39

Analytics-Based AI Techniques for a Better Gaming Experience

Truong-Huy D. Nguyen, Zhengxing Chen, and Magy Seif El-Nasr

39.1 Introduction

Recently, it has become increasingly common to gather gameplay data through telemetry logs. These data can provide valuable insight into the player's interactions with the game and are typically used by designers to balance the game, but they can also be used to algorithmically adjust game systems in order to create better engagement. Specifically, gameplay data analytics can provide ways to (1) algorithmically adapt game content and mechanics on the fly; (2) recommend games to play on game portals, such as Steam; and (3) match teams in multiplayer games.

Adapting game content and design to the user's behaviors and responses has gained momentum in the recent years [Piggyback 09; Newell 08]. Titles that adjust difficulty and/or game mechanics based on the user's skill level and performance include *Fallout 3* [Bethesda 08], *Fallout: New Vegas* [Bethesda 10], *Left 4 Dead* [EA 08], *Left 4 Dead 2* [Valve 09], and *Resident Evil 5* [Capcom 09]. Difficulty adjustment can increase engagement and thus increase user retention. While there are many different methods to develop

adaptive games, in this chapter, we will focus on AI algorithms that use a model of the user to adjust game content or mechanics in real time.

Matching teams within multiplayer games is another trend that has gained momentum [Dota Team 13; Lefebvre 14; GO 13]. Notable examples include *Defense of the Ancients 2*, often better known by its acronym *DotA 2* [Valve 13]; *League of Legends*, a.k.a. *LoL* (all seasons) [Riot 09]; and *Counter Strike: Global Offensive* (CS:GO) [Valve 12]. In these examples, user performance and skills can be gauged through data collected during play (i.e., telemetry data), and team matching can be done based on this analysis.

We reviewed numerous articles in both industry and academia that report the use of analytics to drive better experiences through adaptation, recommendation, or team matching. The approaches taken can be categorized into five groups:

1. *Skill-based adaptation*: Techniques that aim to adapt the game mechanics to the user's skill level. The goal is to maintain a level of challenge that will not overwhelm players with overly hard tasks but will keep them challenged and engaged.
2. *Emotion-based adaptation*: Techniques that change aspects of the game to influence the emotional process of players based on their affective states. The goal can be to evoke certain kinds of emotions or to modulate the player's emotions for a more relatable game experience.
3. *Style-based adaptation*: Techniques that adjust elements in the game (e.g., content) based on the player's play style.
4. *Game recommendation*: Techniques that reach beyond the scope of just one game, recommending games to play based on the player's past choices. Recommendation systems develop a model of the player's preference based on their game choices, rating scores, etc. Using this information and information from other players collectively, such systems can suggest games that players may enjoy, which results in a better experience, more engagement, greater retention, and higher ratings.
5. *Team Balancing*: In games that allow team-based battles, matching appropriate team members/opponents based on their skills can directly impact the experience. Most automated matchmaking systems implemented in commercial titles, such as that of *DotA 2*, *LoL*, or *CS:GO*, match teams with similar skill levels against each other [Dota Team 2013].

In this chapter, we will discuss these systems through a review of two sets of works. First, we will discuss algorithms and techniques that will allow us to model users in terms of performance, preference, or behaviors. This work is the focus of game analytics. Game analytics is a set of techniques drawing from data mining, machine learning, and statistics to generalize or understand play-related data collected through play sessions. Second, we will discuss examples to show these approaches in action, in particular for adaptive games. We will show how game content and mechanics can be modeled and adjusted algorithmically in order to tune them to a user model driven from analytics results. We will draw on examples from academia and industry, indie and AAA titles, as well as single-player and multiplayer games to show how such techniques can be used across a wide variety of game types.

39.2 Game Analytics Approaches to Modeling User Performance, Skills, and Behaviors

Game analytics refers to the process of collecting data pertinent to how players interact with the game world, analyzing that data to uncover implicit patterns and trends, and finally converting the data into useful knowledge [Seif El-Nasr 13]. Play data collected for this purpose are usually referred to as *game telemetry*, which is raw, time-stamped stream of actions taken and player-related events happening in the game. Usually, such data are logged and transmitted for storage in a remote database, thus the name telemetry.

39.2.1 Game Metrics

Once we have the telemetry data, we next need to define a set of features that will allow us to evaluate the player's experience, skills, behaviors, and/or preferences. These features are called *game metrics* and are defined as quantitative measures of one or more attributes of interaction in the context of the game. For example, completion time for each level can be a metric. Depending on the game genre, different metrics can be derived from the telemetry data. Examples of metrics for popular genres include

- *First-person shooters*: Types of weapons used, types of item/asset used, character choice, route choice, loss/win quota, team scores, vehicles used, strategic point captures/losses, jumps, crouches, and special moves
- *Strategy games*: Types of commands given to units, upgrades purchased, win/loss ratio, team/race/color selection, maps used, map settings, match/game settings, and time spent on building tasks versus unit tasks
- *Platformers*: Jumping frequency, progression speed, items collected, power-ups/abilities used, and damage taken/sources of damage
- *Role-playing games* (*RPGs*): Character progression, quest completions, quest time to complete, character ability/item use (including context of use), combat statistics, AI-enemy performance, story progression, nonplayer character (NPC) interactions, damage taken and sources of damage, and items collected

39.2.2 Analysis Techniques

Given these metrics, several techniques can be used for analysis of these data to derive a user model in terms of performance, skills, and behavior. Analysis techniques usually fall under two categories: individual and community based.

39.2.2.1 Individual Analysis

Individual analyses operate on data pertaining to a *single* player, aiming to learn the different processes that drive the targeted player's in-game actions. Here are the common goals of individual analysis:

1. Adapt game difficulty to the skill of an individual player so that the player is faced with an appropriate level of challenge.
2. Predict the player's gameplay behavior to tune the design increasing player retention.

3. Generate personalized game content to satiate the player's appetite, including narratives, maps, and skins.
4. Develop game AIs that match the player's play style and skill level.

We will discuss how these goals are achieved using concrete examples, most of which start from player models. *Player modeling* is defined as the development of computational techniques to model players' cognitive, behavioral, and affective (i.e., emotional) state based on play data or theory [Yannakakis 13]. The development of a player model is usually done using machine learning algorithms, such as artificial neural networks (ANNs), support vector machines (SVMs), or Bayesian networks. In some other cases, simple models using numeric vectors, updated with hand-tuned rules, are also adopted.

ANNs [Bishop 06] are supervised learning algorithms inspired by animals' nervous systems. They are constructed as a network of *neurons* (represented as nodes), divided into one input layer, zero or more hidden layer(s), and one output layer. Each neuron processes its input using an activation function, usually either a sigmoid or a step function. Nodes between adjacent layers are connected by directed links pointing from lower-layer (i.e., closer to the input layer) source nodes to higher-layer target nodes, with the output of the source nodes fed as part of their target node's input. More specifically, the input of a target node is a linear combination of all its source nodes, with coefficients defined by the links' weights.

