac{1}{1+e^{-g(RD_{k_2^i})(r_k - r_{k_2}^i)/173.29}} \left( 1 - \frac{1}{1+e^{-g(RD_{k_2^i})(r_k - r_{k_2}^i)/173.29}} \right) \right)^{-1}$.
    Update $RD_k$:
        $RD_k^{'} = \sqrt{(1/RD_k^2 + 1/d^2)^{-1}}$    (4)
    End Set $r_k \leftarrow r_k^{'}, RD_k \leftarrow RD_k^{'}$
```

2.3 Glicko-2

A modification to Glicko in 2012 resulted in the Glicko-2 algorithm, aiming to account for individual player improvement. Introducing a new variable, volatility σ_i, the Glicko-2 algorithm models the consistency of a player at a given time, allowing for adjustments to a player's awarded rating as they improve or worsen. For instance, a player consistently rated r_k may experience a significant improvement in performance, systematically raising their volatility $\sigma_i \in \mathbb{R}_+$. Consequently, the player receives more rating points for each win. The algorithm is described below.

Step 1: Initialization of the estimated rating (r_k), rating deviation (RD_k), and player volatility (σ_i) of player k at the start of the new rating period. If the player is unrated and has played no games, initialization is set to $r_k = 1500$, $RD_k = 350$, and $\sigma_i = 0.06$. These values can be adjusted for specific effects. If player k has played before, RD_k is set as $\min\{\sqrt{RD_k^2 + c^2}, 350\}$, where c is a constant, and r_k is the rating obtained from the last rating update. Next, the rating is normalized to simplify subsequent calculations:

$$\mu_k = (r_k - 1500)/173.29$$
$$\phi_k = RD_k/173.29.$$

Step 2: Computation of the estimated variance of player k's team based only on game outcomes:

$$v = \left(\sum_{i=1}^{T} \frac{1}{1 + 3\phi_{k_2^i}^2/\pi^2} \times \frac{1}{1 + \exp\left(-\frac{(\mu_{k_2^i} - \mu_k)}{\sqrt{1+3\phi_{k_2^i}^2/\pi^2}} \right)} \times \frac{\exp\left(-\frac{(\mu_{k_2^i} - \mu_k)}{\sqrt{1+3\phi_{k_2^i}^2/\pi^2}} \right)}{1 + \exp\left(-\frac{(\mu_{k_2^i} - \mu_k)}{\sqrt{1+3\phi_{k_2^i}^2/\pi^2}} \right)} \right)^{-1}. \quad (5)$$

In the case where player k plays in a team, this assumes that the team remains constant throughout the rating period. The change in performance for player k from game outcomes is then calculated as

$$\Delta = v \sum_{i=1}^{T} \frac{1}{\sqrt{1 + 3\phi_{k_2^i}^2/\pi^2}} \left[s_i^t - \frac{1}{1 + \exp\left(-\frac{(\mu_{k_2^i} - \mu_k)}{\sqrt{1+3\phi_{k_2^i}^2/\pi^2}} \right)} \right]. \quad (6)$$

Step 3: Determination of the updated volatility of player k, $\sigma_k^{'}$, as the zero of

$$f(x) = \frac{e^x(\Delta^2 - \phi^2 - v - e^x)}{2(\phi^2 + v + e^x)^2} - \frac{(x - a)}{\tau^2},$$

where τ is a constant used to constrain the volatility (a good choice is 0.2). Using the new volatility, a better estimate of the pre-period rating deviation of player k is obtained as $\phi_k = \sqrt{\phi_k^2 + \sigma_k^2}$.

Step 4: Update of all estimates based on the rating period's game outcomes:

$$\phi_k^{'} = \frac{1}{\sqrt{1\phi_k^2 + 1/v}} \tag{7}$$

$$\mu_k^{'} = \mu + \phi_k \sum_{i=1}^{T} \left(s_i^t - \frac{1}{1 + \exp\left(-\frac{(\mu_{k_2^i} - \mu_k)}{\sqrt{1+3\phi_{k_2^i}^2/\pi^2}} \right)} \right). \tag{8}$$

Conversion back to the original scale is the final step:

$$\begin{aligned} RD_k^{'} &= 173.29(\phi_k^{'}) \\ r_k^{'} &= 173.29(\mu_k^{'}) + 1500. \end{aligned}$$

3 Simulation Study: Elo Algorithm for n vs n Games

We conduct a simulation study to empirically investigate the performance of the Elo system (Algorithm 1) in n vs n games. Specifically, we aim to understand how enlarging the size of the teams affects the number of games a player, denoted as k, needs to play to reach their true rank.

