Modeling Player Experience for Content Creation

Christopher Pedersen, Julian Togelius *Member, IEEE,* and Georgios N. Yannakakis, *Member, IEEE*

***Abstract*—In this paper, we use computational intelligence techniques to built quantitative models of player experience for a platform game. The models accurately predict certain key affective states of the player based on both gameplay metrics that relate to the actions performed by the player in the game, and on parameters of the level that was played. For the experiments presented here, a version of the classic Super Mario Bros game is enhanced with parameterizable level generation and gameplay metrics collection. Player pairwise preference data is collected using forced choice questionnaires, and the models are trained using this data and neuro-evolutionary preference learning of multi-layer perceptrons. The derived models will be used to optimize design parameters for particular types of player experience, allowing the designer to automatically generate unique levels that induce the desired experience for the player.**

Keywords: Platform games, player satisfaction modeling, content creation, fun, player experience, preference learning, neuroevolution.

I. Introduction

NUMEROUS theories exist regarding what makes computer games fun, as well as which aspects contribute to other types of player experience such as challenge, frustration and immersion [1], [2], [3], [4], [5]. These theories have originated in different research fields and in many cases independently of each other (however, there is substantial agreement on several counts, e.g. regarding the importance of *challenge* and *learnability* for making a game fun). While useful high-level guidance for game design, none of these theories is quantitative — derived models of player experience are not mathematical expressions — and they tend to apply to games in general rather than specific aspects of specific games. This means that if we want to develop algorithms that design or adapt games (or aspects of games) *automatically*, we have to make several auxiliary assumptions in order to achieve the necessary specificity and preciseness of our models.

It seems clear that we need empirical research on particular games to acquire such models. Recently, research in player satisfaction modeling has focused on empirically measuring the effects on player experience of changing various aspects of computer games, such as non-player character (NPC) playing styles in the Pac-Man game [6]. Similarly, efficient quantitative models of player satisfaction have been constructed using in-game data, questionnaires and physiological measurements in augmented-reality games [7].

At the same time, a parallel research direction aims to find methods for automatically generating entertaining game content. Automatic (or *procedural*) content generation is likely to be of great importance to computer game development in the future; both offline, for making the game development process more efficient (design of content such as environments and animations now consume a major part of the development budget for most commercial games) and online, for enabling new types of games based on player-adapted content. These efforts see some aspect of a game as variable, define a fitness ("goodness") function based on a theory of player satisfaction, and use a learning or optimization algorithm to change the variable aspect of the game so as to become more "fun" according to some definition. The literature on this is so far scarce, as it is a new research direction. The aspects of games that have been considered for optimization include:

- Environments, such as tracks for racing games [8], [9] and levels for platform games [10], [11].
- Narrative [12], [13].
- Rules for board games [14], [15] and Pac-Man-like games [16].
- Camera control parameters, such as distance, height and frame coherence [17], [18].
- Help functions [19] and various gameplay elements [20] in intelligent tutoring games.

What most of the above studies have in common is that the fitness or cost functions used for optimization have been somewhat arbitrary, in that they have been based on intuition in combination with some qualitative theory of player experience. Optimization of game aspects based on empirically derived models have so far been limited to parameters for NPC behavior [6] and high-level game parameters [21]. To the best of our knowledge, game content such as rules or environments has not been generated based on empirically derived models.

We consider modeling of player experience as a whole to be of utmost importance for making automatic content generation techniques more sophisticated and usable. The work we describe in this paper is novel in that computational models of player experience are constructed which are derived from gameplay interaction and can be used as fitness functions for game content generation. For that purpose, we use a modified version of a classic platform game for our experiments and collect player data through the Internet.

In the following, we describe the game used for our experiments; which player interaction data was collected and how; the preference learning method we used to construct player experience models; how feature selection was used to reduce the number of features used in the model; results of an initial statistical analysis; results of training nonlinear perceptrons to approximate the functions mapping between selected gameplay and controllable features, and aspects of player experience; and the result of optimizing the architecture of multi-layer perceptrons (MLPs) and furthermore the performance of the derived MLP models. Finally, we discuss how

Authors are with the Center for Computer Games Research, IT University of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark. Emails: {gammabyte, juto, yannakakis}@itu.dk

the induced models will be used for automatically generating game content. This paper significantly extends [22] in which the core ideas of the methodology proposed are outlined, and [23] in which only three affective states are analyzed and only using single-layer perceptrons (SLPs) and less sophisticated feature selection.

The focus and main contribution of this paper is introducing a refined method for player experience modeling, and exemplifying how it can be used for a well-known game. The particular models we arrive at are only meant to be valid for this game, and possibly closely related games, whereas the method could be generalized to games in many different genres as well as for modeling the experience of user-computer interaction in general.

II. Testbed platform game

The test-bed platform game used for our studies is a modified version of Markus Persson's *Infinite Mario Bros* (see Fig. 1) which is a public domain clone of Nintendo's classic platform game *Super Mario Bros*. The original Infinite Mario Bros is playable on the web, where Java source code is also available[1].

The gameplay in Super Mario Bros consists of moving the player-controlled character, Mario, through two-dimensional levels, which are viewed sideways. Mario can walk and run to the right and left, duck, jump, and (depending on which state he is in) shoot fireballs. Gravity acts on Mario, making it necessary to jump over holes (or gaps) to get past them. Mario can be in one of three states: Small (at the beginning of a game), Big (can crush some objects by jumping into them from below), and Fire (can shoot fireballs).

The main goal of each level is to get to the end of the level, which means traversing it from left to right. Auxiliary goals include collecting as many as possible of the coins that are scattered around the level, clearing the level as fast as possible, and collecting the highest score, which in part depends on the number of collected coins and killed enemies.

The presence of gaps and moving enemies are the main challenges of Mario. If Mario falls down a gap, he loses a life. If he touches an enemy, he gets hurt; this means losing a life if he is currently in the Small state, whereas if he is in the Big and Fire state he shifts to Small and Big state respectively. However, if he jumps so that he lands on the enemy from above, the outcome is dependent on the enemy: Most enemies (e.g. goombas, cannonballs) die from this treatment; others (e.g. piranha plants) are not vulnerable to this and proceed to hurt Mario; finally, turtles withdraw into their shells if jumped on, and these shells can then be picked up by Mario and thrown at other enemies to kill them.

Certain items are scattered around the levels, either out in the open, or hidden inside blocks of brick and only appearing when Mario jumps at these blocks from below so that he smashes his head into them. Available items include coins which can be collected for score and for extra lives (every 100 coins), mushrooms which make Mario grow Big if he is currently Small, and flowers which make Mario turn into the Fire state if he is already Big.

[1]http://www.mojang.com/notch/mario/

Fig. 1. Test-bed game screenshot, showing Small Mario jumping over a piece of flat terrain surrounded by two gaps.

No textual description can fully convey the gameplay of a particular game. Only some of the main rules and elements of Super Mario Bros are explained above; the original game is one of the world's best selling games, and still very playable more than two decades after its release in the mid-eighties. Its game design has been enormously influential and inspired countless other games, making it a good experiment platform for player experience modeling.

While implementing most features of Super Mario Bros, the standout feature of Infinite Mario Bros is the automatic generation of levels. Every time a new game is started, levels are randomly generated by traversing a fixed width and adding features (such as blocks, gaps and opponents) according to certain heuristics. In our modified version of Infinite Mario Bros most of the randomized placement of level features is fixed since we concentrate on a few selected game level parameters that affect game experience.