Before using these algorithms for prediction, ANNs need to be trained, which is an automated process that configures all of the links' weights so that, for each set of input layer values, the values in the output layer approximate those defined in the training data set.

In the context of player modeling, ANNs are often used as a compact predictor that maps from game states/situations to the corresponding player actions [Togelius 07; Tang 09]. Togelius et al. [Togelius 07] presented an approach to model players' driving styles in racing games using ANNs. The player model starts with some default values and then is progressively trained during play using data collected at 30 waypoints along the track. The input features include

1. The speed of the car
2. The angle between the car's heading and the direction to the current waypoint
3. The distance between the car and nearby walls

The output is one of nine action commands in controlling the car, including the arrow keys and combinations of them.

A multiobjective evolutionary algorithm is then used for training the network to optimize three fitness functions matching the observed human's behavior in terms of distance traveled, speed, and maneuver direction. The algorithm represents each ANN candidate as a vector of weight parameters and thus as one individual in a set of candidates. Through generations of evolution (consecutive selection with respect to each fitness function, then multiplication, and mutation), an ANN that approximately optimizes the three aforementioned objectives is achieved and used as the player model.

An *SVM* [Burges 98] is a supervised learning technique that uses a discriminative classifier able to categorize incoming data. Given labeled training data, the algorithm outputs the hyperplane that best separates it. This hyperplane can then be used to categorize

new samples. It can, thus, model the player's play style in a similar manner to ANNs, taking as input a game state and returning the most probable action the player would take.

Missura and Gartner [Missura 09] used SVMs to classify players' play styles in a 2D shooting game where the goal is to destroy alien spaceships. The data used for modeling are gathered from multiple plays of 17 test players, who can set the difficulty level while playing. Each play record, which is at most 100 s long, is first normalized into a fixed-size sequence of time-stamped feature tuples, which record the game state at 100 ms intervals. The features used included the score, the player's health, and the current selected difficulty level. The play records are split into two parts. The input data to the SVM are the first 30 s (without the difficulty information) of play, while the training labels are identified through cluster analysis, namely, K-means [Bishop 06], on the remaining 70 s play data. The SVM is trained to divide its input data into the categories generated by the K-means clustering. Thus, we end up with a predictive model that is able to classify new players into one of K player types based on their health and score over the first 30 s of play. The difficulty is then adjusted according to the average difficulty in that player type. One advantage of this approach is that player types are not crafted by hand but are generated from the training data. Once a player has been categorized as a particular type, his or her difficulty level can be adjusted automatically to be the average difficulty selected by the cluster that the player belongs to.

Bayesian networks [Neapolitan 04] provide a means for representing independency relationships among events' occurrences, as well as a mechanism for inferring the occurrence states of pertinent events under uncertainty. A Bayesian network consists of a collection of nodes, along with directed links that connect them. Unlike ANNs, links here represent the probability that the target node will be true (or false) if the source nodes are observed to be true (or false). In the field of player modeling, Bayesian networks can cope with uncertainty in the player model, allowing for inference on a set of *class variables* given a subset of the input *observation* features, rather than a complete representation of them.

Yannakakis et al. [Yannakakis 05] used Bayesian networks to capture the relationship between a player's behavior statistics and the parameters that affect game content generation. They developed a modified version of Pac-Man where, much like in the original, the player's goal is to collect pellets while avoiding being killed by ghosts. Unlike the original, Yannakakis' version has ghost generators that take as input two parameters: e_v (the ghost evaluation parameter) and p_m (the ghost diversity parameter), which directly affect how competitive the game is. The game uses a Bayesian network in which the class nodes are e_v and p_m, and the observation nodes are score, time-played, numeric measure of player aggressiveness, the initial interest in the game, and the relative interest difference after 10 games. Interest is computed as the linear combination of three metrics quantitatively calculated from game data, which are the challenge metric (based on the difference between maximum and average player's lifetime over 50 games), diversity metric (based on the standard deviation of player's lifetime over 50 games), and aggressiveness metric (based on the stage grid-cell visit average entropy of the ghosts over 50 games). Training data were generated using three different AI players, each with a different strategy that represents a different player type. Each of the AI players was run against multiple predefined configurations of (e_v, p_m). After being trained against the results of these sessions, the Bayesian network was able to infer from an incomplete set of observations (score, time, aggressiveness, etc.), a setting of (e_v, p_m) that will maximize the player's interest.

Numeric weight vectors provide a simple yet popular alternative to machine learning techniques for modeling players. The weight vector indicates how confident the system is in categorizing the player to an archetype. Thue et al. [Thue 07] adopted this method to adjust storylines in an RPG developed using the *Aurora Neverwinter Toolset* [Atari 02]. Thue's archetypes categorized players as Fighters (who prefer combat), Power Gamers (who prefer gaining special items and riches), Tacticians (who prefer thinking creatively), Storytellers (who prefer complex plots), and Method Actors (who prefer to take dramatic actions). The player model expressed a 5D weight vector, with each dimension representing how closely the player's actions resemble those expected from one of the five base types. During the development phase, designers manually identify all actions that are associated with the various archetypes. When the player takes one of these actions, the weight vector is updated accordingly. For example, a player might start with a weight vector of <Fighter = 1, MethodActor = 81, Storyteller = 1, Tactician = 1, PowerGamer = 41>, meaning he is classified as most strongly as a Method Actor but is also something of a Power Gamer. After he asks for a reward for helping an NPC, the vector is updated to <Fighter = 1, MethodActor = 81, Storyteller = 1, Tactician = 1, PowerGamer = 141>, in order to account for the fact that this player has shown an interest in gaining riches. These techniques (and a number of variants on it) are discussed in more detail in Chapter 42.

39.2.2.2 Communal Analysis

In contrast to individual analysis, communal analysis aims to uncover the general patterns or trends of a subset of, if not all, players collectively. Typically, the goal is to cluster player behaviors to recognize different distinct groups; a player group could be a community in a game or a set of players who share similar playing styles. Communal analysis relies on data mining techniques, including clustering algorithms, such as K-means [MacQueen 67], C-means, principle component analysis (PCA) [Jolliffe 02], nonnegative matrix factorization (NMF) [Lee 99], archetypal analysis (AA) [Cutler 94], and other variants of these algorithms.

Clustering algorithms: Given a set of data points, clustering algorithms aim to group similar data points together to form several representative groups such that the number of groups is much smaller than that of the data points. The resultant group data can then be used for classification (K-, C-means, and NMF) or data compression tasks (PCA).

Although all clustering algorithms share the same goal, their requirements and outputs may vary. In particular, while K-means and C-means both assign one unique group label to each data point, C-means additionally restricts the cluster centers to be among the actual data points instead of just averages, that is, medians in place of means. NMF, unlike K-means and C-means, allows data points to fuzzily belong to multiple clusters by representing them as linear combinations of different cluster centers. Moreover, it requires that all data points, group centers, and each data point's membership to groups contain only nonnegative values, making both inputs and clustering results interpretable by human experts. While having similar outputs as NMF, which are basis vectors able to compose the bulk of input data points, PCA aims to capture the main directions of fluctuation or variance exhibited by the input data points. Intuitively, these fluctuations are the most representative components of the point set.