We generate a pool of approximately 20,000 players by sampling from $\mathcal{N}(1500, 350)$, as assumed in Algorithm 1. Each player's estimated rating is initialized at 1500. In each iteration, we assume that every player in the pool participates in one game. The matchmaking system creates games by pairing the best players against each other until everyone is in a game. The game outcomes are then sampled from a Bernoulli distribution with p given in (1). After the games, we update each player's estimated rating using Algorithm 1.

After t games are played, we calculate the error as the average percentage error in the rank of each player:

$$\text{err}_t = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{rank}_i^t - \text{rank}_i}{N}, \tag{9}$$

where rank_i^t is player i's estimated rank after t games, and rank_i is player i's true rank. The results are presented in Figures 1 and 2.

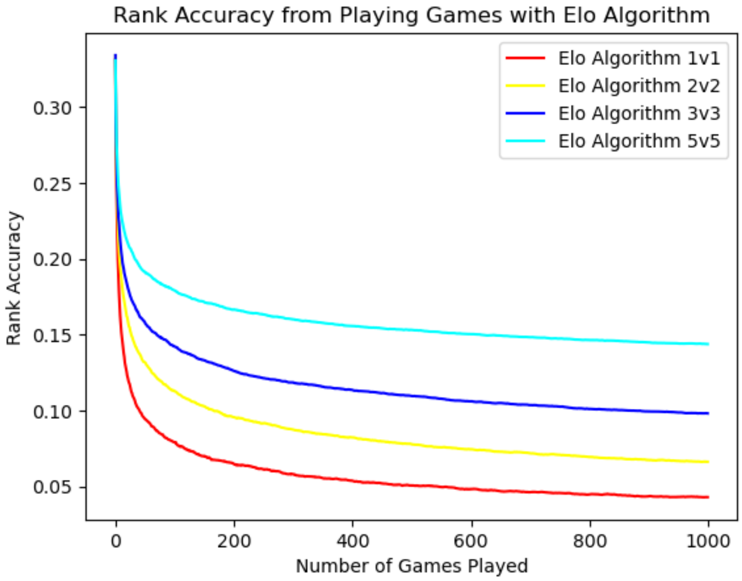


Figure 1: Average percentage rank error against the number of games played for different n vs n settings.

We observe an immediate decrease in performance as n increases, indicating that the Elo system becomes less accurate with larger team sizes. The bias in the system is evident from the plots, and after 1000 games, the average percentage error in rank for 1 vs 1 games is significantly lower than that for 5 vs 5 games—the standard in popular competitive esports titles. This bias is concerning, especially considering that most players play fewer than 1000 games in a year. Our results suggest that even after 1000 games, the average rank error for any player is approximately $\pm(0.15)N$.

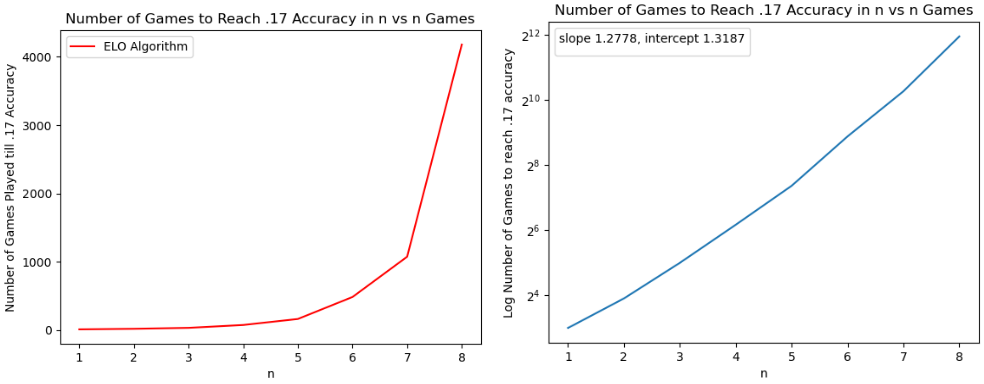


Figure 2: Left side plot shows the number of games required to be played before the average percentage error in rank reaches 0.17 against n. Right side plot depicts the log of the number of games required to be played before the average percentage error in rank reaches 0.17 against n.

