Matchmaking and Case-based Recommendations

Jorge Jiménez-Rodríguez, Guillermo Jiménez-Díaz, and Belén Díaz-Agudo

Dep. Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid, Spain
email: jorjimen@pas.ucm.es, guille@fdi.ucm.es, belend@sip.ucm.es

Abstract. Matchmaking is a technique that allows players to be matched with others for the purpose of playing an online multiplayer game. These players want to have a fun and balanced gameplay experience, either cooperatively or competitively, and often even in teams. Our current work studies what is the role of gameplay experience and CBR in this process of matchmaking in multiplayer games. In this paper we discuss two different approaches: skill-based and role-based matchmaking. The former generates recommendations using player skill ratings while the latter employs cases about role configurations to propose what match is more interesting for the players.

1 Introduction

With the widespread availability of broadband connections and the emergence of unified online gaming services, more and more games provide online multiplayer modes in which players can play together, either cooperatively or competitively, and often even in teams. Playing with or against other human players can be much more interesting and fun than playing with a typical game AI.

An online multiplayer match can be arranged in several ways. For example a group of friends could gather into a server and configure their own match to play together; also a player can join a match from a list of matches currently in progress in the server.

There is an increasing interest on helping players to find matches where they have an enjoyable gameplay experience. Up to now, matchmaking using player experience levels has shown a strong effect on making a multiplayer game balanced and fun. The key idea behind skill-based ranking and matchmaking is that a game is fun for the participating players if each of the them has a fair chance of winning [3]. Current matchmaking techniques can also add additional preferences based on basic social and technical features (such as players' language and country, age, server bandwidth, the game type or the game map).

In this paper we discuss matchmaking approaches from a CBR perspective taking into account previous gameplay experiences. We consider matchmaking as a process that either finds players with similar skills or finds matches that are similar to a set of enjoyable configurations of matches. These configurations are structured cases with information about the behaviour of every player during the match. The analysis of the player behaviour in matches, the reuse of past match

experiences and the inclusion of additional player preferences in matchmaking techniques for recommending a match should enhance the player satisfaction during the recommendation process.

The paper runs as follows. Section 2 describes the matchmaking process as case-based recommendation. In order to perform the recommendation, we describe two basic similarity approaches: based on *skills* and based on *roles*. Section 3 details a recommendation process based on player similarity using skill ratings and Section 4 describes a new recommendation approach as a case-based process as looking for player similarities using roles. Section 5 concludes the paper and presents the ongoing lines of future work.

2 Matchmaking as a Case-based Recommendation Process

Matchmaking consists on recommending a match to a new player. We define two different scenarios of matchmaking in a multiplayer game:

1. Recommend an existing match. This match has not started yet because it is waiting for new players. It is characterized by the players who are currently involved in the match.
2. Recommend a set of players to organize a match. Several players are in the lobby, looking for a match, but none of them are still involved in a match.

In both scenarios matchmaking is based on measuring the similarity among players involved in the match. This similarity compares the gameplay experiences of each player in previous games. This can be measured using two approaches. The first is the skill-based approach (SBA), detailed in Section 3, which is responsible for generating recommendations based on player skill ratings in order to recommended balanced matches. The second, detailed in Section 4, is the role-based approach (RBA), which uses the roles that the players have performed in previous experiences for recommending matches that satisfy the player expectations about enjoyment.

No matter the employed approach, the similarity or compatibility among players is based on the behaviour of each player in previously played games. Player behaviour is defined using playing features extracted during the different matches that the player took part in. The information employed to characterize a player i in a match m –P_{mi}– is a set of metrics $< s_{i1}, .., s_{in} >$ extracted during a match. These metrics depends on the type of game because the information needed to characterize a player is not the same in a first-person shooter or in a race game. For example, in the former it is important the number of kills of a player or the number of times that he was killed. However, in the latter it has more significance the average speed, or the number times that the player crashed his car.

Additionally, matchmaking process should take into account the player behaviour over time. The player profile P_i compiles all player experiences P_{mi}. This profile includes additional synthetic features computed using the metrics

extracted during the matches. These new features will be different according to the method employed when performing the matchmaking – SBA or RBA.

