Application Context Based Algorithm for Player Skill Evaluation in MOBA games

Mirko Suznjevic, Maja Matijasevic
University of Zagreb
Faculty of Electrical Engineering and Computing
Zagreb, Croatia
{mirko.suznjevic, maja.matijasevic}@fer.hr

Jelena Konfic
Ericsson Nikola Tesla
Zagreb, Croatia
jelena.konfic@ericsson.com

Abstract — **This work is motivated by the challenge to improve player rating systems in Multiplayer Online Battle Arena (MOBA) games by incorporating the application context, as opposed to calculating the player rating based solely on the match outcome (winning or losing). The application context is modeled by using a set of parameters describing the players' in-game accomplishments and contributions to the team, as well as the performance of the team as a whole. The proposed application context based rating algorithm functions as an extension of the underlying (unmodified) player rating algorithm by adjusting the rating update step and can thus be applied for different rating systems in MOBA games. We evaluate the proposed algorithm by using a dataset comprising over 400,000 matches of the MOBA game Heroes of Newerth.**

Keywords— rating system; matchmaking; Elo system; MOBA; video game; Heroes of Newerth

I. INTRODUCTION

Multiplayer Online Battle Arena (MOBA) is a genre of video games in which a player controls a single character in one of two competing teams with a goal of destroying the opposing team's base. MOBAs attract millions of users and have massive earnings – in January 2015, their profit was estimated at 54 million US dollars [1]. Like in other multi-player games, a *rating system* in a MOBA is used to assess an individual player's skill, and a *matchmaking system* is used to form teams of players with similar levels of skills. In matchmaking process, it is important that players' skill levels (ideally, corresponding to player ratings) are well-balanced, i.e., that their chances of winning are fairly even, which makes the game more interesting for the less experienced, as well as more experienced players [2]. It may be noted that players can also form teams by themselves, without involvement of the matchmaking system and regardless of the player ratings and balance – such cases are not investigated in this paper. In multiplayer team games, once the teams are formed, each player contributes to the team according to his or her individual skills. Thus, rating systems play three important roles: 1) continuously assess a player's skill level, 2) enable players to determine their own standing in the game, and compare it with that of other players, and 3) use player ratings to provide input to the matchmaking system.

In many current rating systems for multiplayer team games, the change in a given player's rating is calculated based solely on the match outcome. We consider such systems inherently "unfair", in that they do not adequately distinguish, or give credit to, the individual player's in-game performance. In other words, if one of the team players has performed well in a match, while the other player on the same team has performed poorly, both players will receive (or lose) the same number of rating points for winning (or losing) that match (assuming that they entered the match with similar ratings – more about this later). Not only can this be very frustrating – especially for good players, but there is another, less obvious, but more crucial problem: that such an approach can, over time, lead to undeservedly high(er) or low(er) player ratings. This is notable, because the weakening of the correspondence between the player's "actual" skill level and the player rating eventually undermines the fundamental ability of the rating system to correctly predict the outcome of the match, and thus the ability to assure the desired initial balance, or "even chances of winning", when forming teams.

Our approach aims to correct this by proposing an algorithm which analyses the player's performance in a particular match and modifies the number of rating points won or lost based on relating that performance to the historical information about the performance of all players. We also provide a case study based on Heroes of Newerth (HoN) MOBA, developed by S2 Games. We have used publicly available information on the rating system in HoN, rules, and gameplay mechanics, and we have collected a dataset comprising detailed results of over 400,000 matches by using an Application Program Interface (API) provided by the game developers. This dataset has been used for the evaluation of our algorithm and its comparison with the existing rating algorithm in HoN. Our initial findings show that our algorithm achieves up to 10% better accuracy of prediction then the current HoN rating system. Since player's rating is used when forming teams, this improvement has the potential to implicitly improve the matchmaking system as well.

The paper is organized as follows: Section II summarizes the related work, followed by the description of the basic features of rating system in HoN in Section III. In Section IV we describe the proposed algorithm, named ACARI. The evaluation methodology and results are presented in Sections V and VI, respectively, and Section VII concludes the paper.

