Abstracting Glicko-2 for Team Games

A thesis presented to the Graduate School of the University of Cincinnati in partial fulfillment

of the requirements for the degree of

Master of Science

in the College of Engineering and Applied Science

by

Garrick J. Williams

B.A. University of Cincinnati

April 2013

Committee Chair: Fred Annexstein, Ph.D.

Abstract

Research experiments were conducted to further the development of rating player skill within the context of online team games with small team sizes. Modern rating systems tend to display inferior performance when individually measuring players within a team of six or less members. This thesis proposes three modifications to rating update algorithms that abstract a match between two teams as multiple one-on-one matches. These update methods were designed for adaptation with existing skill rating systems based on Glicko or Glicko2. Using a robot soccer simulation environment, an automated testing suite was developed to show how these methods affect the behavior of Glicko2. The results of an experiment utilizing this suite show the viability of these methods and paint a picture for future research towards improving rating system performance for team games with small team sizes.

Table of Contents

I. Introduction 1

II. Proposed Update Algorithms .6

2.1 Overview 6

2.2 Individual Update Method 6

2.3 Composite Opponent Update Method 8

2.4 Composite Team Update Method 10

III. Experiment 13

3.1 Overview 13

3.2 Assumptions 13

3.3 Procedure 14

3.4 Players 14

3.5 Matchmaking 14

3.6 Prediction & Error 15

IV. Results 15

4.1 Criteria 15

4.2 Predictive Performance 16

4.3 Ranking vs. Win Percentage 17

4.4 Convergence 20

V. Analysis 21

VI. Conclusion 22

Bibliography 24

I. Introduction

Skill rating plays a vital role in the commercial viability of online games. Competitive play contributes so heavily in a game's longevity that game companies invest significant amounts of money towards electronic sporting events. *Call of Duty*'s annual championships have a prize pool of $1 million. Riot Games invested more than $2.1 million towards the *League of Legends* 2014 World Championship. With the help of crowdfunding, Valve Corporation boasts the highest prize pool with a staggering $10.9 million for *Dota 2*'s 2014 Internationals.[1] Developers that produce online games place incredible value on competitive play for the success of their products and services.

Competitive play contributes to the longevity of an online game in numerous ways. Ranked gaming provides a goal for players to reach, encouraging them to invest time towards improving their skills. This also stimulates the game's player community. Online streaming services highlight the activities of the esports professionals while most players watch and learn from them, applying pro techniques to their gaming. Finally, the competitive scene increases the game's publicity. Riot Games reported that 21 million viewers watched the *League of Legends* 2014 World Finals with each viewer watching an average of 67 minutes of online streamed footage.[2] Ranked gaming draws new players to the game while keeping current players playing.

[1] "Top 100 Largest Overall Prize Pools :: E-Sports Earnings." *Top 100 Largest Overall Prize Pools :: E-Sports Earnings*. Web. 5 Feb. 2015. <http://www.esportsearnings.com/tournaments>.

[2] "Worlds 2014 by the Numbers." *Riot Games*. 1 Dec. 2014. Web. 5 Feb. 2015. <http://www.riotgames.com/articles/20141201/1628/worlds-2014-numbers>.

Without a quality skill rating system, none of this is possible. A game needs a skill rating system to serve as the engine for its competitive play. Highlighted by the researchers behind TrueSkill, a skill rating system provides three primary functions for online games.[3]

1. **Matchmaking**. Rating systems provide a way for games to match players with others of preferably similar skill. Many online games like *League of Legends* and *Hearthstone* use an automated matchmaking system at the heart of its online play. For these games, customers cannot play the game at all without matchmaking.
2. **Hierarchy**. Rating player skills allows ranking ladders and tournaments that ultimately motivate competitive players. A rating system also has the mechanics necessary for progression or fall along these ladders.
3. **Experience**. A skill rating system dictates the experience of competitive play, determining how a player "feels" when playing ranked games. Players may grow frustrated if their skill rating converges too slowly or if they feel the system updates unfairly. A system that provides a negative experience may cause players to stop participating in ranked play entirely. This is a more subtle purpose unique to commercial games while seldom present in chess or similar games.

However, online games pose unique problems for the design of rating systems. Each game needs to account for possible technical issues, such as network disconnects. There must also exist a policy of whether to allow or disable players joining or leaving a game in session. These issues raise questions over how the rating system responds. For example, *League of Legends*, a five-player team game, does not tolerate dropouts and disallows new players from joining the game. In ranked play, a player that leaves the game for five minutes

[3] Herbrich, Ralf, Tom Minka, and Thore Graepel. "Trueskill™: A Bayesian skill rating system." Advances in Neural Information Processing Systems. 2006.

receives an automatic loss, even if his team wins. However, the game does not adjust rating updates in response to uneven teams.[4]

Perhaps the most significant challenge to skill rating design lies with online team games. Normally, adapting a well known rating system to a team game like basketball or football seems fairly trivial. However, most online games pose the unusual situation where the game must rate the skill level of an individual player in a team game. With exception to professional tournament leagues, most online team games assemble teams anonymously. Each player gets placed in a team full of strangers based on his personal skill rating. This presents a major challenge to designing the skill rating system, which can lead to grave error and frustrating experiences in the ranking ladder.

This project seeks to tackle these design challenges and research possible methods to creating a more accurate rating system that facilitate more positive experiences in competitive play for online team games.

But how do existing rating systems measure in this context?

The Elo rating system still sees use today in some form or another, despite being developed by statistician Arpad Elo more than 50 years ago. Originally designed for chess, the system models the performance of each player as a random variable in a Gaussian distribution. While simple to implement, Elo has numerous issues that make it inappropriate for many online games. When reviewing the various rating systems used across electronic and tabletop games, social software expert Christopher Allen listed three characteristics of Elo that make it unfavorable for the use in online games.