Unlike clustering analysis, whose goal is to find *average* points of a data set, *AA* looks for *extreme* points called *archetypes*. As such, any point in the data set can be interpreted as a combination of these archetypes. In practice, cluster analysis tends to yield similar basis vectors for different groups, which renders the player styles hard to distinguish from one another even for domain knowledge experts. On the contrary, because of the extremeness, archetypes are data points that have the most prominent characteristics and thus easy to interpret by a human. AA is commonly used to find extreme profiles (outliers) and can be used to detect bots, cheating, or extreme player behaviors. In AA, each archetype is a linear mixture of the data points. Each data point is clustered as a combination of all archetypes. AA, however, comes with significant computation cost, which makes it not ideal for large data sets, as its underlying algorithm involves solving many constrained quadratic optimization problems consecutively.

Simplex volume maximization (SIVM) is a variant of AA that was proposed in order to cope with large data sets [Thurau 10]. It exploits the fact that optimal basis vectors maximize the volume of the simplex they construct as vertices. Similar to AA, basis vectors in SIVM are as far (extreme) as possible to each other to maximize the volume of such simplex. The computation needed to find basis vectors in SIVM is however much lower than that in AA due to its algebraic derivation.

To cluster players of *Battlefield: Bad Company 2* [EA 10], Drachen et al. [Drachen 12] applied both K-means and SIVM to the game telemetry of 10,000 players [Drachen 12]. The telemetry data collect 11 game metrics that are closely related to

- Character performance (e.g., score, skill level, accuracy)
- Game asset use (kit stats, vehicle use)
- Playtime

Mixing different unnormalized data metrics may skew the results of clustering, because the metrics tend to contribute unevenly in dissimilarity measurement. The authors employed variance normalization to preprocess the telemetry data from different metrics before running K-means or SIVM. The presence of ratio data (e.g., kill/death ratio, win/loss ratio) may result in zero values in denominators, which required the authors to implement several smoothing algorithms. The two clustering algorithms yielded seven player profiles, among which the three player prototypes (Assassins, Veterans, and Target Dummies) are shared by both algorithms although the percentage of each may vary (see Tables 39.1 and 39.2). For the remaining clusters, SIVM results in profiles that exhibit a higher degree of difference than K-means.

Later, Drachen et al. conducted comprehensive experiments to a data set of 70,000 players' from World of Warcraft [Drachen 13]. The data set included playtime and leveling speed. They compared different clustering algorithms to develop a model of players' performances. Specifically, they used SIVM, K-means, C-means, PCA, and NMF. Results show that the rendered basis vectors are more intuitively interpreted when using SIVM, K-means, and C-means than PCA and NMF. K-means and C-means are the only two among the five algorithms that hard-label players (i.e., each player could only be assigned to one cluster, rather than being assigned with a linear combination of weights from all clusters). Since SIVM is a variant of AA, which represents data points by extreme archetypes, it results in the most distinguishable basis vectors.

Table 39.1 Behavior Clusters Resulted from SIVM (%P Is the Percentage of Players)

Title	%P	Characteristics
Assault Recon	1.4	Active kill and death statistics (high KpM and DpM), low hit accuracy, average scoring ability, second highest kill/death ratio overall
Medic engineer	0.8	Using vehicles more often, high skill level and hit accuracy, good at scoring
Assault specialist	5.0	Focus on assault, but low score, high death statistics and playtime, low skill level, kill/death ratio, and hit accuracy
Driver engineers	1.1	Extremely obsessed in driving vehicle; high playtime, score, and hit accuracy; lowest death statistics; inactive in killing enemies
Assassins	61.6	Highest kill/death ratio, lowest playtime, active kill statistics while keeping from dying efficiently
Veterans	2.01	Highest score, playtime, and rounds played, overall high values
Target Dummies	28.1	Lowest kill/death ratio, low scoring ability, minimal scores for all features but playtime and rounds played

Table 39.2 Behavior Clusters Resulted from K-Means (%P Is the Percentage of Players)

Title	%P	Characteristics
Snipers	7.4	Median scoring ability, overall low-middling values, high death statistics, extremely high hit accuracy
Soldiers	27.9	Median scoring ability, overall low-middling values, high DpM
Assault engineer	13.1	Similar to Soldiers but better skill value, high kill/death ratios
Target Dummies	26.0	Lowest scores for all values (including playtime) except high death statistics
Trainee Veterans	10.7	Similar to Veterans, but second rank in most features, lower playtime
Assassins	10.9	Highest rank in kill/death ratio, most active kill statistics, low playtime
Veterans	4.1	High playtime, second rank in most features, highest overall skill level

39.3 Game Analytics and Adaptive Techniques

We will examine three classes of adaptive techniques:

1. *Skill based*: Adjusting the level of *difficulty* or *challenge* to match the player's skill level
2. *Emotion based*: Personalizing the game content to the player's *affective state*
3. *Style based*: Adapting the game's content and/or mechanics to the player's preferred style

Of these three categories, skill-based adaptation is the most popular in tailoring players' gaming experience, while emotion-based and style-based customization techniques are gaining more attention.

39.3.1 Skill-Based Difficulty Adaptation

The goal of skill-based difficulty adaptation is to tailor the difficulty level of the game, in an evolving manner, to match the player's skill level. Two common approaches to

implementing this are to adjust the behavior of AI-controlled characters and to adjust the layout of the game map (e.g., by adding or removing obstacles in a racing game or side-scroller). The techniques used to control these adjustments include weighted behavior selection, reinforcement learning (RL), and artificial evolution.

Weighted behavior selection techniques select suitable AI behavior or content from a library of predefined templates according to an online-adapted weight vector. Dynamic scripting (DS) [Spronck 06] is an example of a weighted behavior selection technique that has demonstrated to balance the competitiveness of the AI to different skill levels in *Neverwinter Nights* [Spronck 04]. The idea is to dynamically build scripts by probabilistically selecting rules from a rule library after each combat. The failure or success of a script in combat results, respectively, in an increase or decrease in the selection probabilities of the rules used. In other words, if a script is successful, then the rules from that script will be more likely to be selected in the next fight. The rule library that DS operates on is typically constructed manually, although some evolutionary algorithms can be used to enrich it automatically [Ponsen 05; Ponsen 07].