II. Related work

Many research papers have addressed rating systems for games since Elo system was invented for the purpose of rating chess players, in 1959 [3]. Elo system is based on the Bradley-Terry model for pairwise comparison [4]. The main assumption in Elo is that each player has current strength, which is unknown and estimated based on his *rating* – the mean value of his performance in the match. In 1999, Mark E. Glickman invented the Glicko system [5], which incorporates a rating period into reliability assessment of one's rating. To calculate a player's rating, Glicko uses a "rating deviation", which represents uncertainty in the player's current skill. Rating deviation is in fact standard deviation, which grows linearly with the time the player hasn't played the game.

Elo and Glicko are not suitable for multiplayer games. To calculate rating for multiple players and multiple teams in a match, Microsoft Research developed TrueSkill [6]. In TrueSkill, a player's skill is represented as a Gaussian-distributed variable. TrueSkill ratings also incorporate ties. Various extensions to TrueSkill have been proposed, such as: solving the problem of variable team size by selecting a different function for the team performance [7]; to infer entire time-series of skills of players by smoothing through time instead of filtering [8]; and through the incorporation of subgroup ratings into a team's overall rating [9].

Several notable works have incorporated information, other than match outcome, into the rating process. A context aware algorithm has been proposed by Zhang et al. [10]. They develop a factor model, in which individual skills are modeled by the inner product of an agent factor vector and a context factor vector. They generalize the concept of contexts and combine it with the TrueSkill algorithm to create a simple extension, called TrueSkill-ext. This approach is tested on Halo 2 and Dota – a custom scenario for Warcraft 3, which was the base for creation of the whole MOBA genre. As "context" they identify specific maps for Halo 2, and heroes for Dota. Context about players' characteristics (e.g., good machine gunner and poor sniper), coupled with statistics from previous players' matches, was used in a neural network approach to matchmaking and rating presented by Delalleau et al. [11], who assert that their basic algorithm outperforms rating systems, such as TrueSkill. They modify the matchmaking system to maximize the match quality based on the developed neural network and analyze the results on a dataset from the game Ghost Recon. Role based approach to matchmaking in MOBAs was proposed by Myślak and Deja [17], but they could not extract the Elo ratings so an estimate was used for validation (to guess the role they used the position of the player on the map). Latency between gaming hosts as context information was used in Htrae [12] and Switchboard [13]. Jiménez-Dıaz et al. use machine learning approach to identify combinations of player behaviors which yield the best results to create better teams [14]. Di Fatta et al. evaluate the skill of a chess player not by the outcome of the game, but by the moves made by the player, using the Bayesian inference method [15].

Our approach is comparable to Switchboard approach [13], in that it can be applied to any rating algorithm. Also, our approach is similar, and partially based on the context approach described in [10], but with several important differences: 1) we modify only the rating update step of the algorithm, not the algorithm itself; 2) our proposed modification is performed on the rating adjustment (i.e., increase or decrease of player's rating after a match), which is calculated in the final step of the rating process – hence, our algorithm can be used with any existing rating algorithm; and 3) we add two levels of context: one at the overall team level, to capture the synergy of the team play, and the other at the individual player level, to capture the differences in performance between players. Finally, we present a concrete example, using HoN as a use case, and evaluate the algorithm using a very large, recently gathered dataset with very closely estimated Elo ratings, as opposed to [17].

III. Rating system of Heroes of Newereth

To the best of our knowledge, HoN rating system uses a modified version of Elo. A player's rating in HoN is termed *Match Making Rating* (MMR), because this rating is used in the matchmaking process. To the best of our knowledge, MMR is calculated based solely on the match outcome. Exact rules and formulae for rating and matchmaking are proprietary and so far not made public so it is not possible to say whether additional enhancements such as avoidance of repeated opponents of the team-based Swiss Ladder system are implemented. Nevertheless, it can be assumed that due to the sheer number of player in MOBAs (from tens of thousands to millions) such systems are not needed. The description that follows is based on our own experience, observation, or publicly available information[1].