[4] For a variety of reasons, Riot Games deliberately choose not to adjust skill rating updates if a teammate leaves the game. If a losing team receives a lesser penalty to their ratings, this may cause losing players to peer pressure their teammates into intentionally leaving the game to mitigate their rating loss.

1. **New player rankings converge slowly**. Once a new player enters the rating system, they must play a considerable amount of games before arriving at their true rating. This leads to frustrating experiences and makes matchmaking unstable for non-veteran players.
2. **Elo does not support multiplayer**. Made for a two-person game, Elo does not account for additional players. While many game developers modify Elo for multiple players, Elo's shortcomings and quirks tend to amplify with the addition of more players. Disparities in skill rating complicate matchmaking. In addition, Elo punishes higher rating players for playing with lower rating players. Since matchmaking is automated in most games, this can make players feel like the system is unfair and punishing them for something beyond their control.
3. **Elo does not have rating decay**. Player skill typically deteriorates if the player has long periods of inactivity. The Elo system does not account for this, assuming that a player will have the same skill level.
4. **Elo has no measure for a skill rating's reliability**. The accuracy of any skill rating may vary wildly depending on factors such as how many games a player has played or how long since the last time they played a game. For the sake of simplifying the system, Arpad Elo choose to have the system assume each player has the same standard deviation in their performance.

The lack of rating decay and measurement of reliability in Elo led Mark Glickman to develop the Glicko rating system. Glicko adds a standard deviation to the model called RD that measures the reliability of a player's rating. A revision of Glicko, called Glicko-2, also adds an additional variable called violality, which adjusts the rating decay.

In Glicko, RD changes as a player competes or with the passage of time. A high RD indicates a new player to the system or a player that competes infrequently. A low RD player competes frequently and therefore has a highly reliable rating. RD also plays a role in the update algorithm. A player's rating adjustments depend not only on his opponent's rating, but also his opponent's deviation where a more confident opponent rating yields a higher adjustment.[5]

While it solves some of the issues of Elo, Glicko still lacks the capability of measuring players individually in a team game. Thankfully, Microsoft Research developed a significant extension to Glicko with the explicit goal of adapting the system for multiplayer games: TrueSkill.

TrueSkill was developed by a Microsoft research team consisting of Ralf Herbrich, Tom Minka, and Thore Graepel with the intent of deploying the rating system in Microsoft's XBox Live online gaming service. The team specifically created TrueSkill to address the issue of rating individual players in team games and game outcomes treated as a permutation of teams or players rather than merely a winner and a loser.[6]

The Microsoft Research team tested TrueSkill against Elo in four types of games: 8 players against one another, 4-versus-4 team games, 8-versus-8 team games, and a 1-versus-1 games. For most game types, TrueSkill proved to result in lower prediction errors, especially for games consisting of large teams. However, games with small teams showed significantly high prediction errors. For a full run with 4-person team games, Elo surprisingly

[5] Glickman provides more details of the Glicko rating system at his website <http://www.glicko.net/glicko.html>.
[6] Herbrich, Ralf, Tom Minka, and Thore Graepel. "Trueskill™: A Bayesian skill rating system." Advances in Neural Information Processing Systems. 2006.

performed better than TrueSkill.[7] Even though TrueSkill generally showed greater accuracy than Elo in a challenged set of games with closely matched teams, TrueSkill performed weakest when predicting and updating games with small team sizes.

Despite their work with TrueSkill, Microsoft Research has shown the need for further research into developing rating systems for online games with small team sizes. This will be the focus of this paper.

I propose changes to rating system update algorithms to better facilitate online multiplayer games with small team sizes. In order to test the proposed changes, I developed an automated tournament simulation and performed experiments utilizing the proposed update algorithms. In this paper, I present the details of these experiments, an analysis of the results, and an evaluation of whether these methods improve rating players in an online game with small team sizes.

II. Proposed Update Algorithms

2.1 Overview

The following section details three proposed changes to the update algorithm in rating systems. These “update methods” were developed for the Glicko-2 rating system. However, the algorithms can apply to extensions of Glicko or other similar rating systems.

2.2 Individual Update Method

The Individual Update Method takes the naive approach to updating players after a game. This method treats each match outcome as a set of wins or losses, one for each member of

[7] TrueSkill did perform better than Elo in 4-versus-4 games in a “challenged” set of games with closely matched teams. However, even in this case, TrueSkill had a high average error of 37.17%, contrasting with the 29.94% error for 8-versus-8 games.

the opposing team. This method takes advantage of rating systems like Elo or Glicko by abstracting a team match as a set of 1-versus-1 matches.

For example, consider a game where Alice participates on a team versus an opposing team consisting of Bob, Carol, Dan, and Eve. If Alice's team wins, the game updates Alice's rating by considering this game as four wins: one against Bob, one against Carol, one against Dan, and one against Eve.

One of the potential issues with Individual Update stems from the sensitivity of Glicko-2's tau parameter. The tau parameter influences how much a skill rating changes with each outcome. Smaller values of tau are preferred in a game with matches having improbable results where most opponents are evenly matched or where skill has a greater weight in a game's outcome. Since Individual Update treats each match as five matches for each player, a player can experience enormous changes in their rating. Small scale tests supported my suspicion despite using values of tau as low as 0.3.

Algorithm 1 displays the pseudocode for Individual Update Method. The procedure and other update methods take the ratings $\mu_{0..n}$ and deviations $\Phi_{0..n}$ of the participating n players and the outcome $s_{0..t}$ of each of the t teams.