Case-based recommendation is employed in product recommendation domains where individual products are described in terms of a structured set of features [9]. In matchmaking, we consider the matches as the recommended products. Every match m is characterized as the union of the descriptions of every participant $\{P_1, ...P_k\}$, and a set of additional features, like the type of match or the total match time. So, we have a structured description of each player and case-based recommendation techniques can be employed for our purpose. The recommendation is then reduced to find the most similar players to the one who is looking for a match to play.

The case base of items to recommend contains player profiles, more precisely, the player skill rating (SBA) and the player roles observed during previous matches (RBA). However, not all the case-base is employed during recommendations. On the one hand, when we recommend a set of players to generate a new match we employ the players who are currently online and who are not playing in a formed match. On the other hand, only the stand-by matches and, more precisely, the players who conform these matches are used when recommending an existing match. Additionally, the last scenario needs to aggregate the information concerning to each player in the match in order to find the match where the players are *similar* to the one that requests the recommendation. Here, aggregation techniques employed in group recommenders could be applied [7].

Additionally to using a case base of players, we have a case base populated with matches, past experiences of gameplay. Every match in the case base will be a representation of an enjoyable experience. A match is selected as a case experience because an expert –for example, a game designer– describes which player configurations will be enjoyable when playing the game, or because players are requested to assess the gameplay experience after finishig the game. Once these enjoyable matches are defined, the recommendation lies in selecting a match that is currently waiting for players or configuring a match with the players in the lobby, which is similar to the retrieved enjoyable ones.

3 Skill-Based Approach

In this section we describe an approach to provide individual recommendation in multiplayer games. The goal is to recommended to a player the matches whose "level" is in keeping with the player's "level". The recommendation approach described in this section follows the first alternative described in Section 2: the similarity between a player and a match is measured in terms of the similarity between the skill rating of the concerned player and the skill ratings of the other players in the match. In this case, matchmaking lies in joining online players with similar *skill ratings* to the concerned player.

SBA takes into account player expectations over time recommending matches that will maintain the ratio of won/lost games as balanced as possible. We exemplify this approach employing one of the rating systems that we will describe in

Section 3.1. These systems calculate the player skill rating using scoring values, and then use these skill ratings to generate recommendations. Player scoring is computed using the statistics that represent their ability in the game. Skill ratings will be updated after the match has finished.

The recommendation process is divided into two different tasks: the rating update and the recommendation itself. First, we describe several rating systems that can be employed during the skill rating update. Section 3.2 describes the ideas behind the skill rating update process itself, while Section 3.3 concerns to the use of these ratings in order to perform the recommendations.

3.1 Rating systems

A rating system is a method to calculate the relative skill ratings of players in two-player games where the result is a win, a loose or a draw. However, these systems can be adapted to n-player games by transforming the global result into a set of partial results of two-player confrontations.

The Elo rating system [1] is a well-known method for calculating the relative skill ratings of players in two-player games. The Elo system was invented as an improved chess rating system, but today it is also used as a rating system for multiplayer competition in a number of computer games and it has been adapted to team sports

Glicko[1] is a rating system that extends the Elo's by incorporating a measure of uncertainty of a player's rating. The Glicko rating system [2] is a method for assessing a player's strength in games of skill. The main idea is the introduction of a measurement for the ratings reliability called RD (for ratings deviation). The RD measures the accuracy of a player's rating. Twice the RD is added and subtracted from their rating to calculate this range.

Glicko is the base of the well known system called TrueSkill [5], a Bayesian ranking algorithm and matchmaking system developed by Microsoft Research and established in the Xbox 360 live services. With TrueSkill a new player joining million-player leagues can be ranked correctly in fewer than 20 games. In TrueSkill a player's rank is represented as a normal distribution $\aleph$ characterized by a mean value of μ (representing perceived skill) and a variance of σ (representing how "confident" is the system on the player's μ value). It can predict the probability of each game outcome, which enhances competitive matchmaking, making it possible to assemble skill-balanced teams from a group of players with different abilities.

3.2 Skill rating update

The player skill rating value summarizes all the experience of matches stored for the player. The skill rating updating process depends on the player scoring in a match. Player scoring in a certain game depends on the type of game.