The value of MMR for a newly registered player is initially set to 1500. When forming teams, the rating system tries to bring together players with similar ratings so that the differences between average MMRs of the opposing teams, as well as the individual MMRs of the players comprising the team, are relatively small. After each match, depending on the outcome, a player typically gets 5 points added to MMR if his or her team won, or, 5 points subtracted from MMR if his or her team lost the match. Number of points won or lost may vary, depending on the difference between average MMRs of the opposing teams and the difference between MMRs of individual players in the team. The maximum number of points one can get or lose is 10. The players who leave the match always lose 5 points and their teammates lose 3 points, regardless of their MMRs.

There are also a few MMR control mechanisms that serve special purposes, like the mechanism that aims to quickly place a new player into an appropriate skill rank. Each new player is initially put into a so called "placement phase", in which one's rating can change faster than usual so the player can quickly reach the MMR which roughly reflects his or her skill level. Also, the rating system monitors all players' scores, detects inconsistencies in MMR (e.g., if a player, who is currently in the "placement phase", gets an unlikely high number of kills, like, 15 kills in a row) and reacts by quickly increasing the MMR in much larger steps (e.g., +80 MMR). This is an example of a context based adjustment of a rating step.

[1] http://forums.heroesofnewerth.com/showthread.php?562680-Matchmaking-System-Clarifications-and-Feedback

IV. The Proposed Application Context Aware rating algorithm - ACARI

We now define the fundamental terms related to the proposed **A**pplication **C**ontext **A**ware **R**ating algor**I**thm (ACARI, for short) in Table 1.

The primary premise of ACARI is that teams and players who perform significantly better (or worse) than average should be rewarded (punished) with adequate rating adjustment. We also note that every player in each game has a certain role that is correlated with the selected hero (one hero can assume one or more roles per match). In future work we aim to devise role identification techniques, similar to [17], as players can play heroes in a way in which developers did not envision. Players may prefer to play some roles over others, and may have different skill levels for different roles. Therefore, as a final premise we state that player's skill level (and therefore rating) for different roles should have different values. Hence, ACARI introduces two main extensions when compared to traditional rating systems:

- Points distribution based on player's in-game performance, including:
    - Performance of a team.
    - Performance of a single player.
- Different rating for each role a player has in a match.

ACARI takes into account different roles that a player can take during the match, and calculates MMR change separately for each role. Moreover, ACARI tracks player's total MMR, which consists of weighted average of all role MMRs. As an input, ACARI takes changes in MMR calculated by the existing (unmodified) rating algorithm, and adjusts them based on the performance of the opposing teams and performance of a player in regard to historical performance data.

Table 1 – Term definitions for ACARI

Term	**Definition**
Performance parameter	Application level information about player's or team's performance (e.g., number of killed heroes, experience gathered, gold earned, etc.)
Hero	Game character with a unique set of abilities. Each (human) player controls one hero.
Role	The function a specific hero performs within a team. A hero can perform multiple roles (e.g., support, ganker, etc.) in a match.
Modification factor	The factor indicating the performance level (player's or team's) with respect to a given performance parameter. It s a real number, which can range from -0.5 to 0.5.
Weight factor	Factor indicating how important a parameter is for performing a role. Weight factors can take the following values: 0 (not important), 0.5 (slightly important), and 1 (important).
Role MMR	MMR associated with a specific role
Team MMR	MMR associated with a specific team
Total MMR	MMR associated with a player – weighted average of all his or her *Role MMRs*
Rating adjustment	Update of one's MMR after one match

A. *Rating modification based on player's performance*

A player can take multiple roles during a MOBA match. We define a set of roles $R = \{r_1, r_2, \dots r_m\}$, where r_i denotes a role, and m is the number of roles defined in a game. Each hero h_j from the set of heroes $H = \{h_1, h_2, \dots h_n\}$, where n is the number of heroes in the game, can in some amount fulfill the role r_i. Therefore, each hero h_j is assigned a specific role vector $RV_j = [rv_1, rv_2, \dots rv_m]$ with m values. Each value rv_i represents the percentage in which hero h_j fulfils the role r_i. For describing the application level player performance, we define the vector $P = [p_1, p_2, \dots p_u]$, where u is the number of in-game performance parameters taken into account for a specific game. P is constructed for each player in the match. Each value in vector P represents the percentage of the player's contribution to the overall match score of his or her team for a given performance parameter p_k.