RL [Sutton 98] algorithms can be used to fine-tune AI behavior by rewarding beneficial actions and punishing detrimental or failed ones. Depending on the defined reward function, the resulting system may adapt to make NPCs more competitive or more collaborative. For example, Andrade et al. [Andrade 05] used RL techniques to learn the effectiveness (i.e., the amount of damage inflicted) of each action in one-on-one fighting games, such as Ultra Street Fighter IV [Capcom 14]. Using this analysis, when faced with a particular action from the player, the AI responds not with the best action, but rather with one that has an equal chance to beat/be defeated by the player's action. The authors showed that the technique provides manageable challenge to simulated players of different skill levels. Although it was only evaluated with an experimental fighting game, the technique seems promising for adoption in commercial fighting games.

Artificial evolution techniques use population-based evolutionary algorithms [Simon 13] to adaptively create entities that optimize some fitness function. This is through reproduction, mutation, recombination (crossover), and selection of individual entities in the population. When used for skill-based adaptation, the entities are modifications to the game, while the fitness function indicates the quality of each individual.

The general process starts with an initial population of candidate entities and repeatedly applies the following operators:

1. *Reproduction*: Breed new individuals by random recombination and mutation.
2. *Selection*: Retain best-fit individuals and discard worst fits.

When used to adjust the difficulty/challenge of a game, these techniques encode the desired quality of the adapted entities as the (multiobjective) fitness function used for selection.

Togelius et al. [Togelius 07] proposed a multiobjective evolutionary algorithm called *cascading elitism* to build racing tracks that optimize three *fun* criteria: (1) provide the right amount of challenge on average [Malone 80], (2) provide a varying amount of challenge [Koster 13], and (3) provide sections that encourage fast driving (such as straight lines). First, each track is represented as a vector of 30 numbers, encoding the control points of 30 connected Bezier curves with the tail segment attached to the start to form a closed shape. This track is then modified using an evolutionary process called

cascading elitism that seeks to optimize three fitness functions to match different aspects of the observed human's behavior:

1. Total progress (i.e., the number of waypoints passed within 1500 time steps)
2. The speed at which each waypoint was passed
3. The deviation of direction

Representing each ANN candidate as a vector of weight parameters in a generation of individuals, cascading elitism undergoes N steps of selection ($N = 3$ in Togelius' work), each of which selects the individuals that perform best on one of the fitness functions. Specifically, out of a population of 100 genomes with different ANN configurations, the 50 genomes that score highest on the first fitness function are selected first. Next, the 30 genomes that score highest on the second fitness function (out of the 50 from the first step) are picked. Finally, the 20 genomes that score highest on the third fitness function are selected. These surviving 20 individuals are then copied five times each and then mutated into the next generation.

39.3.2 Emotion-Based Adaptation

The goal of emotion-based adaptation is to tailor the game according to the current detected affective state of the player. As such, many of these techniques involve the use of some sensor to automatically record physiological responses of players [Yannakakis 08; Yannakakis 10; Tognetti 10; Ambinder 11], while others try to infer the players' currently experienced emotion based on their behavior in games [Pedersen 09; Shaker 10; Booth 09]. Recently, with the prominence of game consoles coming with some sort of motion-sensing input devices (Kinect in Xbox One or PlayStation Camera in PlayStation 4), there have been works devoted to affect modeling using video data as well [Zeng 09; Kleinsmith 13; Meng 14].

Given the player sentiment data, there are generally two approaches in constructing adaptation rules: the first being via heuristics, possibly informed by psychological facts and theories [Booth 09], and the second resulted from offline study on playtesters. Works following the second approach usually generate rules using supervised learning algorithms, such as ANNs, to model the relationship between game parameters/player in-game behavior and emotion indices. Emotion indices are informed by either postsurvey self-reports [Yannakakis 06; Pedersen 09; Shaker 10] or physiological data recorded from biofeedback devices [Tognetti 10; Yannakakis 10; Ambinder 11]. The knowledge is then used to modulate the game while play so as to evoke author-intended sentiments from players. We will focus on the adaptation techniques, rather than affect modeling techniques, so while we will touch on the latter for a better understanding of the adaptation approaches adopted, surveying emotion modeling is beyond the scope of this section.

Left 4 Dead [EA 08], a postapocalyptic zombie-themed survival game published by Electronic Arts and Valve in 2008, is among the first AAA titles that publicly report their use of players' inferred affective state to adapt gameplay [Booth 09]. In particular, the system keeps track of a measure called the survivor intensity (SI) that associates with each player's *emotional intensity* when battling zombies. The SI value is updated according to heuristic rules. For example, a heuristic rule indicates that SI value increases when player receives damage (e.g., injured by zombies, incapacitated, or pushed off of a ledge)

and decays over time otherwise. It is, therefore, implicitly assumed that players experience more trauma and anxiety when taking more damage in this game setting. Taking SI as input, the adopted adaptive dramatic pacing algorithm follows a cyclic finite-state machine (FSM) to spawn zombies. The FSM consists of four states, the transitions between which are determined by SI values:

1. *Build up*: Continuously spawning zombies until SI exceeds a peak value
2. *Sustain peak*: Trying to maintain SI peak for 3–5 s
3. *Peak fade*: Imposing minimal threat until SI falls below peak range
4. *Relax*: Continuing with minimal threat for 30–45 s or until players have traveled far enough

While SI is arguably tied to players' skill level (i.e., the more skilled they are, the harder for SI to increase over time) and quantities of zombies to game difficulty, the fact that the game does not blindly increase or decrease its difficulty according to the exhibited skillfulness but follows a *drama peak* principle makes it more of an emotion-based adaptive system than a skill-based one. After all, the algorithm's goal is to keep players entertained and not overly exhausted with fighting zombies instead of maintaining the challenge level.

Additionally, Valve has reportedly started looking at biofeedback input to inform gameplay adaptation for their future titles [Ambinder 11]. Specifically, the team has conducted experiments to assess the effectiveness of adaptation based on physiological signals using Left 4 Dead 2's (L4D2) platform. While there are many input possibilities such as heart rate, skin conductance level (SCL), electroencephalography, and body temperature, the experiment focuses on indexing player's arousal based on SCL information. The authors modified the AI Director in L4D2 to take SCL-based arousal values instead of heuristic rules based on SI values and let playtesters try both versions. Preliminary post-surveys showed that adapting the gameplay using biofeedback data has improved both the participants' fun factor and sense of challenge. The team subsequently conducted similar experiments with Alien Swarm [Valve 10], which showed similar positive results. The team, however, acknowledged some major hurdles before such techniques become practical. First, because biofeedback signals vary from player to player, the adaptation algorithm requires a lot of tweaking to work desirably. Second, most state-of-the-art methods for collecting physiological data are intrusive to the gaming process, as they require additional wiring or device attachment to the player's body. Finally, it could be hard to associate the stimuli with in-game events due to a known delay in bodily response. That said, there have been some attempts in releasing commercial gaming systems that include physiological signal recorders as part of the package. One notable example is Wild Divine with their Iom, a USB-based biofeedback reader [Wild Divine 14]. The device attaches to the three fingers of the player to measure her heart rate and SCL, which are used as input to meditation game series named The Journey to Wild Divine.