Algorithm 1 Individual Update Method

```
1: procedure IndividualUpdate(μ_{0..n}, φ_{0..n}, s_{0..t})
2:     for all teams a do
3:         b = index of the opposing team
4:         for all players i in team a do
5:             for all opponents j in team b do
6:                 GlickoUpdate(μ_i, φ_i, μ_j, φ_j, s_a)
7:             end for
8:         end for
9:     end for
10: end procedure
```

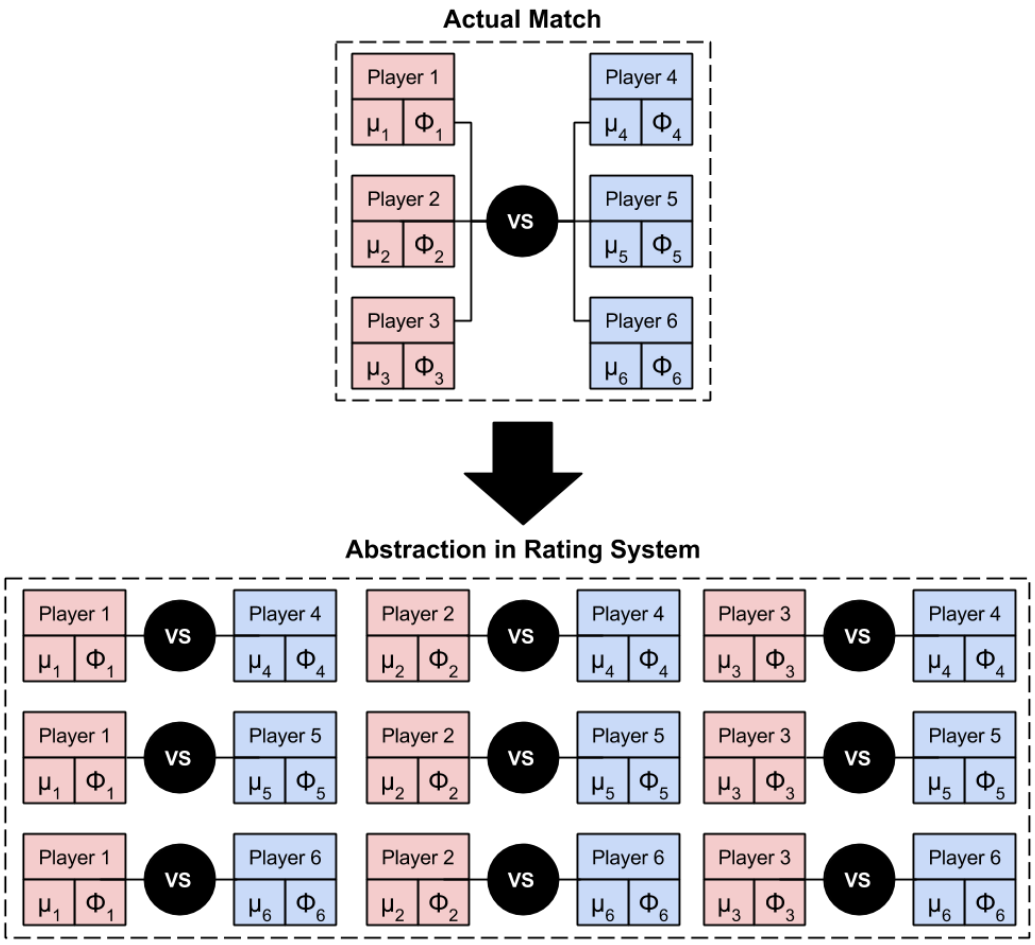

Figure 2.1: Model of Individual Update Method

2.3 Composite Opponent Update Method

The Composite Opponent Update Method considers each outcome as a match against a single player possessing the average rating and deviation of the opposing team players. In effect, the update method creates a composite player out of the opposing team and uses this player's resulting rating and deviation when updating a player.

For example, if Alice competes against a team consisting of Bob, Carol, Dan, and Eve, the game will update Alice's rating by considering the outcome as a 1-versus-1 match against a composite player having the average rating and deviation of Bob, Carol, Dan, and Eve. The Pseudocode for this method is displayed in Algorithm 2.

Algorithm 2 Composite Opponent Update Method

procedure COMPOSITEOPPONENTUPDATE($\mu_{0..n}, \phi_{0..n}, s_{0..t}$)
  for all teams a **do**
    $b =$ index of the opposing team
    $\mu^c = \sum_{i=0}^{n} \frac{\mu_i}{n}$ //Rating of composite opponent
    $\phi^c = \sum_{i=0}^{n} \frac{\phi_i}{n}$ //Deviation of composite opponent
    for all players i of team a **do**
      GLICKOUPDATE($\mu_i, \phi_i, \mu^c, \phi^c, s_a$)
    end for
  end for
end procedure