[1] A comprehensive description of Glicko can be found at `http://math.bu.edu/people/mg/glicko/glicko.doc/glicko.html`

Player scoring should depend on the player behaviour during the match, the events occurred and the achieved goals. All these variables are extracted from the player's actions in the match and the effects produced by these and other players' actions. The most significant metrics $< s_{i1}, .., s_{in} >$ will be collected for each player i and stored.

The final result of a match in a multiplayer game is represented by an ordered list of the players. This ranking is commonly created based on player scorings in the match: the higher scoring, the higher position in the match rank. However, the player skill rating characterization can not be extracted using only this global result. The one-on-one partial results of every pair of players and their current skill ratings should be taken into account. We update the player rating skill every time a game has finished. This way, the player rating will be ready when it will be employed during a recommendation process.

As an example we propose the use of the following general formula of the Elo rating system for updating the skill rating of players after every match:

$$R_{post_A} = R_{prev_A} + k \cdot (S - E) \tag{1}$$

The R-values represent the player skill ratings. R_{post_A} represents the upgraded rating for a player A after the individual confrontation with an opponent and R_{prev_A} is the rating that the player A had before starting the game. S is the actual result obtained by player A after the match. E represents the expected result of the game for the player A. Finally, k represents an attenuation value that can be interpreted as the weight given to the new match in order to update the previous rating. This formula is calculated for each player as many times as different opponents in the game.

Score values for every player are computed using the most significant features of the player during the match P_{mi}. S is later calculated using these score values, which represents the score obtained by player A after the match between two players. Having two players (A, B) with a scoring of $(score_A, score_B)$, and A is the player to which we update its rating :

1. If $score_A > score_B \rightarrow S = 1$
2. If $score_B < score_A \rightarrow S = 0$
3. If $score_A = score_B \rightarrow S = 0.5$

These results are discretized to these values due to its simplicity and because E takes values in the same range. To calculate the value of E, which represents the expected result of the game for the player A in the range of [0 .. 1], also used the formula provided by Elo rating system:

$$E = \frac{1}{1 + 10^{\frac{R_B - R_A}{Range}}} \tag{2}$$

R_A is the rating of player A (the one we want to update). R_B represents the opponent's rating value. $Range$ represents the number of points between two distinguishable levels. The process computes all the R_{post_A} values (using Equation 1) resulting of player A matches with every other opponent. Finally, we update player A rating using the average value.

3.3 Recommendation process

Once we have updated the rating for each player it is time to find the most "balanced" matches in order to generate the recommendation. As we described in Section 2, when using the skill ratings we can find the similarity between a match and the player computing the skill rating of the match as an aggregation of the information for every player who is going to play the match.

The aggregation process is inspired by the works in recommendation to groups [7]. The main approaches to generate a preference aggregation based on the individual user preferences are (a) merging the recommendations made for individuals, (b) aggregation of ratings for individuals and (c) constructing a group preference model. In this paper, we use the aggregation of the individuals due to its simplicity. We can test different alternatives of the aggregation and similarity functions. A match m is characterized by the union of the profiles of every player $\{P_1, ...P_k\}$, which contain every player skill rating. $|m|$ is the number of players in the match. Below we propose two methods for recommending matches using similar skill ratings:

- Average skill rating of the game. This simple function considers that the skill rating of a match m is the average of the skill ratings from the players that conform the match. The similarity function returns the difference between the query player skill rating (R_p) and the match average skill rating.

$$Sim(p, m) = 1 - \left| R_p - \frac{\sum_{i=1}^{|m|} R_i}{|m|} \right| \tag{3}$$

- (R2) Weighting category differences. Skill ratings are discretized into ranges or skill categories. We can consider that a match is more similar to a player as long as more players participating in the match belongs to the same skill category of the concerning player. If we have N skill categories and a player P belongs to the category $Cat(P)$:

$$Sim(p, m) = \frac{\sum_{i=0}^{N} DistW(i, Cat(p)) \cdot numPlayers(m, i)}{|m|} \tag{4}$$

  $numPlayers(m, i)$ counts the number of players in the match that belongs to the category i. $DistW(i, j)$ is a weight that depends on the distance between the categories i and j. The longer distance between categories, the smaller the weight.