It should also be noted that each performance parameter has a different (relative) "importance" for each role. We define a weight factor matrix W, sized $u \times m$, where w_{ki} corresponds to the importance of parameter $p_k, k = 1 \dots u$, for performing the role $r_i, i = 1 \dots m$. Weight factors w_{ki} take three possible values: 0, 0.5, or 1. These values have been determined empirically, based on the practical knowledge of the game mechanics of several experienced HoN players who indicated the importance of each parameter. In future work, we plan to evaluate the sensitivity of ACARI with respect to different combinations of weight factors.

The algorithm is described next. First, we calculate the *score of parameter* SP_{ki} for each parameter p_k for each role r_i. Note that SP_{ki} is defined for a role, and not for a hero. Given a match outcome, if the player wins, SP_{ki} is calculated as:

$$SP_{ki} = 1 + c \tag{5}$$

If the player loses, SP_{ki} is calculated as:

$$SP_{ki} = (-1) \cdot (1 - c) = c - 1 \tag{6}$$

where c is the modification factor ($-0.5 \le c \le 0.5, c \in \Re$). In this way, rating adjustment can be increased or decreased by 50%, based on the performance. Modification factor c for role r_i is calculated by using a function of a given in-game performance parameter p_k. An example of a function to determine the value of c is shown in Figure 1.

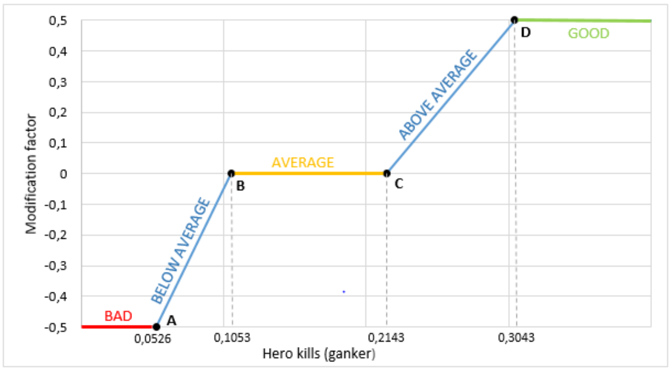

Figure 1 – Modification factor function (example)

Points A, B, C, and D split the line into five segments, denoting *bad*, *below average*, *average*, *above average*, and *good* player's performance with respect to the parameter p_k. Values of p_k at points A, B, C and D would typically be different for different roles. In the example in Figure 1, they have been calculated based on a statistical analysis of 10,000 matches. They represent: $\frac{1}{6}$ or 16.66 (point A), $\frac{2}{6}$ or 33.33 (point B), $\frac{4}{6}$ or 66.66 (point C), and $\frac{5}{6}$ or 83.33 (point D) percentile of the selected performance parameter distribution extracted from 10,000 matches.

Next, *score of role* (SRi) for the role r_i is calculated as:

$$SR_i = \sum_{k=1}^{u}(wf_{ki} \cdot SP_{ki}) \qquad (7)$$

where wf_{ki} is the weight factor from matrix W, and SP_{ki} is the score of parameter p_k for role r_i, and u is the total number of parameters. Considering the modification factor, the number of parameters and their weight factors, there is a maximum number of points SR_{max} for every role. To allow comparison, the number of points calculated using (7) needs to be scaled:

$$SR_{i\_final} = \frac{SR_i}{SR_{max}} \cdot C_{base} \qquad (8)$$

where SR_{max} is the maximum number of points for the role r_i, and C_{base} is the rating adjustment, i.e., a number of points given (or subtracted), obtained by the existing ("base") algorithm.