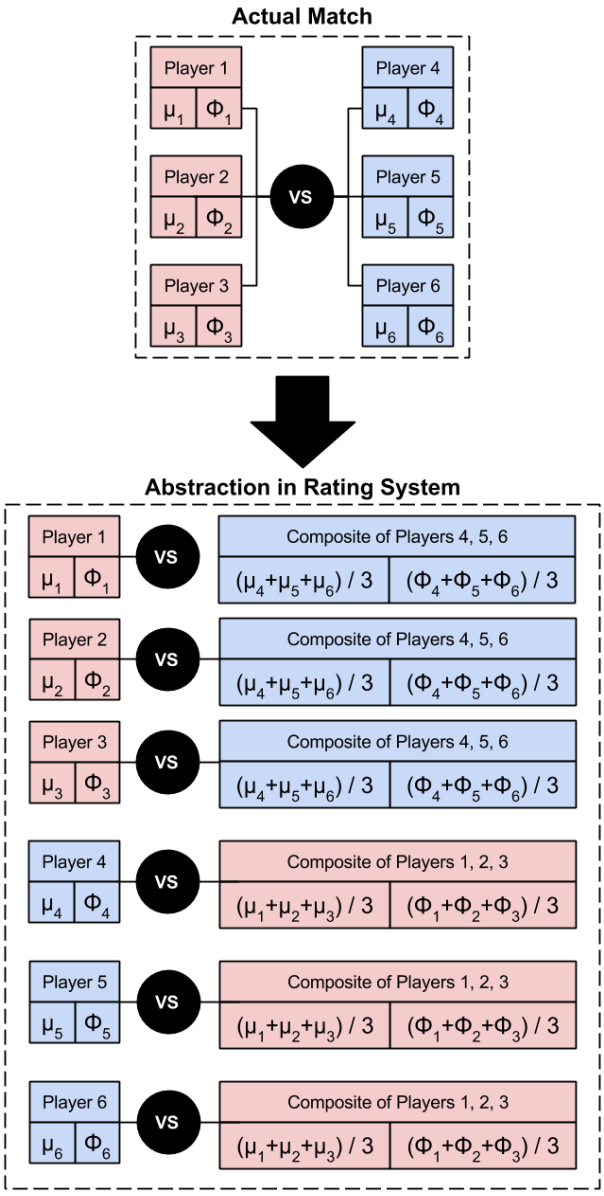

Figure 2.2: Model of Composite Opponent Update

2.4 Composite Team Update Method

Composite Team Update Method uses the average statistics of a player's team when updating the player's rating. This method works similarly to Composite Opponent except the

update algorithm substitutes a player's statistics for a composite of their team and uses the resulting rating and deviation delta to adjust the player's actual skill rating.

For example, the game will not use Alice's raw statistics when updating her rating against her opponents. Instead, it will take the average of her team's ratings and deviations and the average of her opponents. The rating system computes a delta value by matching these averages and uses this delta value when updating Alice's ratings.

This method attempts to mitigate "Boosting," a situation where a player will receive a dramatic change to their rating as a result of participating in a game with higher rated players. Since a player's rating increase or decrease depends on team averages, an unstable distribution of skilled players will have less of an effect.

As shown in the pseudocode below in Algorithm 3, Composite Team Update Method requires altering of Glicko-2's algorithm.[8]

Algorithm 3 Composite Team Update Method

procedure CompositeTeamUpdate($\mu_{0..n}, \phi_{0..n}, s_{0..t}$)
  for all teams a **do**
    $\mu_a^c = \sum_{i=0}^{n} \frac{\mu_i}{n}$ //Rating of composite team
    $\phi_a^c = \sum_{i=0}^{n} \frac{\phi_i}{n}$ //Deviation of composite team
  end for
  for all teams a **do**
    $b =$ the index of the opposing team
    $\Delta = v g_{\phi_b^c}(s_a - E_{\mu_a^c, \mu_b^c, \phi_b^c})$ //Glicko2
    for all players i of team a **do**
      $\sigma_i' = \text{Violatility}(\Delta, \sigma_i, \phi_i, v)$ //Glicko2
      $\phi_i^* = \sqrt{\phi_i^2 + \sigma_i'^2}$ //Glicko2
      $\phi_i' = 1/\sqrt{\frac{1}{\phi_{\phi_i}^{*2}} + \frac{1}{v}}$ //Glicko2
      $\mu_i' = \mu_i + \phi_i' g_{\phi_b^c}(s_a - E_{\mu_a^c, \mu_b^c, \phi_b^c})$ //Glicko2
    end for
  end for
end procedure

[8] See Glickman, Mark E. "Example of the Glicko-2 system." *Boston University*(2012).