Once the similarity is computed, the matches are ordered by similarity. The matches at the top of the recommendation will be the ones more similar according to the skill rating of the player searching for a match.

4 Role-Based Approach

This second approach represents a novel idea in matchmaking. Using only skill ratings to generate recommendations can get balanced matches, but the matches

can be monotonous and boring. Using the player roles, the matchmaking approach can form matches with a wide variety of behaviours so that the game can be entertaining and diverse. Moreover, this new approach can take into account additional user's preferences that can change over time. The matchmaking is not limited to balanced games and RBA adds new features to the game experience, such as the type of game that the player wants to play –for example, a short match with heavy players or a more strategical match– or additional goals that the player wants to achieve –for example, matches for training concrete abilities.

To achieve that satisfaction in the game, the idea is to create prototypical role configurations (that may represent matches, teams, or even both), which will be then used in the recommendation process. These configurations are structured in cases that contain information about the roles adopted by each player in a prototypical match. Following this approach, the recommendation process focuses on finding matches made of a set of players whose observable behaviour fits one of these prototypical configurations.

To carry out this idea there are three key steps that are defined below: the definition and update of the player models, the creation of the case base with prototypical game configurations and the recommending process itself.

4.1 Roles and player modeling

The generation of player models, more specifically for this case, the role extraction, three tasks to perform:

- Define the roles set.
- Infer player roles from the metrics extracted during the matches.
- Update global player roles based on the obtained roles in each match.

The first task is to determine how many and what type of roles will be used to characterize player behaviour. This decision will depend on each type of game. The roles are not the same in a FPS game than one of sports but for many games within the same type may coincide. These roles can be defined by an expert in the game. For example, in a FPS game a player can be characterized with the role of a sniper –small number of shots, high kill rates and no movement after achieving a concealed position– or a suicide –high number of shots, high number of kills and own deaths and fast movements along all the scenario.

The second task describes how the roles should be inferred from the player behaviour in a game. Player modeling techniques can be used in this task [6]. For each role defined in the set we should create a method that estimates the membership of a player to that role using the metrics generated by the player during a match. Each method can take into account different metrics, according to the ones that are most important to characterize this role. A player may be characterized by various roles in different degrees, not only has to be associated with a particular role. The role extraction process must be performed after every finished match and for all the participants.

The third task must reflect the overall behaviour of the player throughout his career in the game. This requires an update method to get the player's overall

performance after each match played by the player and once the observed roles are extracted. Using the role values obtained in the second task, the process must update the global player roles by some method that reflects adequately the exact roles of the player (e.g. average). This is necessary because these roles will be used then in the recommendation process.

4.2 Creation of the case base

Besides the modeling process, our approach also requires the creation of a memory with cases containing the acceptable configurations. A configuration can be characterized by the roles that each player plays during the match. The contents of the cases depends on the type of game –individual or team. Configurations of individual matches can contain information about the individual roles of each player, while team game configurations can indicate which team each player belongs to and the roles that they adopt. Another key point when creating the cases is the importance of the number of players within each configuration because then will have to take account into the recommendation process.

The creation of these cases is not a trivial matter, and there are several methods for doing it. As stated before, an alternative is the generation of prototypical configurations using the knowledge of an expert in the game. Other alternatives are the generation of these configurations using information about what the players themselves consider fun, like in the work described in [4], or observing and analysing characteristics of gameplay as [8].

4.3 Recommendation process

At this stage the recommendations are generated based on the existing resources in the system and the cases stored in memory. The recommendation consist on computing the similarity between attempting matches within the target player and selected cases from the case base. The best matches –according to the similarity with the selected case– are proposed to the player.

To determine the similarity between a prototypical configuration in the case base and a match we have to determine the similarity between each role in the proposed case and every player in the match. Therefore, two similarity measures are essential to carry out this recommendation process:

- Similarity between players and roles. Since the player profile contains information about the degree of membership of the player to each predefined role, we need to measure how similar is a player with every role in the case. Although the similarity measure can be easily defined using the highest degree of membership of the player as her dominant role, we can establish additional similarity measures between roles. This way, we can adapt a match to a configuration by replacing some roles in the configuration with the players characterized by similar roles, although they do not match exactly with their dominant roles.