Rating adjustment for each player with the hero h_j is calculated by summing the scores of all roles, multiplied with the percentage in which hero h_j fulfils the role r_i. Since we cannot be certain which role player had in the game, all roles that the hero h_j fulfils are taken into account.

$$CP_j = \sum_{i=1}^{m} rv_{ji} \cdot SR_{i\_final} \qquad (9)$$

The idea of the algorithm for player rating up to this point is illustrated in Figure 2. External data in the figure denotes data defined by the existing mechanics in specific game.

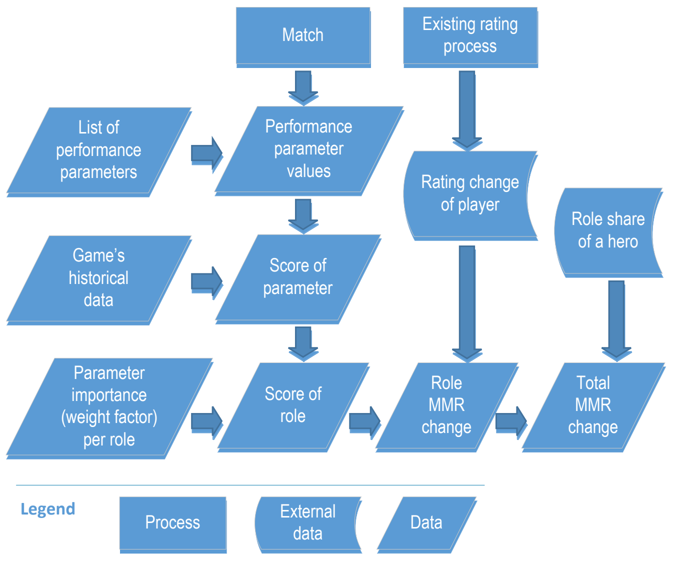

Figure 2 – Illustration of the ACARI's logic regarding player's performance rating

The result CP_j is scaled so that the sum of points players get based on their performance fits the number of points the whole team can get:

$$CP_{j\_final} = \frac{CT}{\sum_{i=1}^{v} CP_i} \cdot CP_j \qquad (10)$$

where CT is the rating adjustment of the whole team, and v is the number of players in the team. We explain how CT is calculated in the next section. ACARI tracks role MMRs and total MMR per player, so modified rating adjustments are in the end added to each of the defined MMRs.

B. *Rating modification based on team's performance*

For describing the application level team performance, we define the vector $TP = [tp_1, tp_2, \dots tp_z]$ where *z* is the number of application context team performance parameters (tp_i) taken into account when calculating team performance. Team performance is then calculated as:

$$\prod_{i=1}^{z} tp_i, tp_i \neq 0 \qquad (11)$$

Using the notation tp_{iw} to denote the i-th team performance parameter of the winning team, and tp_{il} to denote the corresponding i-th parameter of the losing team, the *performance difference* between the teams may be expressed as the ratio of their respective performances:

$$DP = \frac{\prod_{i=1}^{z} tp_{iw}}{\prod_{i=1}^{z} tp_{il}} \qquad (12)$$

Rating adjustment obtained from the existing algorithm (i.e., the sum of rating adjustments of all players in the team) C_{base} is modified to get the final rating adjustment for the team (CT) using the following expression:

$$CT = (1 + c) \cdot C_{base} \qquad (13)$$

Analogously to single player performance, the modification factor c is a function of DP, which is shaped like in Figure 1, where values A, B, C and D determine in which category the ratio of team's performances falls into. The higher the value of c, the more points will the winning (losing) team gain (lose).

V. EVALUATION METHODOLOGY

A. *Dataset gathering and filtering*

To evaluate ACARI, we collected real match data and players' rating data (Table 2), by using HoN Statistics API (https://api.heroesofnewerth.com), which provides users with statistical data about matches, players, heroes, and items in the game. As data can be retrieved by HTTP, we designed and implemented PHP scripts to collect it and save it to a database. Data collection process was made difficult by the limitation on the number of requests API could serve, presence of erroneous information, and lack of some needed information. Finally, only the statistical match data, e.g., number of heroes killed per each player, which player was playing with which hero, information who won and who lost, etc., and history of played matches per player were retrieved by using the mentioned API.