We also explore how the number of games required to reach ϵ error scales with n. For this, we continue playing games until the average percentage rank error reaches 0.17 and then plot the number of games played against n. We observe in figure 2 that $\log_2(\text{num\_games})$ scales linearly with n. From this, we approximate that the number of games required to reach ϵ average percentage rank error scales as $\sim 2^{4n/3}$. This implies that for 5 vs 5 games, the Elo system takes approximately 100 times as many games as it would for 1 vs 1 games to achieve the same average percentage error in rank. This finding strongly suggests that Algorithm 1 is not suitable for the $n > 1$ setting.

Finally, we expect the Glicko and Glicko_2 algorithms to experience a similar increase in error rate, as neither algorithm's improvements on the Elo system are likely to mitigate the loss of information on individual player performance in n vs n games where $n > 1$.

4 Conclusion

Throughout this project, we delved into the challenging task of rating players in n vs n zero-sum games, where large player pools engage in diverse matchups with varying teammates and opponents. We explored three prominent publicly available rating algorithms: the Elo system (2), Glicko (1), and Glicko-2 (3). Each algorithm aimed to enhance its predecessor by incorporating additional variables. Our investigation included a simulation study that empirically revealed the poor performance of these algorithms in n vs n games where $n > 1$.

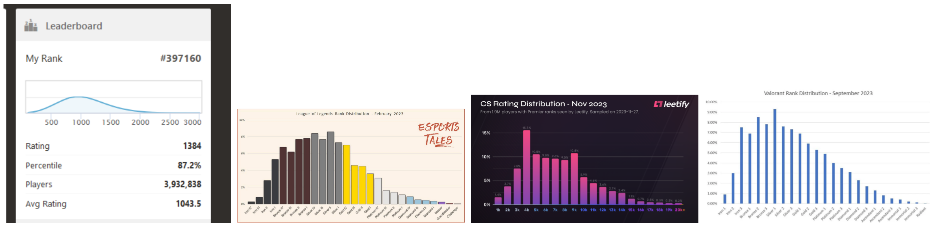
This observation holds particular significance due to the overwhelming popularity of online esports titles, each boasting millions of concurrent players daily, in contrast to traditional 1 vs 1 sports like chess, tennis, and badminton. The widely criticized rating and matchmaking systems, rumored to employ variants of the Elo system or Glicko algorithm, now face scrutiny for their evident shortcomings.

An inherent flaw in these systems is their failure to account for the performance gap between players during a game. Introducing a rating scale that considers game closeness could potentially expedite players' journey to their true rating. Notably, the Glicko rating assumes a Normal distribution with a mean of 1500 and a variance of σ_0^2, a presumption empirically challenged by the right-skewed nature of player skill rating distributions in online games (see Figure 2).

Our analysis underscores the need for a revamped rating system for $n > 1$ scenarios, one that intricately incorporates individual player performance in calculating rating rewards. This introduces a new challenge, requiring accurate methodologies for assessing individual player contributions. Despite this challenge, several popular online esports titles have already established their performance ratings systems, such as the hltv 2.0 (4) and Leetify (5) performance metrics for Counterstrike.

In conclusion, the quest for an effective rating system in the realm of n vs n games remains an ongoing challenge, demanding innovative solutions to accurately capture the dynamic and individualized nature of player performance.

5 Annex



(a) Chess.com (b) League of Legends (c) Counter Strike 2 (d) Valorant

Figure 3: Player rank distributions in popular online games is observed to be right skewed.

References

[1] M. E. Glickman, "Parameter Estimation in Large Dynamic Paired Comparison Experiments," *Journal of the Royal Statistical Society Series C: Applied Statistics*, vol. 48, no. 3, pp. 377–394, 01 2002. [Online]. Available: https://doi.org/10.1111/1467-9876.00159

[2] A. E. Elo, *The Rating of Chessplayers, Past and Present*. New York: Arco Pub., 1978. [Online]. Available: http://www.amazon.com/Rating-Chess-Players-Past-Present/dp/0668047216

[3] [Online]. Available: http://www.glicko.net/glicko.html

[4] Tgwri1s, “Introducing rating 2.0,” Jun 2017. [Online]. Available: https://www.hltv.org/news/20695/introducing-rating-20

[5] A. Ekman, “What is leetify rating?” Dec 2022. [Online]. Available: https://blog.leetify.com/what-is-leetify-rating/#:~:text=Benchmarks%20for%20Leetify%20Rating&amp;text=Zero%20means%20you%20did%20not,odds%20of%20winning%20the%20round.