In the academia, Tognetti et al. [Tognetti 10] showed that by using a biofeedback device, namely ProComp Infiniti device by Thought Technology, more objective measures based on biological signals can be achieved, analyzed, and used for adaptation. In the work, they propose to use *linear discriminant analysis* to model player experience from recorded physiological responses such as blood-volume pulse (BVP), electrocardiogram, galvanic skin response, respiration, and temperature. The technique yields comparable

performance to other modeling techniques [Yannakakis 08] while incurring lower computation overhead. Additionally, the technique is demonstrated in The Open Racing Car Simulator [Torcs 07], an open-source racing game platform used mainly for research.

Changing the player's view in the game world is another approach to influencing their affective state [Yannakakis 10]. The idea is to first train a predictive model that maps each configuration of camera control parameters and biosignal features to an affective state value. The investigated biosignal features are BVP, skin conductance, and heart rate, recorded using an IOM biofeedback device, a highly unobtrusive device. Predictive models used are based on ANNs trained using an evolutionary algorithm with the goal of matching players' reported emotional preferences and the ANN output. Next, these models are then evaluated and adjusted to satisfaction before being integrated into the content generator as a fitness function to predict the effect of candidate outputs. The technique was evaluated in a test 3D predator/prey game.

39.3.3 Style-Based Adaptation

Style-based adaptation techniques aim at tailoring the game experience to fit the preferred play style of players. Unlike skills that can be used to compare players' expertise with the game, that is, their chance in beating AI NPCs or other fellow players, play styles refer to in-game behaviors that differ due to personal choice or interests. As such, players with similar experience and skill levels can exhibit very different styles in interacting with the game world. For instance, in Assassin's Creed [Ubisoft 07], players can either exterminate opponent NPCs in a high-profile noisy fight or assassinate them silently. While the game promotes low-key stealth behavior, either style choice is valid for advancing in the game.

It is worth noting that many skill-based techniques can be adapted for use with the style-based adaptation approach by appropriately changing the evaluation functions in finding best-suited candidates [Lopes 11]. Targets for style-based adaptation usually include game content [Thue 07] or interface components [Magerko 08].

In order to deliver players a highly personalized story-based experience, Thue et al. [Thue 07] propose to decompose the gameplay into story events or *encounters*, which are selected appropriately from a library at run time based on the observed play style of the player. Each encounter comes with a set of preconditions for happening and a set of *branches* or available actions that players can take in that situation. In commercial games whereby a main storyline needs to be in place to reflect the overarching story of the game, these encounters that can be flexibly presented to players can be thought of as side quests, that is, optional play that potentially increases fun and replayability. The encounter selection system, namely, PaSSAGE, takes as input a player model that fuzzily categorizes players as one of five archetypes according to Robin's laws [Laws 02]: Fighters (combat inclined), Power Gamers (item collectors), Tacticians (creative thinkers), Storytellers (who like complex plots), and Method Actors (who likes to connect the in-game drama to their own selves). This model is constructed in real time based on predefined recognition rules and their in-game behavior. Next, PaSSAGE consults a library of encounters, each of which has been annotated to appeal to specific play styles, to find the one with a story branch that fits the current player model the most. The selected encounter then goes through a refinement phase in which various parameters are determined so that its instantiation is consistent to the modeled play style and current story line. Finally, the selected encounter is

actualized when its preconditions (such as the appearance of required actor types) are satisfied. For instance, when an encounter of exchanging conversation with a friendly trader is selected for a Power Gamer player type, the system will populate the dialog with lines that mention locations for valuable items (which preferred by Power Gamers) and waits until a trader is nearby before conducting the encounter. In contrast, the same encounter will inform Fighter types of where to find monster hoards instead. PaSSAGE was evaluated with 90 university students by letting them play a scenario of the *Red Riding Hood* story, built using the *Aurora Neverwinter Toolset*. Participants reported to perceive more fun and sense of agency in playing the adaptive scenario over the fixed one.

In a similar manner, Magerko et al. [Magerko 08] advocated for generating highly personalized game interfaces in learning games based on the detected player–learner type, be it Explorer, Achiever (interested in quickly collecting awards/badges), or Winner (seeking to complete the level thoroughly). In particular, Explorers are presented with bonuses and trivia, but less information on how they are performing, such as leader board or timer. Winners and Achievers on the other hand are shown the leader board more frequently, with Winners having the option to access trivia questions or tutorials, while Achievers the timer.

39.4 Game Analytics and Recommendation Systems

Reaching beyond the scope of personalizing a single game, many publishers are interested and in fact already striving, for example, Steam [Orland 10] and PlayStation Store [Lempel 10], to personalize the gaming experience of visitors/members of their game networks by introducing players to games that they are likely to enjoy. As such, the goal of these recommendation systems is to apply analytics technologies to player's behavior and build preference models that can inform the recommendation process.

The idea is inspired by major Internet and e-commerce websites, such as Google News [Das 07], Amazon [Linden 03], or Netflix [Amatriain 12], which suggest new products to site visitors based on their purchase and viewing history. Recommendation systems [Schafer 99] have been used in many prominent game release networks, such as PlayStation Store [Lempel 10] and Steam [Orland 10], to suggest titles that players are more likely to enjoy, but the input for these algorithms is not the players' in-game behavior, but their preferences expressed out of game (such as their ratings of played titles). The goal of these automated recommendation systems is to predict, among yet-to-play titles, which game a specific player with some history of played games will enjoy the most. While automated recommendation is a popular research topic [Bobadilla 13], we would like to highlight a technique that is reportedly behind the success of Amazon [Linden 03; Mangalinda 12] in marketing highly relevant products to customers. The algorithm belongs to the class of item-based collaborative filtering (CF) algorithms [Sarwar 01; Linden 03; Su 09].

CF is based on the observation that people who have behaved similarly in the past tend to act similarly in the same setting. This observation extends to purchase behavior on products (a game or some movie). This means we can deduce the rating of user *X* on product A from that of user *Y*, who has a similar rating history on similar or same products and who has already rated product A. A predicted positive rating entails that *X* has a high chance of enjoying A (i.e., will probably rate A positively after purchasing it),

thus suggesting A to *X* is worthwhile [Su 09]. This approach, *user-based CF*, can be summarized as a two-step process:

1. Compute the similarity in rating behavior between users.
2. Given a product A, approximate *X*'s rating of A based on those of the users' most similar to *X*.

Product A is then suggested to *X* if *X*'s predicted rating for A is sufficiently high. The metric function to compute similarity in step 1 is often chosen to be the cosine of the angle between two rating vectors of the involved users, that is,

$$\text{Similarity}\left(\vec{X},\vec{Y}\right) = \cos\left(\vec{X},\vec{Y}\right) = \frac{\vec{X}\cdot\vec{Y}}{\left\|\vec{X}\right\| * \left\|\vec{Y}\right\|} \tag{39.1}$$

in which $\vec{X},\vec{Y}$ are respective rating vectors of user *X* and *Y* on all items in the item catalogue.