III. Data collection

Before any modeling could take place we needed to collect data to train the model on. We collected three types of data from hundreds of players over the Internet:

1) Controllable features of the game, i.e. the parameters used for level generation, and affecting the type and difficulty of the level. These were varied systematically to make sure all variants of the game were compared.
2) Gameplay characteristics, i.e. how the user plays the game. We measured statistical features such as how often and when the player jumped, ran, died etc. These features cannot be directly controlled by the game as they depend solely on the player's skill and playing style in a particular game level.
3) The player's experience of playing the game, measured through a 4-alternative forced choice questionnaire administered after playing two pairs of games with different controllable features and asking the players to rank the games in order of emotional preference.

Below we describe in detail which features were collected for each type of data.

A. Controllable features

We modified the existing level generator to create levels according to four controllable parameters presented below. Three of these parameters are dedicated to the number, width and placement of gaps. The fourth parameter defines a new function (i.e. game mechanic), the direction switch.

- The number of gaps in the level, G.
- The average width of gaps, $E\{G_w\}$.
- The spatial diversity of gaps which is measured by the entropy of the number of gaps appearing in a number of G (equally-spaced) segments of the level. The entropy of gap-placements H_g in the G segments is calculated and normalized into $[0, 1]$ via (1):

$$H_g = \left[-\frac{1}{logG} \sum_{i=1}^{G} \frac{g_i}{G} log\left(\frac{g_i}{G}\right) \right] \quad (1)$$

  where g_i is the number of gap-placements into level segment i. If the gaps are placed in all G level segments uniformly then $g_i = 1$ for all G parts and H_g will be 1; if all gaps are placed in one level segment H_g is zero. This controllable feature provides a notion of unpredictability of gaps and, therefore, jumps in the level. Unpredictability has been identified as an important factor of playing experience [24].
- Direction switch S. This parameter defines the percentage of the level played in the left direction. Zero direction switch means that the player needs to move from left to right in order to complete the level, as in the original Super Mario Bros. If $S > 0$ the level direction will be mirrored at certain switch points, forcing the player to turn around and go the other way for $S\%$ of the level, until reaching the end of the level or the next switch.

The selection of these particular controllable features was done after consulting game design experts, and with the intent to find features which where common to most, if not all, platform games.

Two states (low and high) for each of the four controllable parameters above are investigated. The combinations of these states result in $2^4 = 16$ different variants of the game which are used in the user study designed. In the Super Mario Bros game investigated here the number of coins, opponents, coin blocks, powerup blocks and empty blocks are fixed to 15, 3, 4, 2, and 8 respectively.

B. Gameplay features

Several statistical features are extracted from playing data which are logged during gameplay and include game completion time, time spent on various tasks (e.g. jumping, running), information on collected items (e.g. type and amount), killed enemies (e.g. type, amount, way of killing) and information on how the player died. The choice of those specific statistical features is made in order to cover a decent amount of Super Mario Bros playing behavior dynamics. In addition to the four controllable game features that are used to generate Super Mario Bros levels presented earlier, the following statistical features are extracted from the gameplay data collected and are classified in five categories: jump, time, item, death, kill and misc.

Jump: difference between the total number of jumps, J, minus the number of gaps, G; number of jumps over gaps or without any purpose (e.g. to collect an item, to kill an opponent), J'; difference between J' and the number of gaps, G; and a jump difficulty heuristic, J_d, which is proportional to the number of Super Mario deaths due to gaps, number of gaps and average gap width.

Time: completion time, t_c; playing duration of last life over total time spent on the level, t_{ll}; percentage of time that the player: is standing still, t_s, running, t_r, is on Big Mario mode, t_l, is on fire Mario mode, t_f, is on powerup mode, t_p, is moving left, t_L, is moving right, t_R, and is jumping, t_j.

Item: number of collected items (coins, destroyed blocks and powerups) over total items existent in the level, n_I; number of times the player kicked an opponent shell, n_s; number of coins collected over the total number of coins existent in the level, n_c; number of empty blocks destroyed over the total number of empty blocks existent in the level, n_{eb}; number of coin blocks pressed over the total number of coin blocks existent in the level, n_{cb}; number of powerup blocks pressed over the total number of powerup blocks existent in the level, n_p; and the sum of all blocks pressed or destroyed over the total number of blocks existent in the level $n_b = n_{eb} + n_{cb} + n_p$.

Death: number of times the player was killed: by an opponent, d_o; by jumping into a gap, d_g; by jumping into a gap over the total number of deaths d_j.

Kill: number of opponents died from stomping over the total number of kills, k_s; number of opponents died from fire-shots over the total number of kills, k_f; total number of kills over total number of opponents, k_T; number of opponent kills minus number of deaths caused by opponents, k_P; and number of cannonballs killed, k_c.

Misc: number of times the player shifted the mode (Small, Big, Fire), n_m; number of times the run button was pressed, n_r; number of ducks, n_d; number of cannonballs spawned, n_{cs}; and whether the level was completed or not C (boolean).

C. Reported player experience and experimental protocol

We designed a game survey study to solicit pairwise emotional preferences (preferences of affective states) of subjects playing different variants of the test-bed game by following the experimental protocol proposed in [7]. Each subject plays a predefined set of four games in pairs: a game pair of game A and game B played in both orders. The games played differ in the levels of one or more of the four controllable features presented previously. For each completed pair of games A

and B, subjects report their emotional preference using a 4-alternative forced choice (4-AFC) protocol:

- game A [B] was/felt more E than game B [A] game (*cf.* 2-alternative forced choice);
- both games were/felt equally E or
- neither of the two games was/felt E.

Where E is the affective state under investigation and contains *fun*, *challenging*, *boring*, *frustrating*, *predictable* and *anxious*. The selection of these six states is based on their relevance to computer game playing and their popularity when it comes to game-related user studies [25]. While in [23] we focused on modeling only fun, challenge and frustration, in this paper we model all six affective states. Note that the affective modeling procedure followed in this paper focuses on cognitive player responses which are labeled as discrete affective states and thereby models constructed capture the cognitive aspect of the player's affective state [26]; the physical component of affect is not investigated here. Also note that we can, strictly speaking, only claim to model an approximation of affect expressed via self-reports rather than the actual affect.

Data is collected over the Internet. Users are recruited via posts on blogs, mailing lists and *Facebook* and are directed to a web page containing a Java applet implementing the game and questionnaire[2]. As soon as the four games are played and the questionnaire is completed, all the features (controllable, gameplay and player experience) are saved in a database at the server hosting the website and applet. Data collection is still in progress and at the moment of writing, 181 subjects have participated in the survey experiment. The minimum number of experiment participants required is determined by $C_2^{16} = 120$, this being the number of all combinations of 2 out of 16 game variants. The experimental protocol is designed in such a way that at least 2 preference instances should be obtained for each pair of the 16 game variants played in both orders (1 preference instance per playing order). The analysis presented in this paper is based on the 240 game pairs (480 game sessions) played by the first 120 subject participants.

IV. Preference Learning for Modeling Player Experience

Based on the data collected in the process described above, we try to approximate the function from gameplay features (e.g. number of coins gathered) and controllable game level features (e.g. number of gaps) to reported emotional preferences using neuroevolutionary preference learning. We proceed in a bottom-up fashion, starting with finding linear correlations, then trying simple nonlinear models, and finally more complex but also more powerful nonlinear models.