History of played matches was used for filtering of match data, so as to retain only the ranked matches (and exclude public and "easy mode" matches in which the skill is not tracked, for beginners or people who want shorter games).

Table 2 – Statistics of the retrieved data

No. of retrieved matches – dataset 1 (D1)	1,101,299
No. of valid matches – dataset 2 (D2)	1,101,137
No. of matches for which ACARI ratings could be calculated – dataset 3 (D3)	443,356

Apart from that, we retrieved players' rating data from the Ranked MMR Ladder (http://www.heroesofnewerth.com/ladder/), which was then used to calculate player's initial MMR, because the data from the HoN API contains only MMR changes and not the player's initial MMR at the beginning of each match. The "initial MMR" is considered to be the player's MMR in the first match he played, that was recorded in the retrieved match data. It is calculated "backwards" – by subtracting the rating adjustments, available in the match data, from the most recent MMR obtained from the ladder. The initial MMR was also needed to calculate MMR for every role and total MMR used for evaluation of ACARI. Combining the data from HoN Statistics API and Ranked MMR Ladder, we have obtained match data from the December 20, 2014, to the April 29, 2015. Ladder was retrieved on the April 29, 2015. This data corresponds to the dataset D1 (Table 2), which needed to be filtered. First, we removed all malformed entries (e.g., only one player or one team, results exist only for one player, and no information about the hero used). Such matches do not contain all the information needed for calculating the role MMR and total MMR, so they have been filtered out as invalid – resulting in the filtered dataset D2. Finally, the total number of players participating in the matches (166,502) in dataset D2 is greater than the total number of players listed on the ladder (135,007), and it is also greater than the total number of players for whom the initial MMR had been calculated (102,245). A possible explanation for this discrepancy is that some players did not participate in any of the retrieved matches (which could be because they did not play ranked matches, but played other types of matches). The dataset D3 represents all matches for which role MMR could be calculated for all participants. To validate our filtering process, we calculated the distribution of the initial MMR for all players in D3. The best fit was normal distribution with parameters μ=1536, σ=112.5, which is in line with previous findings about HoN's MMR distribution (μ=1528, σ=112) [16]. Obtained datasets are freely available, and interested parties may contact the authors to obtain them.

B. Evaluation process

First we should note the parameters of the ACARI model which were used for the use case of HoN. Using the performance parameters of the team, we define the vector TP:

$$\mathrm{TP} = [\text{no. of killed heroes, experience per minute, gold gain per minute}]$$

Note that the performance parameters comprising the TP are calculated for the team as a whole. Using the performance parameters of an individual player, we define the vector P:

$$\mathrm{P} = [\text{no. of killed opponents, no. of assisted kills, damage to the opponents, experience from kills, no. of deaths, no. of team creeps killed, no. of neutral creeps killed, experience per minute, gold per minute, no. of placed wards}].$$

In HoN there are over 120 heroes available, and each hero has a role vector RV determined by the game designers (this data is accessible through the HoN Statistics API). The defined set of roles R in HoN is:

$$\mathrm{R} = \{\text{Carry, Ganker, Support, Initiator, Jungler, Pusher}\}$$

We evaluate ACARI by comparing the expected match result (ER) according to Elo algorithm for two different input ratings: 1) ratings of the existing system in HoN, and 2) ratings calculated with extending the existing system in HoN with ACARI. The reason for using Elo algorithm is because, to the best of our knowledge, it is the base algorithm for rating and matchmaking system in HoN. The ER ($0 \leq \mathrm{ER} \leq 1$) serves as a predictor of which team is more likely to win (according to a given rating system), and it is calculated by the expression [3]:

$$\mathrm{ER} = \frac{1}{1+10^{-(s_a - s_b)/400}} \tag{14}$$

where s_a and s_b are the average MMRs of the opposing teams A and B, respectively. Value of ER is interpreted as follows:

- if ER is greater or equal than 0.5, it is assumed that team A will win,
- if ER is less than 0.5, it is assumed that team A will lose.