Unfortunately, this user-based approach does not scale well with the number of customers and items, as it has to examine every combination of customers and item ratings during the training phase (step 1). In order to overcome this problem, Amazon [Linden 03], and subsequently Steam [Orland 10], adopted an item-based variant of CF, which builds similarity relationships between items, instead of users. The algorithm *item-to-item CF* replaces the two steps in user-based CF with the following:

1. Compute similarity in rating scores between items.
2. Given a user *X*, approximate *X*'s rating of product A based on ratings of *X* on items that are similar to A.

In other words, if we combine the ratings of users on catalogued items as a matrix with columns being items and rows being users, user-based CF operates row-wise, while item-to-item CF operates column-wise. Because the similarity metric now works on items' rating vectors, users who never rate anything they purchase can still be recommended suitable items as long as there are ratings on pertinent items from other fellow shoppers who purchased similar items.

With the recent explosive growth of social networks (such as Facebook and Twitter), players now have many channels to voice their preferences. As such, many game publishers such as Steam have incorporated data gleaned from these sources in their recommender systems [Orland 10; Williams 13].

39.5 Game Analytics and Team Matching Techniques

In many games that feature an online player versus player game mode, such as those in the *multiplayer online battle arena* genre [Ryan 12] or the *real-time strategy game* genre [Dor 14], the fun factor no longer lies solely in the hands of game designers. While game mechanics and aesthetics still need to be sufficiently rich, it is the human factor and social interactions in those virtual matches that make playing exciting or boring [Madigan 13]. If games such as DotA or LoL are going to retain players over the long run, it is crucial for them to match teams up appropriately.

We surveyed the matchmaking systems in DotA [Dota Team 13], LoL [Zileas 09; Pereira 13; Lefebvre 14], and *CS:GO* [Totilo 11; GO 13; Rose 14]. In all of these games, the common approach is to

- Represent players' skill level using an aggregating score system. Examples include the matchmaking rating (MMR) in DotA, the Elo rating system in LoL (seasons 1 and 2) and CS:GO's initial launch, the League Point and Matchmaking Rating in LoL (season 3), and the Skill Groups in CS:GO (current). Players' scores are updated after finishing each battle
- Elicit additional matchmaking constraints from players, for example, LoL's Team Builder [Pereira 13; Lefebvre 14] allows players to join specific queues for the specific champions (in-game characters) that they want to play
- Rule-based heuristics to match players with similar skill scores while satisfying the players' matchmaking conditions

While differing in their aggregating formulae, the adopted score systems always take into account performance-based metrics, such as win/loss statistics, at the team level and performance statistics, such as kill/death, at the individual level [Dota Team 13; Setola 13]. Score update rules are designed such that beating stronger opponents yields a larger score increase, while losing to weaker opponents yields a larger score decrease.

The key goal that drives the design of matchmaking rules is to form teams that are equally balanced, both internally and externally. The assumption is that players enjoy fighting against others with similar skill level because of the uncertainty in the battle outcome and that they prefer to play with others at a similar skill level so that they will feel needed, but not like they're having to hold the team up [Zileas 09; Dota Team 13]. For instance, to achieve such a balance, DotA 2 adjusts the involved players' MMRs before starting the match by (1) boosting the MMRs of all players in low-ranked teams to match the opponents' average MMR and (2) reducing the MMR difference gap among team members in the process. LoL does something similar in that team members' ratings are also boosted to more balanced state before starting a match, but refuses to provide more details on these trade secrets [Zileas 09]. Note that this is only an example and there could be other ways to achieve skill balance when matching teams, which are often classified information to avoid players from gaming the system.

39.6 Conclusion

In this chapter, we discussed analytic approaches for developing player models using individual and communal analysis. We then illustrated, using both academic and industry examples, how the resulting player models can be used to adjust the gaming experience and increase engagement. We gave examples for adapting various aspects of the game based in the modeled skill level, affective state, and play style. Lastly, we analyzed in detail the techniques used behind many real-world systems for product recommendation and team matching.

There are several areas that we see as opportunities for growth in leveraging game analytics to enhance players' experiences. In particular, recommendation systems do not leverage player styles and temporal analysis of their game behaviors; instead, the techniques adopted by the industry have been approaches used by Amazon and Netflix, which

do not tend to have depth of behavioral data to stimulate its decisions. It should be possible, for example, to leverage game analytics and collective intelligence algorithms to make better predictive algorithms for recommendations based on behavior data and game buying behavior in addition to review data.

Another area that has received very little attention is the development of adaptive algorithm to adjust content based on interest or play styles. While there has been much work on modeling players' interest, styles, and personality in regard to behaviors and while there have been designers who have advocated for more personalized designs leveraging personality theory [VandenBerghe 12], there has been very little work in procedural content personalization based on style. This is an area that deserves more attention and can lead to better and more sustained engagement, especially for social games that currently do not have high sustained engagement [Playnomics 12].

In conclusion, we hope that this chapter has provided readers with a broad overview of an area that is currently up and coming in both research and industry works. We also hope that the discussion has sparked some interest in using these algorithms in more creative and innovative ways to advance current game designs and AI systems in games.

References

[Andrade 05] Andrade, G., G. Ramalho, H. Santana, and V. Corruble. 2005. Extending reinforcement learning to provide dynamic game balancing. In *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence* (*IJCAI*), Edinburgh, Scotland, UK, pp. 7–12.

[Amatriain 12] Amatriain, X. and J. Basilico. 2012. Netflix recommendations: Beyond the 5 stars. Netflix. http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html (accessed August 18, 2014).

[Ambinder 11] Ambinder, M. 2011. Biofeedback in gameplay: How valve measures physiology to enhance gaming experience. In *Game Developers Conference,* San Francisco, CA.

[Atari 02] Atari. 2002. Neverwinter nights. http://www.ign.com/games/neverwinter-nights/pc-12077 (accessed August 18, 2014).

[Bethesda 08] Bethesda Softworks. 2008. Fallout 3. http://www.ign.com/games/fallout-3/xbox-360-882301 (accessed August 18, 2014).

[Bethesda 10] Bethesda Softworks. 2010. Fallout: New Vegas. http://www.ign.com/games/fallout-new-vegas/pc-14341979 (accessed August 18, 2014).

[Bishop 06] Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*, Vol. 1. New York: Springer.

[Bobadilla 13] Bobadilla, J., F. Ortega, A. Hernando, and A. Gutiérrez. 2013. Recommender systems survey. *Knowledge-Based Systems* 46: 109–132.

[Booth 09] Booth, M. 2009. The AI systems of left 4 dead. In *Keynote, Fifth Artificial Intelligence and Interactive Digital Entertainment Conference* (*AIIDE 09*), Palo Alto, CA.

[Burges 98] Burges, C. J. C. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* 2(2): 121–167.

[Capcom 09] Capcom. 2009. Resident evil 5. http://www.ign.com/games/resident-evil-5/xbox-360-760880 (accessed August 18, 2014).