The data is assumed to be a very noisy representation of the underlying function, given the high level of subjectivity of human preferences and the expected variation in playing styles. Together with the limited amount of training data, this makes overfitting a potential hazard and mandates that we use a robust function approximator. We believe that a nonlinear function such as an artificial neural network (ANN) is a good choice for approximating the mapping between reported affect and input data. Thus, both simple single-layer and multi-layer perceptrons are utilized for learning the relation between features (ANN inputs) — selected from feature selection schemes presented in Section V — and the value of the investigated emotional preference (ANN output) of a game. The main motivation for using single-layer perceptrons (SLPs) in addition to MLPs used in this study is that we want to be able to analyze the trained function approximator and discuss the underlying physical meaning of the nonlinear relationships obtained; e.g. see discussion in Section VII-A. While an MLP can potentially approximate the function investigated with a higher accuracy, it is much easier for a human to understand the obtained function when represented as a single-neuron ANN.

The single neuron uses the sigmoid (logistic) activation function; connection weights take values from -5 to 5 to match the normalized input values that lie in the [0, 1] interval. Since there are no prescribed target outputs for the learning problem (i.e. no differentiable output error function), ANN training algorithms such as back-propagation are inapplicable. Learning is achieved through preference learning using artificial evolution of neural networks (neuroevolution) [27]. In one of the authors' recent empirical comparison [28] of preference learning algorithms on a problem similar to the one considered in this paper, neuro-evolution has been found to be more effective than a number of other approaches, including variants of large margin algorithms and bayesian learning.

A generational genetic algorithm (GA) was implemented, using a fitness function that measures the difference between the subject's reported emotional preferences and the relative magnitude of the corresponding model (ANN) output. More specifically, the logistic (sigmoidal) function $g(e, \epsilon) = 1/(1 + e^{-\epsilon e(f_k)})$ is used where $e = e(A) - e(B)$ is the difference of the ANN output values (investigated emotion/affective state) between game A and game B; $\epsilon = 30$ if $A \succ B$ and $\epsilon = 5$ if $A \prec B$. Both the sigmoidal shape of the objective function and its selected ϵ values are inspired by its successful application as a fitness function in neuro-evolution preference learning problems [28], [27].

A population of 1000 individuals was used, and evolution run for 100 generations. A probabilistic rank-based selection scheme was used, with higher ranked individuals having higher probability of being chosen as parents. Reproduction was performed by uniform crossover, followed by Gaussian mutation with a 5% probability.

V. Feature Selection

We would like our model to be dependent on as few features as possible, both to make it easier to analyze, and to make it more useful for incorporation into future games for purposes of e.g. content creation. Additionally, there is evidence that cutting out unnecessary features improves learning quality for evolutionary training of neural networks [29]. Therefore, feature selection is utilized to find the feature subset that yields that most accurate user model and save computational effort

[2]The game and questionnaire are available at www.bluenight.dk/mario.php

of exhaustive search on all possible feature combinations. The quality of the predictive model constructed by the preference learning outlined above depends critically on the set of input data features chosen. Using the extracted features described earlier the *n best individual feature selection* (nBest), the *Sequential Forward Selection* (SFS), the *Sequential Floating Forward Selection* (SFFS) and the *Perceptron Feature Selection* (PFS) schemes are applied and compared.

Note that neither method presented is guaranteed to find the optimal feature set since neither searches all possible combinations (they are all variants of hill-climbing). To evaluate the performance of each input feature subset, the available data is randomly divided into thirds and training and validation data sets consisting of 2/3 and 1/3 of the data respectively are assembled. The performance of each user model is measured through the average classification accuracy of the model in three independent runs using the 3-fold cross-validation technique on the three possible independent training and validation data sets. Since we are interested in the minimal feature subset that yields the highest performance we terminate selection procedure when an added feature yields equal or lower validation performance to the performance obtained without it. On the same basis, we store all feature subsets selected by PFS (as described below) and explore the highest performing subset starting with the smallest feature subset generated.

A. nBest

nBest feature selection ranks the features used individually in order of model performance; the chosen feature set of size n is then the first n features in this ranking. The nBest method is used for comparative purposes, being the most popular technique for feature selection.

B. SFS

SFS is a bottom-up search procedure where one feature is added at a time to the current feature set. The feature to be added is selected from the subset of the remaining features such that the new feature set generates the maximum value of the performance function over all candidate features for addition. The SFS method has been successfully applied to a wide variety of feature selection problems, yielding high performance values with minimal feature subsets [7], [28]

C. SFFS

Several studies (e.g. [30] among others) have demonstrated the benefits of sequential floating search over standard sequential search. Floating search algorithms can be classified into forward and backward search. The sequential floating forward search (SFFS) algorithm performs the sequential steps of the SFS algorithm. However, each time an SFS step is performed, SFFS checks whether the performance function value can be increased if a feature is excluded from the current feature subset (i.e. one step of sequential backward selection is performed).

D. PFS

The fourth method we investigate is an aggressive-search variant of neural pruning and sequential backward selection. Rosenblatt's perceptron is used as a methodology for selecting appropriate feature subsets. Our algorithm which is similar to [31] is adjusted to match preference learning problems. Thus, the perceptron used employs the sigmoid activation function in a single output neuron. The ANN's initial input vector has the size of the number of features examined. The perceptron feature selection (PFS) procedure is as follows:

Step 1 Use artificial evolution to train the perceptron on the pairwise preferences (see Section IV). Performance of the perceptron is evaluated through 3-fold cross-validation. The initial input vector consists of all features extracted $\mathcal{F}$ (40 in this paper).

Step 2 Eliminate all features $\mathcal{F}'$ whose corresponding absolute connection weight values are smaller than $E\{|\mathbf{w}|\} - \sigma\{|\mathbf{w}|\}$, where $\mathbf{w}$ is the connection weight vector.

Step 3 If $\mathcal{F}' = \emptyset$ continue to Step 4, otherwise use the remaining features and go to Step 1.

Step 4 Evaluate all feature subsets obtained using the neuro-evolution preference learning approach presented in Section IV.

VI. Statistical Analysis

This section describes testing for correlations between playing order, controlled features and gameplay features and the six reported affective states.

To check whether the order of playing Super Mario game variants affects the user's judgement of emotional preferences, we follow the order testing procedure described in [6] which is based on the number of times that the subject prefers the first or the second game in both pairs. The statistical analysis shows that order of play does not affect the emotional preferences of fun and frustration; however a statistically significant effect (significance equals 1% in this paper) is observed in challenge (p-value $= 0.006$) and anxiety (p-value $= 0.007$) preferences. The effect reveals a preference for the second game played which implies the existence of random noise in challenge and anxiety preference expression. On the other hand, the insignificant order effects of fun, frustration, predictability and boredom in part, demonstrate that effects such as a user's possible preference for the very first game played and the interplay between reported affective states and familiarity with the game are statistically insignificant.

More importantly, we performed an analysis for exploring statistically significant correlations between subject's expressed preferences and extracted features. Correlation coefficients are obtained through $c(\mathbf{z}) = \sum_{i=1}^{N_s}\{z_i/N_s\}$, where N_s is the total number of game pairs where subjects expressed a *clear preference* for one of the two games (e.g. $A \succ B$ or $A \prec B$) and $z_i = 1$, if the subject preferred the game with the larger value of the examined feature and $z_i = -1$, if the subject chooses the other game in the game pair i. Note that, N_s is 161, 189, 151, 158, 128 and 138 respectively, for

reported fun, challenge, frustration, predictability, anxiety and boredom.