The closer the value of ER is to 1, the more "confident" is the algorithm in the team A's victory. It is also important to note that the expected result determines the balance of the match – the closer the average MMRs of the opposing teams, the more balanced the match. (The balance must be appropriate for a match to start.) Only matches in which all players have a known initial MMR can be considered for evaluating ACARI, meaning that the dataset D3 (with 443,356 matches) is used.

We evaluate ACARI in two ways – by using the role MMR and by using the total MMR. We assume that using the role MMRs will give better results, since it better reflects the skill level of the player for all roles that the player's hero took in that particular match (as recorded in the match data).

VI. Results

The expected result is calculated for the winning team, and six categories of the evaluation outcomes are defined in Table 3.

Table 3 – Evaluation outcome categories

Code	Interpretation
YYA	Both rating algorithms had correctly calculated outcomes, but ACARI was more certain in the calculated result
YYE	Both rating algorithms had correctly calculated outcomes, but the existing HoN algorithm was more certain in the result
YNA	ACARI had a correctly calculated outcome, while the existing HoN rating did not
NNE	Both rating algorithms did not have correctly calculated outcomes, but ACARI was less certain
NNA	Both rating algorithms did not have correctly calculated outcomes, but ACARI was more certain
NYE	The existing HoN rating had a correctly calculated outcome, while ACARI did not

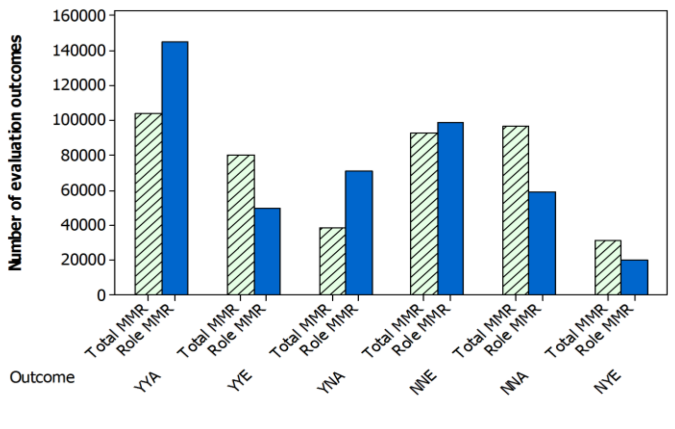

Figure 3 – Evaluation results

Each category is represented by a three-letter code, in which the first letter denotes whether ACARI ratings resulted in a correct prediction of the winner (Y or N), the second letter denotes whether existing HoN rating resulted in a correct prediction (Yes or No), and the third letter shows which algorithm (ACARI, or Existing HoN algorithm), did better in terms of prediction "certainty" (based on the ER value, e.g., ER value of 0.8 means higher prediction certainty than 0.65).

The evaluation results are shown in the Figure 3. In both cases ACARI gives better results compared to results of the current rating system. In total, ACARI predicted the match outcome correctly in 50.22% of the total number of matches when using the total MMR, and 59.99% when using the role MMR, while for the existing HoN ratings that number is 48.48%. In both cases, most of the results belong to the category in which both ACARI and existing ratings correctly predicted the outcome, but ACARI was more certain in the calculated result (YYA). Using role MMR yielded significantly better results than using total MMR, thus increasing the overall accuracy (YYA, YYE, and YNA) and certainty of prediction (increased YYA and YNA). Number of the outcomes when ACARI rating resulted in a correct prediction and the existing HoN ratings did not, compared to reverse case (YNA and NYE respectively) is almost threefold. As expected, role MMRs give better results, but in both cases an improvement over the existing rating algorithm has been convincingly demonstrated.

In practical terms, to enable the use of role MMR, the order of hero selection and the start of matchmaking process should be reversed. Namely, in current HoN implementation, the teams are formed first, followed by the hero selection, and to enable the use of role MMR, it should be the other way round, so that the player chooses a hero first, followed by the matchmaking process.

VII. Conclusions and Future Work

To summarize, ACARI gives better results and outcome is correctly predicted in more than 50% of the total number of matches when using total MMR and almost in 60% when using role MMR. ACARI can be used in all MOBAs based on same principles as HoN (i.e., that every player has a role in the match) with minor adjustments.