[Capcom 14] Capcom. 2014. Ultra street fighter IV. http://www.ign.com/games/ultra-street-fighter-4/xbox-360-20002351 (accessed August 18, 2014).

[Cutler 94] Cutler, A. and L. Breiman. 1994. Archetypal analysis. *Technometrics* 36(4): 338–347.

[Das 07] Das, A. S., M. Datar, A. Garg, and S. Rajaram. 2007. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web,* Banff, Alberta, Canada, pp. 271–280.

[Dor 14] Dor S. 2014. A history of real-time strategy gameplay from decryption to prediction: Introducing the actional statement. http://www.kinephanos.ca/2014/real-time-strategy/ (accessed August 18, 2014).

[Dota Team 13] Dota Team. 2013. Matchmaking. http://blog.dota2.com/2013/12/matchmaking/ (accessed August 18, 2014).

[Drachen 12] Drachen, A., R. Sifa, C. Bauckhage, and C. Thurau. 2012. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *2012 IEEE Conference on Computational Intelligence and Games* (*CIG*), Granada, Spain, pp. 163–170.

[Drachen 13] Drachen, A., C. Thurau, R. Sifa, and C. Bauckhage. 2013. A comparison of methods for player clustering via behavioral telemetry. *Foundations of Digital Games* 2013: 245–252.

[EA 08] Electronic Arts. 2008. Left 4 dead. http://www.ign.com/games/left-4-dead/xbox-360-875936 (accessed August 18, 2014).

[EA 10] Electronic Arts. 2010. Battlefield: Bad company 2. http://www.ign.com/games/battlefield-bad-company-2/xbox-360-14293277 (accessed August 18, 2014).

[GO 13] GO. 2013. Competitive skill groups FAQ. http://blog.counter-strike.net/index.php/2012/10/5565/ (accessed August 18, 2014).

[Jolliffe 02] Jolliffe, I. T. 2002. *Principal Component Analysis*. New York: Springer.

[Kleinsmith 13] Kleinsmith, A. and N. Bianchi-Berthouze. 2013. Affective body expression perception and recognition: A survey. *IEEE Transactions on Affective Computing* 4(1): 15–33.

[Koster 13] Koster, R. 2013. *Theory of Fun for Game Design*. Sebastopol, CA: O'Reilly Media, Inc.

[Laws 02] Laws, R. D. 2002. *Robin's Laws of Good Game Mastering*. Austin, TX: Steve Jackson Games.

[Lee 99] Lee, D. D. and H. S. Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755): 788–791.

[Lefebvre 14] Lefebvre, E. 2014. League of legends introduces the new team builder queue. http://massively.joystiq.com/2014/03/27/league-of-legends-introduces-the-new-team-builder-queue/ (accessed August 18, 2014).

[Lempel 10] Lempel, E. 2010. Next PS3 firmware update adds playstation store recommendations. Sony Network Entertainment. http://blog.us.playstation.com/2010/07/26/next-ps3-firmware-update-adds-playstation-store-recommendations/ (accessed August 18, 2014).

[Linden 03] Linden, G. D., B. Smith, and J. York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7(1): 76–80.

[Lopes 11] Lopes, R. and R. Bidarra. 2011. Adaptivity challenges in games and simulations: A survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3(2): 85–99.

[MacQueen 67] MacQueen, J. B. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, Berkeley, CA, Vol. 1, pp. 281–297.

[Madigan 13] Madigan, J. 2013. Modifying player behavior in League of Legends using positive reinforcement. http://www.gamasutra.com/view/news/184806/Modifying_player_behavior_in_League_of_Legends_using_positive_reinforcement.php (accessed August 18, 2014).

[Magerko 08] Magerko, B., C. Heeter, J. Fitzgerald, and B. Medler. Intelligent adaptation of digital game-based learning. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share,* Toronto, Ontario, Canada, pp. 200–203.

[Malone 80] Malone, T. W. 1980. What makes things fun to learn? Heuristics for designing instructional computer games. In *Proceedings of the Third ACM SIGSMALL Symposium and the First SIGPC Symposium on Small Systems*, New York, pp. 162–169.

[Mangalinda 12] Mangalinda, J. 2012. Amazon's recommendation secret. http://fortune.com/2012/07/30/amazons-recommendation-secret/ (accessed August 18, 2014).

[Meng 14] Meng, H. and N. Bianchi-Berthouze. 2014. Affective state level recognition in naturalistic facial and vocal expressions. *IEEE Transactions on Cybernetics* 44(3): 315–328.

[Missura 09] Missura, O. and T. Gärtner. 2009. Player modelling for intelligent difficulty adjustment. In *Proceedings of the 12th International Conference on Discovery Science,* Porto, Portugal, pp. 197–211.

[Neapolitan 04] Neapolitan, R. E. 2004. *Learning Bayesian Networks*, Vol. 38. Upper Saddle River, NJ: Prentice Hall.

[Newell 08] Newell, G. 2008. Gabe Newell writes for edge. http://www.edge-online.com/features/gabe-newell-writes-edge/ (accessed August 18, 2014).

[Orland 10] Orland, K. 2010. Valve launches game recommendation feature for steam. Gamasutra. http://www.gamasutra.com/view/news/122261/Valve_Launches_Game_Recommendation_Feature_For_Steam.php (accessed August 18, 2014).

[Pedersen 09] Pedersen, C., J. Togelius, and G. N. Yannakakis. 2009. Modeling player experience in Super Mario Bros. In *IEEE Symposium on Computational Intelligence and Games* (*CIG*), Milan, Italy, pp. 132–139.

[Pereira 13] Pereira, C. 2013. League of legends' team builder lets you preselect your role. http://www.ign.com/articles/2013/10/15/league-of-legends-team-builder-lets-you-preselect-your-role (accessed August 18, 2014).

[Piggyback 09] Piggyback. 2009. *Resident Evil 5: The Complete Official Guide.* Roseville, CA: Prima Publishing.

[Playnomics 12] Player Engagement Study Q3. 2012. Playnomics report. http://www.insidesocialgames.com/wp-content/uploads/2012/10/Playnomics_Q3-report_Final-copy.pdf (accessed August 19, 2014).

[Ponsen 05] Ponsen, M. J. V., H. Muñoz-Avila, P. Spronck, and D. W. Aha. 2005. Automatically acquiring domain knowledge for adaptive game AI using evolutionary learning. In *Proceedings of the National Conference on Artificial Intelligence,* Pittsburgh, PA 20(3): 1535.

[Ponsen 07] Ponsen, M. J. V., P. Spronck, H. Muñoz-Avila, and D. W. Aha. 2007. Knowledge acquisition for adaptive game AI. *Science of Computer Programming* 67(1): 59–75.

[Riot 09] Riot Games. 2009. League of legends. http://www.ign.com/games/league-of-legends/pc-14287819 (accessed August 18, 2014).