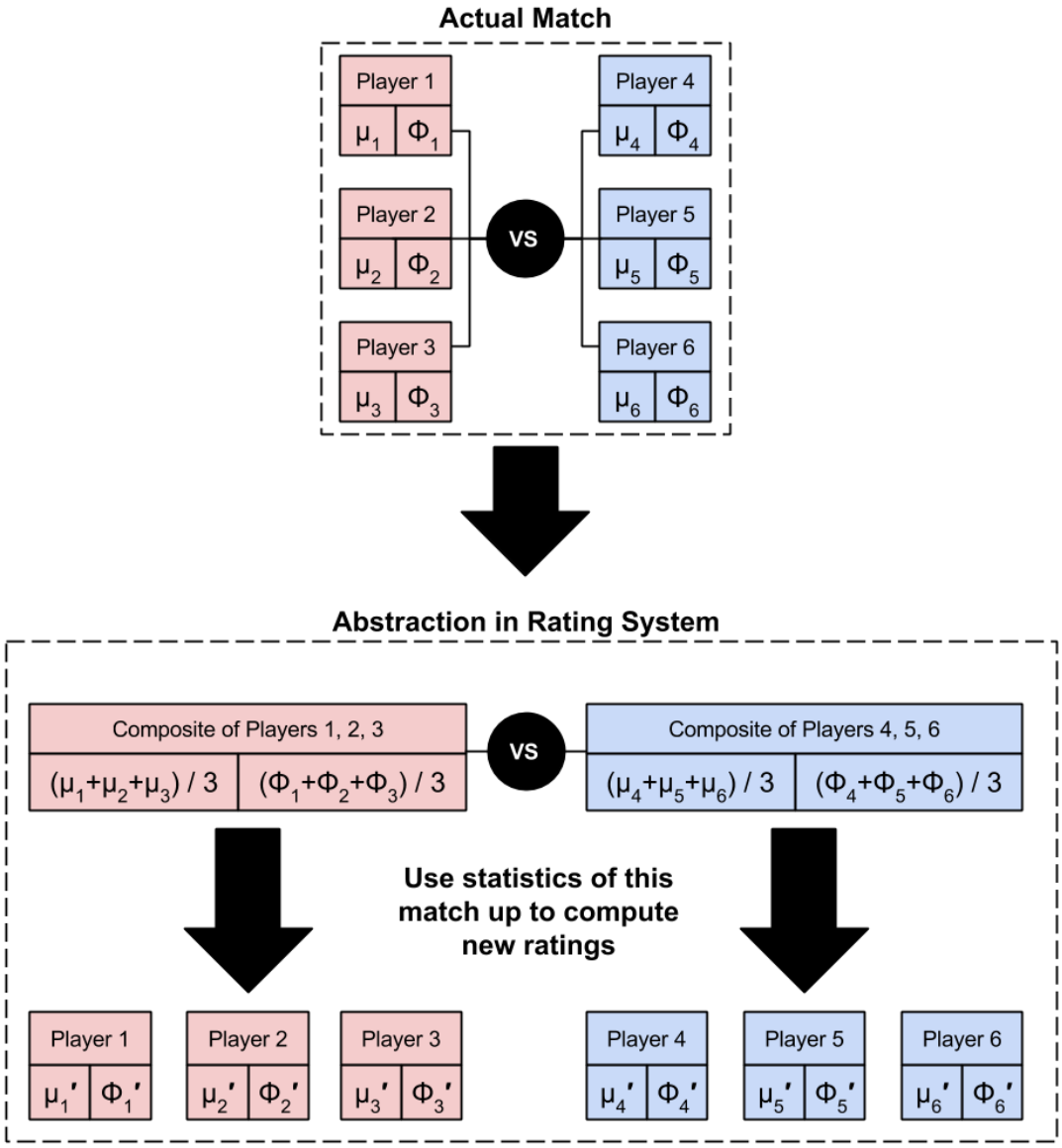

Figure 2.3: Model of Composite Team Update Method

III. Experiment

3.1 Overview

I developed a testing program in order to evaluate the proposed update methods and further research rating system improvements in online team games.

The testing program utilizes SoccerBots and SBTournament, a robot simulation environment created by Tucker Balch and enhanced by Group for Artificial Intelligence Applications.[9] SBTournament runs regulation RoboCup matches, five-player soccer matches played by robots. Though originally used as an artificial intelligence teaching tool, SBTournament boasts its use in a variety of research projects, including a project to create an automated coach using clustering algorithms.[10]

3.2 Assumptions

The experiment makes the following assumptions about the context of the game. These assumptions mimic the common traits of online games with small team sizes.

1. **Five versus five team game.** Two, five-person teams compete against one another.
2. **Players rated individually**. Each player has their own skill rating and other variables that the game uses for matchmaking.
3. **Anonymous, autonomous matchmaking**. The game performs the matchmaking and assembles the teams. This test assumes players cannot choose or assemble their own teams.

[9] SBTournament is available for download at <http://gaia.fdi.ucm.es/research/sbtournament>

[10] See Jiménez-Dıaz, Guillermo, et al. "Predicting performance in team games." II for systems, C. Technologies of Information, and Communication, editors, ICAART(2011): 401-406.

4. **Players participate in their preferred role**. Similar to games such as soccer and American football, many online team games have roles or positions fulfilled by the players. This experiment assumes a player will participate in their preferred role.

3.3 Procedure

The testing program runs 100 RoboCup soccer games with a pool of 25 players. Each match lasts 4 minutes to ensure a decisive outcome. With the assistance of SBTournament's logging features, the program logs the activity of each experiment run, records matchups and results, and tracks the rating changes for each player. In the experiment, I performed and recorded data of four quality 100-game runs for each update method.

3.4 Players

The testing program draws its player pool from pre-packaged descriptions in SBTournament, each created by a Complutense University AI student. Normally, SBTournament uses team description files written as Java classes to define each team and the strategies employed by each of the five players. However, the testing program disassociates these players from their original teams in order to perform matchmaking on an individual player basis.

For this experiment, I choose five team descriptions to generate a pool of 25 players. These teams were chosen based on their performance in SBTournament runs and whether or not they possess a similar team composition as other teams. This ensures each team assembled in matchmaking will have similar roles. For example, each team should have a goal keeper.

3.5 Matchmaking

The test program attempts to create a fair match around a player with the least amount of games played. It chooses a "pivot player" with the least amount of games played and

computes a prediction between this player and each of 20 players with the closest rating. From the top predictions, the program greedily alternates between assigning players to the pivot player's team and the opposing team.

For the experiment, this matchmaking algorithm remained consistent for each update method in order to better evaluate the methods individually. I did not wish to taint the resulting data by having matchmaking differ for each update method.

3.6 Prediction & Error

The test program predicts the outcome of a match by averaging the ratings and deviation of each team and performing the rating system's (Glicko-2) prediction function. After the match, the rating system's error function is called using this prediction. Like with matchmaking, I choose to keep this consistent across testing each update method to prevent having a different prediction algorithm adversely change the data.

IV. Results

4.1 Criteria

I used several criteria in order to ascertain the quality of each update method based on methods utilized by Microsoft Research team as well as methods mentioned in Joshua Menke, Shane Reese, and Tony Martinez's paper on estimating individual ratings.[11] Specifically, I looked at each method's predictive performance, how each player's ranking measures against their win percentage, and how quickly a top player's skill rating converges to a stable value.