In future work we aim to use machine learning algorithms to find out which coefficients that relate parameters to roles would yield the most precise results, to construct methods for exact role detection, and to compare ACARI to similar rating systems mentioned in Section II.

Acknowledgment

This work has been supported in part by Croatian Science Foundation under the project 8065 "Human-centric Communications in Smart Networks"".

References

[1] M. Minotti, "World of Warcraft helps the digital games market generate almost $1 billion a month," Online: http://venturebeat.com/2015/03/12/world-of-warcraft-helps-the-digitalgames-market-generate-almost-1-billion-a-month/ [*accessed Sept. 4, 2015*].

[2] M. Csikszentmihalyi, "Flow: the psychology of optimal experience," New York: Harper Collins, 1991.

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

[4] R. A. Bradley, and M. E. Terry, "The rank analysis of incomplete block designs: I. The method of paired comparisons," Biometrika, Vol. 39, No. 3/4, Dec. 1952, pp. 324–345.

[5] M. E. Glickman, "The Glicko system," Boston University, 1995. Online: http://www.glicko.net/glicko.html [*accessed Sept. 4, 2015*].

[6] R. Herbrich, T. Minka, and T. Graepel, "TrueSkill^{TM}: A Bayesian skill rating system," Advances in Neural Information Processing Systems 20, MIT Press, Jan. 2007, pp. 569–576.

[7] S. Nikolenko, and A. Sirotkin, "A new Bayesian rating system for team competitions," Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 601–608.

[8] D. Pierre, R. Herbrich, T. Minka, and Thore Graepe, "Trueskill through time: Revisiting the history of chess," Advances in Neural Information Processing Systems, 2007, pp. 337–344.

[9] C. DeLong, N. Pathak, K. Erickson, E. Perrino, K. Shim, and J. Srivastava, "TeamSkill: modeling team chemistry in online multi-player games," Advances in Knowledge Discovery and Data Mining. Springer Berlin Heidelberg, 2011. pp. 519-531.

[10] L. Zhang, J. Wu, Z.-C. Wang, and C.-J. Wang, "A factor-based model for context-sensitive skill rating systems," Proc. of the 22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Vol. 2, 2010, pp. 249–255.

[11] O. Delalleau, E. Contal, E. Thibodeau-Laufer, R. Chandias Ferrari, Y. Bengio, and F. Zhang, "Beyond Skill Rating: Advanced Matchmaking in Ghost Recon Online," IEEE Transactions on Computational Intelligence and AI in Games, 2012, Vol. 4, No. 3, pp. 167–177.

[12] S. Agarwal, and Jacob R. Lorch. "Matchmaking for online games and other latency-sensitive P2P systems," ACM SIGCOMM Computer Communication Review, 2009, Vol. 39, No. 4, pp. 315-326.

[13] J. Manweiler, S. Agarwal, M. Zhang, R. R. Choudhury, and P. Bahl, "Switchboard: a matchmaking system for multiplayer mobile games," Proc. of the 9th international conference on Mobile systems, applications, and services, 2011, pp. 71-84.

[14] G. Jiménez-Díaz, H. D. Menéndez, D. Camacho, and P. A. González-Calero, "Predicting performance in team games – the automatic coach," ICAART 2011 Proc. of the 3rd International Conference on Agents and Artificial Intelligence, 2011, pp. 401-406.

[15] G. Di Fatta, G. M. Haworth, and K. W. Regan, "Skill rating by Bayesian inference," IEEE Symposium. on Computational Intelligence and Data Mining, 2009, pp. 89-94.

[16] N. Caplar, M. Suznjevic, and M. Matijasevic, "Analysis of player's in-game performance vs rating: Case study of Heroes of Newerth," Proc. of the Foundations of Digital Games (FDG 2013), 2013, pp. 237-244.

[17] M. Myślak, and D. Deja, "Developing game-structure sensitive matchmaking system for massive-multiplayer online games," Social Informatics, 2014, pp. 200-208.