The variation of the N_s numbers above indicates, in part, the difficulty in expressing a clear emotional preference on different game variants. The percentage of $A \succ B$ and $A \prec B$ selection occurrences over all 240 preference instances for different affective states varies from 78.7% (challenge) to 53.3% (anxiety). These percentages provide some first evidence that the selected game level and rule parameters have an dissimilar impact on the affective states investigated. For instance, challenge and fun appear to be very much affected by varying the selected parameters whereas anxiety and boredom, on the contrary, do not appear as an affective state which is directly affected by level feature and game rule variations in the game.

A. Fun

Statistically significant correlations are observed between reported fun and seven features: number of times the player kicked a turtle shell, proportion of coin blocks that were "pressed" (jumped at from below), proportion of opponents that were killed, number of times the run button was pressed, proportion of time spent moving left, number of enemies killed minus times died, and proportion of time spent running. All of these were positive correlations.

Such correlations draw a picture of most players enjoying a fast paced game that includes near-constant progress, plenty of running, many enemies killed and many coins collected from bouncing off the coin blocks. One might argue that this picture fits with the concept of *Flow*, in that the player makes unhindered progress [3]. However, the Flow concept also includes a certain level of challenge, and no features that signify challenge are associated with fun in this case. It might be that players enjoy when the game is easy — at least when they only play a single level of the game.

The feature that correlates the most with fun preferences is kicking turtle shells. Kicking a turtle shell is a simple action which often results in the unfolding of a relatively complex sequence of events, as the shell might bounce off walls, kill enemies, fall into gaps etc. The fun inherent in setting of complex chains of events with simple actions is something many players can relate to and which features prominently in many games, and relates to the theory supporting the relationship between emergent gameplay and enjoyment [32].

B. Challenge

Eighteen features are significantly correlated with challenge. The ten most highly correlated are ($+/-$ in parenthesis signifies positive or negative correlation): whether the level was completed ($-$), proportion of power-up blocks pressed ($-$), proportion of Mario deaths that were due to falling into a gap ($+$), number of times Mario died from falling into a gap ($+$), jump difficulty ($+$), average width of gaps ($+$), number of times Mario ducked ($-$), proportion of time spent in the last life ($-$), proportion of coin blocks that were pressed ($-$), and the number of gaps ($-$). In addition, a weaker but still significant positive correlation was found between gap entropy, H_g, and challenge.

A first observation is that it is obviously much easier to predict challenge than to predict fun — many more features are significantly correlated, and the correlations are stronger. It also seems that challenge is somehow orthogonal to fun, as almost none of the features that are correlated with challenge are correlated with fun. The exception is the proportion of coin blocks pressed, but while this feature is positively correlated with fun it is negatively correlated with challenge. (This is somewhat expected: if the level is so hard that the player has to struggle to survive it, she does not have time to make detours in order to collect more coins.)

Most of the correlations are easy to explain. That a level is perceived as less challenging if you complete it should not come as a surprise to anyone. Likewise, we can understand that players think a level is hard when they repeatedly die from falling into gaps. Three particularly interesting correlations are those between the controllable features and challenge: increase in gap width, $E\{G_w\}$, and gap entropy, H_g, imply increased challenge whereas increased number of gaps, G, implies a linear decrease of challenge. These effects suggest that challenge can be controlled to a degree by changing the number, width and distribution of gaps.

The *negative* correlation between number of ducks and challenge has a slightly less intuitive explanation. The main reason for ducking in Super Mario Bros (at least in the tested levels) is to avoid cannonballs, generally perceived as some of the most difficult elements on a level, which would suggest that ducking more would indicate a harder level. However, players reported lower challenge on levels where they ducked many times. The explanation is that ducking is only possible when Mario is in the Big or Fire state, so being able to duck means that you have not gotten hurt, which indicates a lower challenge.

C. Frustration

Twenty-eight features are significantly correlated with frustration, and some of the correlations are extremely strong. Of the top ten correlated features, most are also in the top ten list for features correlated with challenge, and correlated in the same way. The exceptions are proportion of collected items ($-$), time spent standing still ($+$), proportion of killed opponents that were killed with fireballs ($-$), and proportion of coins collected ($-$).

From these new features, it seems that a frustrated player is most likely one that spends time standing still and thinking about how to overcome the next obstacle; is far too busy overcoming obstacles to collect coins and power-ups; and as a result of not collecting power-ups is rarely in the Fire Mario state (necessary to shoot fireballs). But frustration can also be very well predicted from not winning the level and from falling into gaps often, just like challenge.

D. Predictability

Seven features are statistically correlated with reported predictability. The features are: the difficulty of the jumps,

TABLE I
TOP TEN STATISTICALLY SIGNIFICANT (P-VALUE $< 1\%$) CORRELATION COEFFICIENTS BETWEEN REPORTED EMOTIONS AND EXTRACTED FEATURES. CONTROLLABLE FEATURES APPEAR IN BOLD.

Fun		Challenge		Frustration		Predictability		Anxiety		Boredom	
n_s	0.345	C	-0.600	C	-0.826	J_d	-0.395	C	-0.500	$\mathbf{E\{G_w\}}$	-0.308
n_{cb}	0.311	n_p	-0.480	n_p	-0.815	$\mathbf{E\{G_w\}}$	-0.383	d_j	0.377		
k_T	0.256	d_j	0.469	n_{cb}	-0.688	d_j	-0.378	$\mathbf{E\{G_w\}}$	0.373		
n_r	0.253	d_g	0.447	d_g	0.578	C	0.362	d_g	0.333		
t_L	0.237	J_d	0.439	d_j	0.564	d_g	-0.333	J_d	0.326		
k_P	0.222	$\mathbf{E\{G_w\}}$	0.409	n_I	-0.544	t_{ll}	0.224				
t_r	0.192	n_d	-0.368	t_s	0.520	t_R	0.205				
		t_{ll}	-0.312	k_f	-0.515						
		n_{cb}	-0.292	t_{ll}	-0.513						
		$\mathbf{G}$	-0.287	n_c	-0.511						

J_d $(-)$, gap widths $E\{G_w\}$ $(-)$, number of deaths by falling into gaps over the total number of deaths d_j $(-)$, whether or not the level was completed C (+), total number of deaths by falling into gaps d_g $(-)$, time spent of the last life in the level over the total time spent on the level t_{ll} $(+)$, and the time spent going in the right direction over the total time t_R $(+)$.

The majority of the features correlated to predictability are in some way linked to gaps, for example, it appears that the game is less predictable when the difficulty of the gaps is higher, the player dies from falling into gaps more often, and when the gaps are wider. Moreover, unsurprisingly, a game is reported as being more predictable if the player manages to complete the level, which might indicate the existence of gameplay experience with similar levels. On the same basis, more time is spent on the level when either level completion has been achieved or the game is played comfortably by the player; both are indicators of higher level predictability which is reported by the test subjects.

It also appears that players which spend more time on the last life, compared to the times spent on previous lives, find the game more predictable. This positive correlation is easily justified since players have already seen parts of the level when their last life is played.

Somewhat surprisingly, the entropy of gap placement is not correlated with predictability. This suggest a possible nonlinear relationship.

E. Anxiety

All five features correlated with reported anxiety are also correlated with reported predictability; however, correlation values for those features are inverted. This generates the assumption that players get more anxious the less predictable the level is. Intra-correlations between the two reported emotions verify elements of this relationship; see Section VI-H. It is also worth noticing that all five features are present among the ten most correlated features of challenge; correlation values of those features have the same sign in both affective states. This observation is supported through the intra-correlation between reported anxiety and challenge which is found to be positive and statistically significant (see Section VI-H).