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

4.2 Predictive Performance

The first metric involves evaluating the effects of each update method on predictive performance. The table below displays the average prediction error across all matches using each update method. A lower error percentage indicates the rating system predicting the outcome of a match with greater accuracy using a particular update method. The table below displays the average error for all games using that update method (“full”) and the average error for the last 20 games (“top 20”).

	Individual	Composite Opponent	Composite Team
Full	21.46%	22.11%	22.23%
Top 20	21.71%	22.96%	22.08%

Out of the three, Individual Update Method displayed the best influence over predictive performance, having the lowest average error. Composite Opponent had the highest error, but performed better than Composite Team across all matches.

However, I did not look at average error alone. The stability and convergence of the predictive performance is also important. A favorable rating system has a predictive performance that improves quickly across matches without severe high spikes. The chart below demonstrates how the prediction error trends over the matches.

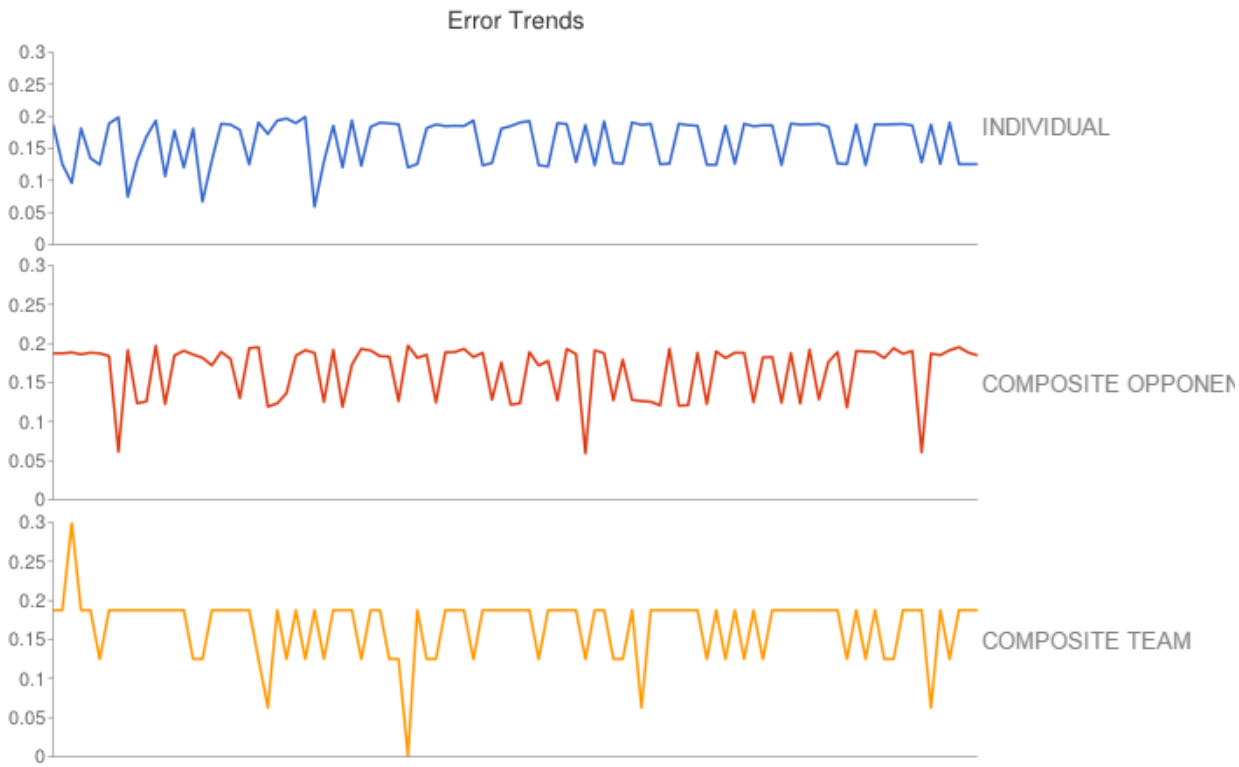

Figure 4.1: Error Trends

The chart shows that the Composite Opponent Method converges slightly faster in the experiment than the Individual Update Method. However, Individual displays the best stability with a smaller deviation. Composite Team Method has the highest deviation. Note that the outliers and spikes descending downward indicate cases where match resulted in a tie. Such spikes do not have much bearing on evaluation of the update method due to the fact that the underlying rating system (Glicko-2) has strict definitions of match outcomes. The rating system considers each outcome as either a win/loss or a tie.

4.3 Ranking vs. Win Percentage

This metric involves comparing each player's ranking versus their win percentage. A player with a high win percentage should likely have a high skill rating. This isn't always the case in

practice. A number of factors may cause a high skill player to have a lower win rate or a low skill player to have a higher win rate. Changes in actual skill and recent win/loss streaks can influence that. However, the RoboCup players inherently lack some of those factors. For this and other reasons, a comparison between a player's ranking against their win percentage provides insight to the effectiveness of a rating system.

The graphs below compare each player's skill ranking with their win percentage based on results averaged across the experiment runs for each update method. Each bar represents one of the players and their win percentages. The graphs arrange the bars according to ranking. The left-most bar represents the player with the lowest skill rating whereas the right-most bar represents the highest rated player.

Since one expects the highest rated player will also have the highest win rate, I can anticipate the right-most bar will have the highest win percentage. Though I can expect a few outliers, each graph should display an upward slope.

Individual Update Method does display an upward slope. However, there exists a few outliers. Particularly, the player with the lowest win rate has a middling ranking due to many of their games resulting in ties.

Composite Opponent Update Method displays the most expected behavior. By contrast, Composite Team has a very irregular curve where the players with the highest win rate are not ranked among the top five.