– Similarity between a case and a match. The similarity between a case and an attempting match is measured in terms of the similarities between the players in the match and the roles of the case. For each real player in the hypothetical match,we compute the similarity with each of the roles stored in the case (using the similarity method of the previous point). Then, we have to combine this individual similarities to compound a global one. Finally, the similarity between the case and a match will be the the maximum similarity value among all the possible combinations. As we can see, the problem here is that the number of different configurations grows extremely fast as long as the number of players in the prototypical case increases.

According to the possible scenarios that were described in Section 2, the following may occur:

1. In the case that a player will join to one of the matches that are still waiting for new players, the matchmaking should try to include the player in each of the existing matches and, using similarity functions, comparing with the prototypical configurations in the case base to determine which is best match based on player expectations. Not all stored cases are used in the comparison, but only those with similar characteristics to the evaluated match (e. g. the number of players). Once evaluated and obtained a similarity value for all matches, the ones with the higher similarity values will be recommended.
2. The scenario where a set of players are arranged for playing a match should be similarly resolved as described above, except that to make matches with players will have to create all possible combinations for the selected players. This is a combinatorial problem that should have a high cost in time. An alternative is to prune the number of players using only the players with similar characteristics, such as the skill rating seen in Section 3.

5 Conclusions

In this paper we propose the application of case-based recommendation approaches to matchmaking tasks in videogames. First, we have detailed how the matchmaking can be performed as a case-based recommendation approach. The analysis of the player behaviour in matches, the reuse of past match experiences and the inclusion of additional player preferences in matchmaking techniques for recommending a match will enhance the player satisfaction during the recommendation process. We have discussed two different ways to find the matches that best fit the player expectations: skill-based approach and role-based approach. For skill-based approach we have described some general methods that can be used to carry out such recommendation, giving some examples of use. In the role-based approach we have sketched the steps and tasks to be followed to develop it. In our opinion, the latter is a more interesting approach because it could fit additional player preferences that the skill-based approach cannot satisfy, producing more enjoyable gameplay experiences. However, the cost of

the similarity metrics due to the combinatorial aspects of this approach should need a deeper analysis.

As a future work we want to focus on the role-based approach and analyse its performance. More concretely, we want to carry out a case study about the application of these two approaches to Unreal Tournament 2004, a First Person Shooter game. Finally we need to develop evaluation measures that will help us to study how satisfactory have been the recommendations for both approaches, in order to compare them.

Acknowledgements

This work has been supported by the Spanish Ministry of Sci. and Ed. (TIN2009-13692-C03-03).

References

1. Glickman, M.E.: Chess rating systems. American Chess Journal 3, 59–102 (1995)
2. Glickman, M.E.: Parameter estimation in large dynamic paired comparison experiments. Applied Statistics 48, 377—394 (1999)
3. Graepel, T., Herbrich, R.: Ranking and matchmaking. Game Developer Magazine pp. 25–34 (2006)
4. Hagelbäck, J., Johansson, S.J.: Measuring player experience on runtime dynamic difficulty scaling in an RTS game. In: Procs. of the IEEE 2009 symposium on computational intelligence and games: CIG'09. pp. 49–52. IEEE Press (2009)
5. Herbrich, R., Minka, T., Graepel, T.: TrueSkill(TM): a bayesian skill rating system. Advances in Neural Information Processing Systems 20, 569—576 (2007)
6. van den Herik, H., Donkers, H., Spronck, P.: Opponent modelling and commercial games. In: Kendall, G.; Lucas, S. (ed.) Procs. of the IEEE 2005 symposium on computational intelligence and games: CIG'05. pp. 15–25 (2005)
7. Jameson, A., Smyth, B.: Recommendation to groups. In: The Adaptive Web. Lecture Notes in Computer Science, vol. 4321, pp. 596–627. Springer (2007)
8. Pedersen, C., Togelius, J., Yannakakis, G.: Modeling Player Experience in Super Mario Bros. In: Procs. of the IEEE 2009 symposium on computational intelligence and games: CIG'09. pp. 132–139. IEEE Press (2009)
9. Smyth, B.: Case-based recommendation. In: The Adaptive Web. Lecture Notes in Computer Science, vol. 4321, pp. 342–376. Springer (2007)