F. Boredom

The only feature highly correlated to boredom is the average gap width $E\{G_w\}$ $(-)$. According to this statistically significant effect, the game is reported as less boring the wider the gaps existent in the level. The existence of only one correlated feature shows the difficulty of predicting player boredom with a linear model.

G. Controllable features and reported emotions

When only looking at linear correlations, it would appear that fun and frustration are not connected to any of the four controllable features. Fun and boredom are also less strongly correlated with gameplay features than is the case for the other four emotions. Challenge is easiest to model with linear models, and it is also correlated with controllable features, namely gap width and gap placement. Predictability, anxiety and boredom are also correlated to gap width.

As the ultimate goal of this project is to be able to optimize game levels for specified emotions given data on a particular player's playing style, we need to find models of these emotions that include controllable features. It could therefore be seen as a partial failure to only be able to find significant correlations between controllable features and for of the six investigated emotions. However, this overlooks that controllable features might affect reported emotions in a nonlinear fashion — it is for example plausible that a particular controllable feature (e.g. jump width) contributes positively to fun for a particular group of players (e.g. skilled players), but negatively for another group (i.e. novice players). This points to the need for nonlinear modeling of these emotions.

H. Intra-correlations among reported emotions

This section presents an analysis of the correlations $(c(\mathbf{z}))$ between the six reported emotions. The significant effects presented in Table II show that challenging games are likely to be more fun, more frustrating, less boring, less predictable, and eliciting more anxiety. Games reported as fun are more likely to be less boring, less predictable, more challenging and less frustrating. Statistically significant effects are also observed between frustration and challenge $(+)$, fun $(-)$, predicability $(-)$, and anxiety $(+)$. Players expressing more boredom for a game appear to express less challenge, fun and frustration, and

more anxiety. In game variants perceived as more predictable, players are more likely to report less challenge, fun, frustration and anxiety, and more boredom. Finally, when Mario players feel anxious, they appear more challenged and frustrated, and less bored; furthermore those players feel that levels are less predictable.

All aforementioned significant effects appear reasonable and show the linear inter-dependencies between reported emotions. These interrelationships appear orthogonal in several occasions; e.g. challenge is positively correlated to fun and frustration but fun and frustration are negatively correlated. Such orthogonal dependencies might generate difficulties when player experience needs to be optimized; game design implications that may arise are discussed in the last section of the paper.

TABLE II
CORRELATIONS BETWEEN REPORTED CHALLENGE (C), FUN (F), FRUSTRATION (FR), BOREDOM (B), PREDICTABILITY (P) AND ANXIETY (A). STATISTICALLY SIGNIFICANT (P-VALUE $< 1\%$) VALUES APPEAR IN BOLD.

	F	FR	B	P	A
C	**0.346**	**0.462**	**−0.600**	**−0.640**	**0.636**
F		**−0.266**	**−0.596**	**−0.465**	0.217
FR			−0.134	**−0.245**	**0.472**
B				**0.619**	**−0.491**
P					**−0.260**

VII. MODELING OF PLAYER EXPERIENCE PREFERENCES

This section presents the two-phase procedure followed towards modeling player experience. First, we utilize SLPs to approximate the emotional preferences of the players. The four dissimilar feature selection schemes are used to generate the input vector for the SLPs. All features (both player and controllable) are investigated at this stage.

After features that contribute to accurate SLP models are found we optimize the topology of MLPs using neuroevolutionary preference learning. The ultimate aim of this study is to control for level generation based on player experience. On that basis, it is desired that level features and game mechanics are adjusted dynamically so that the player experience (output value of MLPs) is optimized. For that purpose, all four controllable features (if not already selected from the first phase) are forced into the input vector of the MLP which includes the feature subset selected via the SLP modeling procedure. The procedure followed is depicted in Fig. 2.

The rationale behind this two-phase approach is three-fold:

1) *Expressiveness of SLP models*. Using simple nonlinear models (rather than more complex MLPs) allows for a clearer observation of the player characteristics, level features and game mechanics that contribute to each affective state investigated. This discussion is vital for the deeper understanding of the unknown function that lies between statistical features of play, controllable in-game parameters and reported emotions. The MLPs ultimately have more expressive power (and, as we shall see, are capable of learning more accurate models) and thus it is possible that there are some feature subsets, depending e.g. on XOR-like relationships that are suitable for the MLPs but will not be found by SLP-based feature subset selection. However, this is a tradeoff we have to make, given that performing the feature subset selection directly on MLPs is prohibitively computationally expensive. If we had unlimited computational time, we could have performed the feature selection using MLPs, but we could also have used exhaustive search in feature subset space rather than the local search heuristics currently employed.
2) *Computational effort*. It is computationally preferred to apply feature selection on SLPs and then optimize the topology of MLPs using the selected feature subset rather than attempting to optimize both the input (feature selected) and the topology of an MLP. To further support this hypothesis we provide some indicative CPU times for the two phases of the modeling process of *fun*. The CPU time of SFS (or SFFS), being the most expensive feature selection method, applied to an SLP equals 643.4 seconds for ten features respectively — for this example we restrict the investigation to 10 features. Furthermore, the total CPU time of a run investigating all possible MLP architectures of two hidden layers, with up to 30 and 10 hidden neurons respectively in the first and second layer equals 493417.8 seconds. Thus $643.4 + 493417.8 = 494052.2$ seconds are required for the whole procedure as proposed. If both the input and the topology of an MLP were to be optimized in a single phase that CPU time would have been roughly 705769.1 seconds. Note that this is a lower-bound CPU-effort approximator resulting from the addition of 643.4 seconds (SFS effort) to all 330 architectures investigated. All experiments presented in this paper run on a 2.53 GHz, 4GB ram, 64 bit MS Windows machine.
3) *Representation completeness*. We force all four controllable features to be embedded in the model. That enforcement gives the designer all the flexibility the parameter space offers to effectively tailor player experience by generating personalized content for the player.

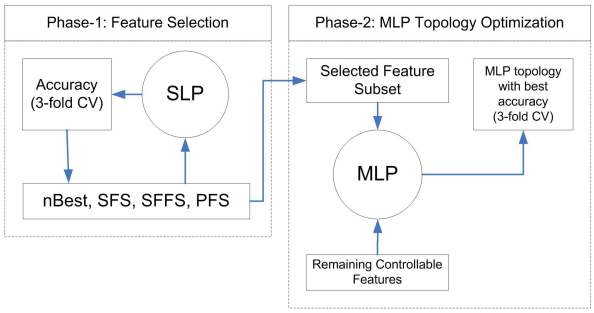


Fig. 2. The two-phase modeling approach followed

A. Single Layer Perceptron Models: Feature Selection

The correlations calculated above provide linear relationships between individual features and reported emotions. How-

ever, these relationships are most likely more complex than can be captured by linear models. The aim of the analysis presented below is to construct nonlinear computational models for reported emotions and analyze the relationship between the selected features and expressed preferences. Furthermore the feature subsets that derive from the combination of feature selection on the SLP models presented here (phase 1 in Fig. 2) are used as inputs of the MLP models constructed in the next subsection.

For this purpose we evolve weights for nonlinear SLPs as described in Section IV. The weights of the highest performing networks are presented in Table III. All evolved networks performed much better than networks with random weights, which reached chance level prediction accuracy.