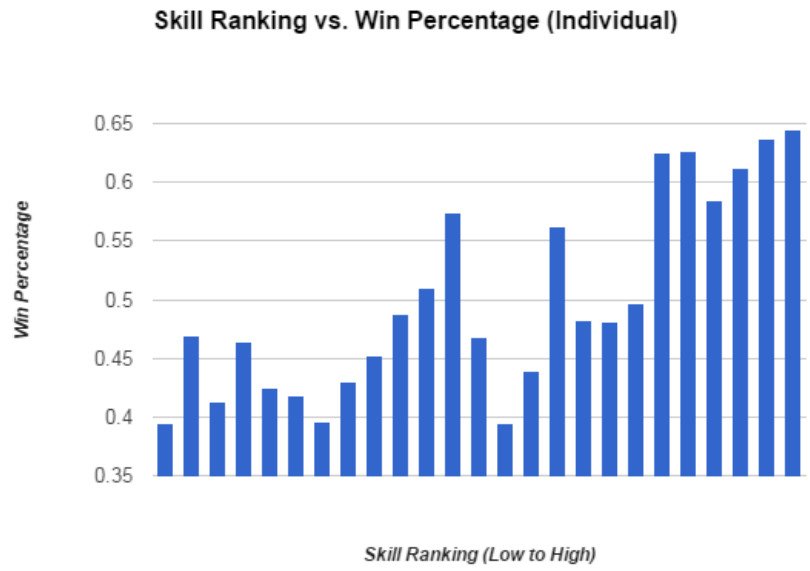

Figure 4.2: Skill Ranking vs. Win Percentage (Individual)

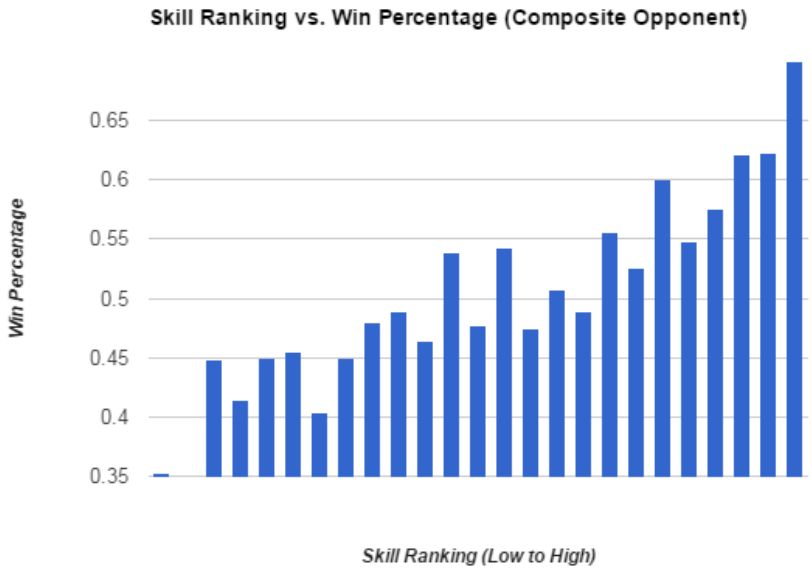

Figure 4.3: Skill Ranking vs. Win Percentage (Composite Opponent)

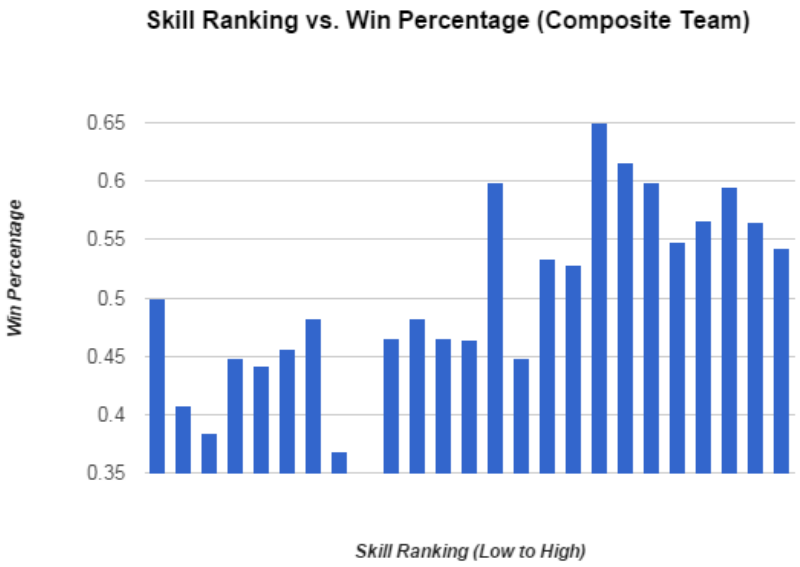

Figure 4.4: Skill Ranking vs. Win Percentage (Composite Team)

4.4 Convergence

Rating convergence demonstrates how quickly a new player's rating will rise and stabilize before reaching the neighborhood of their true skill. I looked at the ratings of the average top player from runs with each update method. The recorded data displays the progression of each player's rating as they competed. The chart below displays the rating progression of the top player from each update method.