[Rose 14] Rose, M. 2014. How counter strike global offensive turned itself around. http://kotaku.com/how-counter-strike-global-offensive-turned-itself-arou-1575373000 (accessed August 18, 2014).

[Ryan 12] Ryan, C. 2012. Multiplayer online battle arena explained. http://www.alteredgamer.com/pc-gaming/43646-multiplayer-online-battle-arenas-and-dota-explained/ (accessed August 18, 2014).

[Sarwar 01] Sarwar, B., G. Karypis, J. Konstan, and J. Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web,* Hong Kong, pp. 285–295.

[Schafer 99] Schafer, J. B., J. Konstan, and J. Riedl. Recommender systems in e-commerce. 1999. In *Proceedings of the First ACM Conference on Electronic Commerce,* Denver, CO, pp. 158–166.

[Seif El-Nasr 13] Seif El-Nasr, M., A. Drachen, and A. Canossa. 2013. *Game Analytics: Maximizing the Value of Player Data*. New York: Springer.

[Setola 13] Setola, H. 2013. Valve's counter-strike "Global Offensive" is a failing strategy. http://www.gizorama.com/feature/opinion/valves-counter-strike-global-offensive-is-a-failing-strategy (accessed August 18, 2014).

[Shaker 10] Shaker, N., G. N. Yannakakis, and J. Togelius. 2010. Towards automatic personalized content generation for platform games. In *The Sixth AAAI Conference on Artificial Intelligent and Interactive Digital Environment* (*AIIDE 10*), Stanford, CA.

[Simon 13] Simon, D. 2013. *Evolutionary Optimization Algorithms*. Hoboken, NJ: John Wiley & Sons.

[Spronck 04] Spronck, P., I. Sprinkhuizen-Kuyper, and E. Postma. 2004. Difficulty scaling of game AI. In *Proceedings of the Fifth International Conference on Intelligent Games and Simulation* (*GAME-ON 2004*), EUROSIS, Belgium, pp. 33–37.

[Spronck 06] Spronck, P., M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma. 2006. Adaptive game AI with dynamic scripting. *Machine Learning* 63(3): 217–248.

[Su 09] Su, X. and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence* 2009: 4.

[Sutton 98] Sutton, R. S. and A. G. Barto. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

[Tang 09] Tang, H., C. H. Tan, K. C. Tan, and A. Tay. 2009. Neural network versus behavior based approach in simulated car racing game. In *IEEE Workshop on Evolving and Self-Developing Intelligent Systems, 2009* (*ESDIS'09*), Nashville, TN, pp. 58–65.

[Thue 07] Thue, D., V. Bulitko, M. Spetch, and E. Wasylishen. 2007. Interactive Storytelling: A player modelling approach. In *Proceedings of the Third AAAI Conference on Artificial Intelligent and Interactive Digital Environment* (*AIIDE 07*), Stanford, CA, AAAI Press, pp. 43–48.

[Thurau 10] Thurau, C., K. Kersting, and C. Bauckhage. 2010. Yes we can: simplex volume maximization for descriptive web-scale matrix factorization. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management,* Toronto, Canada, pp. 1785–1788.

[Togelius 07] Togelius, J., Nardi, R. D., and Lucas, S. 2007. Towards automatic personalised content creation for racing games. In *Proceedings of IEEE Symposium on Computational Intelligence and Games (CIG)*, pp. 252–259.

[Tognetti 10] Tognetti, S., M. Garbarino, A. Bonarini, and M. Matteucci. 2010. Modeling enjoyment preference from physiological responses in a car racing game. In *Proceedings of IEEE Symposium on Computational Intelligence and Games* (*CIG*), Copenhagen, Denmark, pp. 321–328.

[Torcs 07] Torcs. 2007. The open racing car simulator. http://torcs.sourceforge.net/ (accessed August 18, 2014).
[Totilo 11] Totilo, S. 2011. An hour with counter-strike: GO. http://kotaku.com/5834542/an-hour-with-counter-strike-go (accessed August 18, 2014).
[Ubisoft 07] Ubisoft. 2007. Assassin's creed. http://assassinscreed.ubi.com/en-US/home/index.aspx (accessed August 18, 2014).
[Valve 09] Valve. 2009. Left 4 dead 2. http://www.ign.com/games/left-4-dead-2/xbox-360-14352241 (accessed August 18, 2014).
[Valve 10] Valve. 2010. Alien swarm. http://store.steampowered.com/app/630/ (accessed August 18, 2014).
[Valve 12] Valve. 2012. Counter-strike: Global offensive. http://www.ign.com/games/counter-strike-global-offensive/mac-116695 (accessed August 18, 2014).
[Valve 13] Valve. 2013. DotA 2. http://www.ign.com/games/dota-2/mac-89236 (accessed August 18, 2014).
[VandenBerghe 12] VandenBerghe, J. 2012. The 5 domains of play: Applying psychology's big 5 motivation domains to games. Presentation at the *Game Developer Conference*, San Francisco, CA.
[Wild Divine 14] Wild Divine. 2014. Wild divine website. http://www.wilddivine.com/ (accessed August 18, 2014).
[Williams 13] Williams, R. 2013. Valve launches beta for steam reviews, allowing you to share your recommendations with the world. http://hothardware.com/News/Valve-Launches-Beta-for-Steam-Reviews-Allowing-You-to-Share-Your-Recommendations-With-the-World/ (accessed August 18, 2014).
[Yannakakis 05] Yannakakis, G. N. and M. Maragoudakis. 2005. Player modeling impact on player's entertainment in computer games. In *User Modeling 2005*, eds. L. Ardissono, P. Brna, and A. Mitrovic, pp. 74–78. Berlin, Germany: Springer.
[Yannakakis 06] Yannakakis, G. N. and J. Hallam. 2006. Towards capturing and enhancing entertainment in computer games. In *Advances in Artificial Intelligence*, eds. G. Antoniou, G. Potamias, C. Spyropoulos, and D. Plexousakis, pp. 432–442. Berlin, Germany: Springer.
[Yannakakis 08] Yannakakis, G. N. and J. Hallam. 2008. Entertainment modeling through physiology in physical play. *International Journal of Human-Computer Studies* 66(10): 741–755.
[Yannakakis 10] Yannakakis, G. N., H. P. Martínez, and A. Jhala. 2010. Towards affective camera control in games. *User Modeling and User-Adapted Interaction* 20(4): 313–340.
[Yannakakis 13] Yannakakis, G. N., P. Spronck, D. Loiacono, and E. André. 2013. Player modeling. In *Artificial and Computational Intelligence in Games,* S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, Eds., *DFU* Vol. 6, Dagstuhl, Germany, pp. 45–59.
[Zeng 09] Zeng, Z., M. Pantic, G. I. Roisman, and T. S. Huang. 2009. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(1): 39–58.
[Zileas 09] Zileas. 2009. LOL matchmaking explained. http://forums.na.leagueoflegends.com/board/showthread.php?t=12029 (accessed August 18, 2014).