As a general observation, sequential forward selection appears to be the most appropriate feature selection mechanism for all six emotions. SFS develops features subsets that feed SLP models which achieve the highest cross-validation performance. Even though SFFS is a more efficient hill-climber which allows backward search it does not showcase that advantage in small feature sets (e.g. less than 8 features) as the ones examined in this paper. Note that SFFS performance is different from SFS performance only in challenge prediction since two backward steps occurred during that run; no successful backward step was performed in any other affective state predicted resulting in equal performance values for SFS and SFFS. A detailed analysis of the SLP predictors of each affective state is presented below.

1) Fun: In the comparison between the four different selection mechanisms applied it is evident that SFS and SFFS have advantages over nBest and PFS for fun preferences (see Fig. 3(a)). nBest achieves a satisfactory performance (67.92%) but requires 10 features as inputs to the ANN. PFS generates the lowest classification accuracies; its best network has an accuracy of 63.52% with a selected subset of 11 input features.

The best obtained perceptron model of fun preferences is designed by SFS (and SFFS). This model achieves a performance of 69.18% which is with a selected feature subset of size three. The selected perceptron input vector consists of the time spent moving left t_L, the number of opponents died from stomping over the total number of kills, k_s, and the controllable *switching* feature which is defined as the percentage of level played in the left direction S[3].

Fun is the second least correlated of the six modeled emotions, and the hardest (along with boredom) to model with a nonlinear perceptron as well. Still, it's remarkable that this complex affective state can be predicted to a moderate degree simply by observing that Mario keeps running left and kills enemies by stomping.

2) Challenge: The best-performing ANN for challenge prediction has an accuracy of 77.77%. It is more complex than the best fun predictor, using five features: time spent standing still $(-)$, jump difficulty $(+)$, proportion of coin blocks pressed $(-)$, number of cannonballs killed $(-)$ and proportion of kills by stomping $(-)$. While the jump difficulty heuristic has the largest corresponding weight — a testament to the central role of gap placement and size for challenge — it is also the only feature related to gaps used by this model, pointing to the adequateness of this particular heuristic.

3) Frustration: Our best evolved ANN for predicting frustration has an accuracy of 88.66%. We can predict with near-certainty whether the player is frustrated by the current game by just calculating the time spent standing still $(+)$, the proportion of time spent on last life $(-)$, the jump difficulty $(-)$, and the proportion of deaths due to falling in gaps $(+)$.

Somewhat surprisingly, time spent standing still counts *against* challenge, whereas it is a strong *positive* predictor of frustration. This observation could be valuable if trying to design a feedback system that keeps the game challenging but not frustrating. Another feature that has different effect on challenge and frustration is jump difficulty, where frustration is connected with *lower* jump difficulty. Maybe the player gets frustrated by falling into gaps that she knows are not that hard.

That the player feels frustrated when dying after a short time during his last life is understandable — many players feel that their last attempt should be their best. Additionally, a high frustration level can cause the player to care less about the game and play worse in her final life.

4) Predictability: Predictability can be predicted relatively accurately (76.28%). More features are selected as relevant for predicting this emotion than any of the five other emotions, and consists of jump difficulty $(-)$, number of cannonballs killed $(+)$, gap width $(-)$, time spent moving left $(-)$, number of mode shifts $(-)$, difference between number of jumps and gaps $(+)$ and time spent moving right $(+)$.

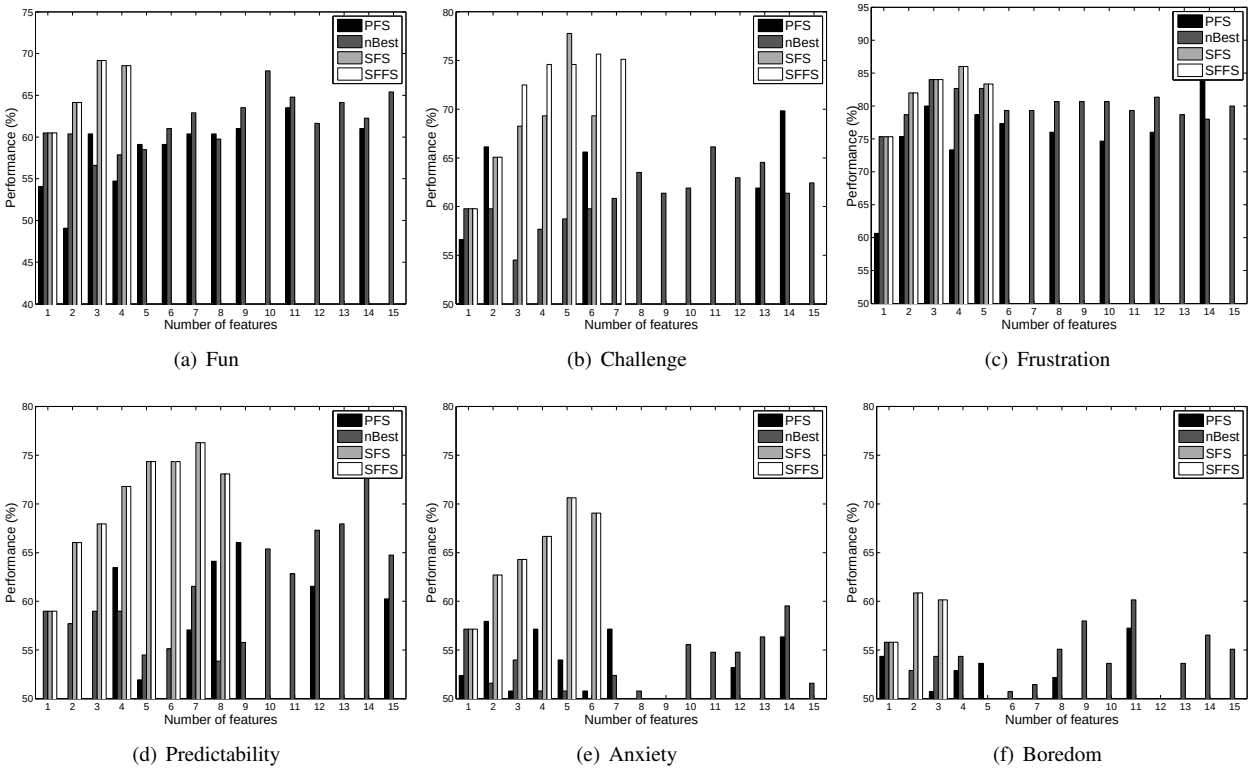
Overall, certain features that point to a more challenging game also make it less predictable (harder gaps, failed jumps over gaps, getting hurt more often); this is reinforced by the strong negative correlation between challenge and predictability. Additionally, and predictably, the direction switch feature also makes the game less predictable. The role of the cannon ball kills is not entirely clear, but one hypothesis is that players kill more cannon balls when they are able to predict at what time they are fired, and this capacity for predicting cannon balls contributes to a feeling of being able to predict the game as a whole.

5) Anxiety: Five selected features contribute to predicting anxiety with an accuracy of 70.63%: gap width $(+)$, completion time $(-)$, number of ducks $(-)$, proportion of coin blocks pressed $(-)$ and number of cannon balls killed $(+)$.

That several features that are associated with challenge (difficult gaps, long completion time, not having time to press coin blocks and having to kill many cannon balls) contribute to anxiety is not surprising, given the strong positive correlation between challenge and anxiety. However, lest one think that high anxiety, high challenge and low predictability is the same cognitive state, it is worth pointing out that they differ with respect to at least one important feature: number of cannon balls killed is associated with high challenge and high anxiety, but also with *high* predictability.