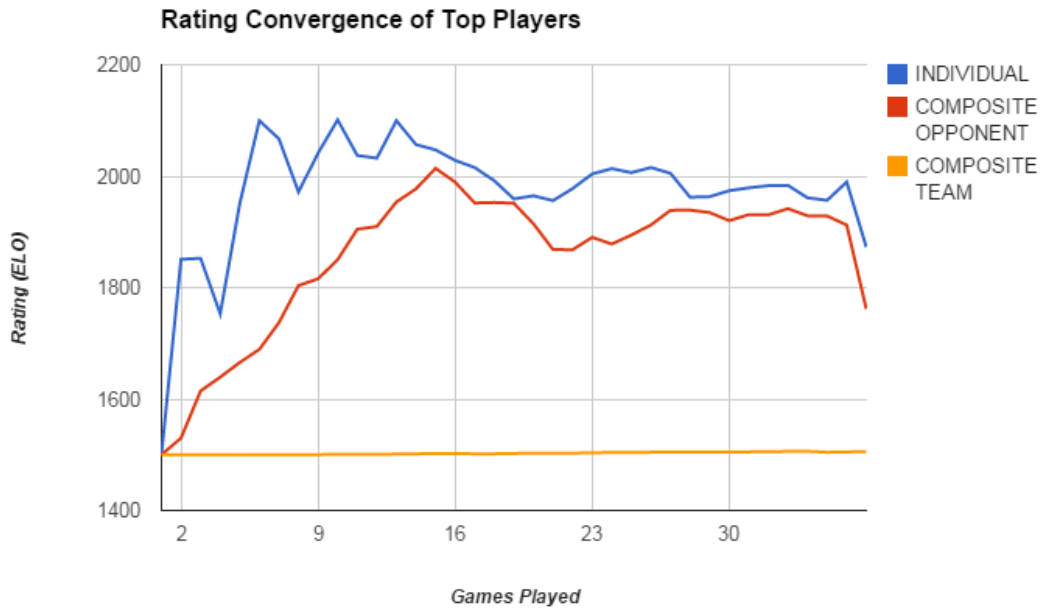

Figure 4.5: Rating Convergence of Top Players

The chart shows that the top player from Individual Update Method raised rather quickly in rating before falling off and stabilizing. Composite Opponent Update Method displays a slower, but more stable convergence. The skill rating of both these players converge between 16 to 23 games. However, the top player of the Composite Team Update Method receives little change to its rating throughout the games.

V. Analysis

The experiment proved both Individual and Composite Opponent as viable methods of adapting a Glicko-based rating system towards team games that require individually rated players. However, the Composite Team Update Method displayed unsatisfactory results. In addition to having the least favorable predictive performance, the method resulted in abnormal rating convergence and results where a player's win rate does not adequately correlate to a higher ranking.

I conclude that Composite Opponent resulted in the most favorable rating system behavior out of the three update methods. While Individual Update Method had a superior predictive performance, Composite Opponent displayed more satisfactory convergence behavior and the appropriate kind of correlation between win rate and ranking I expect from a rating system.

Additionally, Composite Opponent had more stable results in the experiment. Individual Update Method had a more irregular win rate graph and a more erratic rating convergence curve. This aligns with my suspicions that Individual Update Method would possess greater sensitivity to Glicko's tau parameter, which determines the weight each match outcome has on updating player ratings. Even with a reasonably low value of tau, Individual Update Method displayed less stability than Composite Opponent Update Method. As a result, the experiment demonstrated Composite Opponent as causing a more desired behavior in the rating system.

VI. Conclusion

The Composite Opponent Update Method displayed the most favorable influence over a Glicko-based rating system out of the three update methods proposed and tested. The experiment using the modified SBTournament system demonstrated promise in abstracting a small team game match as a series of 1-on-1 matches against a player with the opposing team's average rating and deviation. These findings open the door to further research in expanding this change to a rating system's update algorithm. Instead of merely taking the average, more methods of creating a composite player from the opposing team could yield better results.

The research I performed furthers the cause of improving rating systems for small team games, especially within the context of online computer games that rely on competitive

play for commercial viability. With this, the industry takes one further step towards improving where current rating systems fall short.

Bibliography

1. Allen, Christopher, and Shannon Appelcline. "Collective Choice: Competitive Ranking Systems." Life with Alacrity. N.p., 3 Jan 2006. Web. 9 Jan 2014. <http://www.lifewithalacrity.com/2006/01/ranking_systems.html>.
2. Glickman, Mark E. "Example of the Glicko-2 system." *Boston University*(2012).
3. Glickman, Mark E. "The glicko system." *Boston University* (1995).
4. Graepel, Thore, and Tom Minka. "TrueSkill Ranking System." Microsoft Research. Microsoft. Web. 9 Jan 2014. <http://research.microsoft.com/en-us/projects/trueskill/>
5. Hacker, Severin, and Luis Von Ahn. "Matchin: eliciting user preferences with an online game." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2009.
6. Herbrich, Ralf, Tom Minka, and Thore Graepel. "Trueskill™: A Bayesian skill rating system." Advances in Neural Information Processing Systems. 2006.
7. Jiménez-Dıaz, Guillermo, et al. "Predicting performance in team games." II for systems, C. Technologies of Information, and Communication, editors, ICAART(2011): 401-406.
8. Menke, Joshua E., C. Shane Reese, and Tony R. Martinez. "Hierarchical models for estimating individual ratings from group competitions." *American Statistical Association*. 2007.
9. Nikolenko, SERGEY I., and ALEXANDER V. Sirotkin. "Extensions of the TrueSkill™ rating system." Proceedings of the 9th International Conference on Applications of Fuzzy Systems and Soft Computing. 2010.

10. "Top 100 Largest Overall Prize Pools :: E-Sports Earnings." *Top 100 Largest Overall Prize Pools :: E-Sports Earnings*. Web. 5 Feb. 2015. <http://www.esportsearnings.com/tournaments>.

11. "Worlds 2014 by the Numbers." *Riot Games*. 1 Dec. 2014. Web. 5 Feb. 2015. <http://www.riotgames.com/articles/20141201/1628/worlds-2014-numbers>.