The negative contribution to anxiety from number of ducks

[3]The S feature is there to correct for the fact that when the level direction switches, Mario moves right rather than left to move forward, and so t_L is diminished. This points to an oversight on our part when designing the gameplay features: we should have measured the time spent moving *towards the end of the level* rather than moving left.



(a) Fun (b) Challenge (c) Frustration

(d) Predictability (e) Anxiety (f) Boredom

Fig. 3. Performance comparison of the four feature selection mechanisms for all six affective states investigated.

TABLE III
LEARNING FROM PREFERENCES: FEATURES AND CORRESPONDING CONNECTION WEIGHTS FOR HIGHEST PERFORMING ANNS. FEATURES ARE PRESENTED IN DESCENDING ORDER OF THEIR CORRESPONDING ABSOLUTE CONNECTION VALUES. ANN 3-FOLD CROSS VALIDATION ACCURACY IS DEPICTED AT THE BOTTOM ROW OF THE TABLE. CONTROLLABLE FEATURES APPEAR IN BOLD.

Fun		Challenge		Frustration		Predictability		Anxiety		Boredom	
t_L	4.90	J_d	3.80	t_s	3.26	J_d	-3.75	$\mathbf{E\{G_w\}}$	2.35	d_o	-4.70
$\mathbf{S}$	-3.87	t_s	-1.70	t_{ll}	-1.85	k_c	2.75	t_c	-1.10	t_R	2.11
k_s	0.94	n_{eb}	-1.50	J_d	-0.99	$\mathbf{E\{G_w\}}$	-2.49	n_d	-0.89		
		k_c	1.07	d_g	0.23	t_L	-1.38	n_{cb}	-0.85		
		k_s	-0.18			n_m	-1.16	k_c	0.74		
						J'	0.54				
						t_R	0.05				
69.18%		77.77%		88.66%		76.28%		70.63%		60.87%	

could signify that players who frequently duck to avoid incoming cannon balls, rather than attempting to jump over them, are better Mario players and therefore less anxious.

6) Boredom: Boredom is the hardest of the six reported emotions to predict, with an accuracy of only 60.87%. The trained network used only two features: number of deaths from opponents $(-)$ and time spent going right $(+)$.

That dying from opponents makes the game less boring could mean that those players who find the game boring are good players that like to be presented with a challenge (indeed, boredom and challenge are negatively correlated), and that if such players die from anything it is from opponents. However, given the low accuracy of boredom predictors, this indication is somewhat tentative. As for the time spent moving right, this is simply a confirmation that the direction switch feature is appreciated by players.

B. Multi Layer Perceptron Models: Optimizing Topology

Above we have shown that fun, challenge, frustration, predictability, anxiety and boredom of Super Mario Bros players can be approximated with reasonable accuracy via a single-layer perceptron model. In particular, reported fun, challenge, frustration, predictability, anxiety and boredom were approximated with respective accuracies of 69.18%, 77.77%, 88.66%, 76.28%, 70.63% and 60.87% using SFS as the feature selection mechanism. However, these nonlinear predictors (at least for fun anxiety and boredom) are still not as good as a designer would like them to be. Furthermore a designer cannot predict those emotions from all available *controllable* features since those are not embedded in the derived models. Since controllable features (such as level design parameters) are those that we can vary, and therefore those that can be optimized by evolution or other global optimizers, we need to

be able to predict emotions, at least partly, from controllable features. Tailoring player experience in real-time via automatic game content generation defines the ultimate aim of this study; however, such a goal is not possible without controllable features embedded in the affective models constructed.

These observations point to the need for better models and/or input feature sets. For that purpose all remaining controllable parameters which are not included in the selected feature subset are forced into the input of multi-layer perceptron (rather than single-layer perceptron) models of emotional preferences and their topologies are optimized for maximum prediction accuracy (the reader is referred to the second phase of the modeling approach depicted in Fig. 2). MLPs have the advantage of universal approximation capacity; in particular, combinatorial relationships (such as XOR) can be represented. For instance, we might very well have a situation were one controllable feature (such as gap width) can be both negatively and positively connected with an emotion (such as frustration) depending on the player's playing style, as measured through gameplay features (such as number of jumps). Such relationships can be captured by MLPs but not by nonlinear perceptrons.

The experiment designed to optimize the topology of MLP affective models is as follows. MLP topologies of two hidden layers, with up to 30 and 10 hidden neurons respectively in the first and second layer are investigated; this sums to 330 different topologies which are tested for each input vector (feature set selected from SLPs plus the remaining controllable features). The model training procedure follows the preference learning method described in Section IV. The smallest possible MLPs (with respect to the number of connection weights and number of hidden layers) that achieve the highest cross-validation performance are selected and presented in Table IV. For each emotion the SLP performance, the corresponding MLP built on the selected feature subset (MLP_s); and the MLP built on controllable features solely MLP_c are presented for comparison purposes. MLP_s and MLP_c have the same topology as the optimized MLP as presented in Table IV.

In general it is worth noticing that the networks have a wide variety of sizes: from the relatively small networks of fun and challenge, to the moderate-sized MLPs of predictability and anxiety, to the large MLPs of frustration and boredom. Independently of size, all MLP models exhibit an improvement in their performance compared to the corresponding SLP models (predictability is excluded from this observation since the MLP model's performance is maintained at 76.28%).

The best topology found for each affective state varies a lot across all six emotions showcasing, in part, the complexity of predicting the preferences of users for each affective state individually. The topology of the fun MLP model (74.21%) is the simplest among all six affective states that are predicted consisting of two hidden layers that include two hidden neurons each. Challenge is predicted with a very small MLP consisting of 3 hidden neurons. The SLP performance (77.77%) is improved slightly resulting to the MLP performance of 79.37% which places the challenge model as the second best affective predictor. The generated MLP topology for frustration is of moderate size and achieves the highest performance (91.33%) among all six affective state models. Predictability is predicted with an accuracy of 76.28% via a moderate-sized MLP which equals the performance of the corresponding SLP for this emotion. The MLP architecture-optimization phase was beneficial for anxiety since the performance of the resulted moderate-sized MLP (77.78%) improved the prediction accuracy of the SLP by more than 7%. This indicates the high impact of ANN topology to anxiety prediction. Finally, boredom prediction accuracy is improved by 12.32% resulting to an MLP performance of 73.19%; however, the MLP architecture generated is the largest among all ANN topologies consisting of two hidden layers with 21 and 10 hidden neurons in the first layer and second layer respectively.

It is also worth noticing that MLPs trained solely on the four controllable features (MLP_c) achieve a considerable performance for most emotions. The difference between the performance of MLP and MLP_c shows, in part, the level of personalization (subjectivity) required (via the player features) to predict each emotion. For instance, player features contribute to a much better predictor of fun, predictability and frustration whereas the improvement due to the existence of player features is only 4.76% for anxiety. Moreover, the impact of forcing all controllable features to the MLP model varies across the different emotions. The reader may notice that the MLP performance drops in fun and predictability when all controllable features are embedded to the model — by comparing the performance of MLP_s versus the corresponding MLP performance for each affective state. While the performance decrease for fun (0.63%) does not appear significant the corresponding decrease for predictability (3.85%) reveals that all controls are not necessarily appropriate for predicting predictability. A designer might choose to use the MLP_s, instead of the MLP, model for predictability since the MLP_s model already contains the controllable feature of average gap width (see Table III).

VIII. Discussion

Using a combination of gameplay features and controllable features, we are able to predict several key affective states with a relatively high accuracy, but obviously not as high as we would have liked to. It should, however, be noted that we trained our predictors based on samples from only 120 players (480 games).Even though this is a considerable data set derived though experimental game surveys, and the results of cross-validation show that the data is indeed rich enough to support substantial conclusions, we believe that better models can be built on more data. It has recently been observed that even relatively simple learning algorithms can perform much better (including capturing more complicated nonlinear relationships) when given access to order of magnitudes more data [33]. Thus, data collection is continuing at the time of writing, and probably at the time of reading (the reader is welcome to contribute by visiting the project's web site). We intend to further use social media to attract new experimental subjects and use those new data to improve the accuracy of our predictions.

For the experiments presented in this paper, we only defined four controllable features, of which three relate to the gaps

TABLE IV
BEST MLP TOPOLOGY AND CORRESPONDING PERFORMANCE. THE PERFORMANCE OF THE SLP NETWORKS IS COMPARED TO MLPS BUILT ON SELECTED FEATURES: MLP_s; CONTROLLABLE FEATURES: MLP_c; AND SELECTED AND FORCED CONTROLLABLE FEATURES COMBINED: MLP.

	Fun	Challenge	Frustration	Predictability	Anxiety	Boredom
MLP Topology	6-2-2-1	9-3-1	8-10-10-1	10-4-6-1	8-5-3-1	6-21-10-1
SLP	69.18%	77.77%	88.66%	76.28%	70.63%	60.87%
MLP_s	74.84%	78.84%	87.33%	80.13%	75.40%	67.39%
MLP_c	66.04%	74.60%	78.67%	68.59%	73.02%	70.29%
MLP	74.21%	79.37%	91.33%	76.28%	77.78%	73.19%

in the level. The primary reason for restricting the level parametrization to four features is that we wanted to the data set to include players' judgements on all possible combinations of high and low states of the features, and that we were concerned about the availability of test subjects. However, it would be plausible to consider many more features, by letting each generated level/game variant use a random sample of values on all features. With a sufficiently large set of test subjects, it would be possible to consider each feature in sufficient independence from the other features. A number of meaningful additional controllable features could easily be designed; for example features relating to the number, type and distribution of enemies and items, the existence of dead ends in the level (forcing backtracking), height differences between various positions in the level etc. Moving outside of level design, it would be possible to design controllable features that relate to the physics of the game (such as gravity and inertia) and other aspects of game design (such as the meaning of "dying" or winning a level).

Additionally, it would be interesting to include features that are based on ordering in space or time. For example, we would like features that somehow encapsulate whether a player received a reward before or after a particular action was taken. While such features could to some extent be encoded using the current scheme and ad hoc definition of order relationships, a more powerful and flexible alternative would be to use techniques from sequence data mining, such as recurrent neural networks.

Another question concerns the generality of the methodology and results gathered here — do they apply to just the players and the particular game tested here, or do they have wider applicability? Similar experimental methodologies have been applied to a variety of game genres and interaction modes [27], [7], [18] to construct predictors of affective states (mainly fun). This paper supports the generality of the method proposed here to more affective states of player experience in a platform game. Furthermore, we venture that, as Super Mario Bros more or less defined the platform game genre, the results apply to some extent to all games of the same genre. As for the population of experimental subjects, it is believed to be very diverse, but this needs to be verified. A possible critique is that the emotions reported are those that have been elicited after only a few minutes of play. It is possible that challenge or predictability would factor in more if play sessions were longer, so subjects would have had a chance of getting bored with the game.

We believe the approach presented here would generalize well to other game genres, including but not limited to popular genres such as first-person shooter (FPS) and real-time strategy (RTS) games. Many of the features we define here have straightforward analogues in such games. For an FPS game, relevant gameplay features could include the numbers of frequencies of jumps, shots, weapon and weapon switches, the time spend running, shooting, hiding (ducking) and in vicinity of other bots, and entropy of position over time. For an RTS game it could include number of and entropy of clicks, proportion of clicks that were on own units, other units, and own base building, proportion of resources that are spend on base building and unit production, resource gathering speed, and entropy of position of own units. Controllable features for an FPS level could include number of rooms, size, average connectedness, and number, types and entropy of distribution of both power-ups and enemies. For an RTS, suitable controllable features could include number and spatial entropy of resource sources (e.g. mines), proportions and distributions of different terrain types (e.g. free space, mountains, forest), and connectedness between open areas. Additionally, a number of game-specific features could be implemented for each game.

The approach we present here could be used in games in a number of different ways. Levels could be generated offline, during development of a sequel to a game or of downloadable content, based on data collected from players of the current game. Here, a game designer could identify the most common player types (clusters of gameplay feature values) using unsupervised learning. The content generation could also be part of the game and done online, serving players new game content such as levels and game modes based on how they individually have played previous levels. At the extreme of online content creation, levels could be modified in real-time by e.g. removing or adding enemies, obstacles, shortcuts, items etc. based on the performance of the player or group of players. The latter approach is already taken by the collaborative FPS *Left 4 Dead 2* (Valve 2009); the approach is deemed effective and adding to replay value even though the aspects of the levels that can be varied are rather limited, and the player modeling is very simplistic (a single scalar representing "intensity of play"). The more sophisticated approach to player modeling we present in this paper needs relatively large data sets of player behavior and preferences, which would require instances of the game to "phone home" to central game servers, but this is routinely done today in many games for quality assurance purposes.

One of the limitations of the experimental protocol proposed is post-experience. Users report affective states after playing a pair of games which might generate memory-dependencies in the reports. Effects such as order of play and game learnability might also be apparent and interconnected to memory. The experimental protocol, however, is designed to test for order of play effects which, in part, reveal memory (report consistency over different orders) and learnability effects, if any. Results showcase that reported challenge and anxiety are affected by the order of play which in turn reveals potential memory and/or learnability effects for these two particular affective states.

Forced report (4-AFC) provides viable data for a machine learner but it does not necessarily capture the dynamics of the experience. On the other hand, free emotional report could potentially provide more genuine response but it is harder to analyze and requires a laboratory experimental setup, which is not desired given the aims of this study. There exist solutions for both testing affective models over different time windows as introduced in [21] and capturing the association between gameplay dynamics and emotional responses via e.g. recurrent neural networks. Both include future directions of this research prior to tailoring the game content for maximizing player experience.

A more general limitation with our approach is that self-evaluation of affects is inherently sensitive to self-deception. There is, however, no clear way identifying such an effect. On the other hand, no other information source would supply us with quality subjective data on all those affective aspects we are seeking to approximate. (Note that controlling the order of games and questions, as we do, alleviates the order effects inherent in naive questionnaires.) Other information sources, including physiological measures and additional gameplay metrics such as when a player stops playing the game, could be used to complement but not supplant the self-reports.

We chose to model six different discrete and predefined affective states, but it would certainly be interesting to model more aspects of player experience. For example, we believe that the concept of fairness deserves further study. Whether a player judges a game to be fair or not can be a decisive factor when deciding to continue playing or not.

Currently, work is ongoing to optimize the four controllable features for producing desired emotions in particular players. Our approach is to first let a player play a test level, and record gameplay features. We then use the gameplay features in combination with one of our trained predictors that depend on both gameplay and controllable features. The controllable features are then optimized using either genetic search (e.g. genetic algorithms) or gradient search (e.g. steepest ascent improved with line search), as proposed in [21], in order to reach a desired output of the predictor (keeping both predictor parameters and gameplay features fixed). If our predictors are accurate enough, and our test subjects' playing styles and preference do not change too much between two play sessions, this method will allow us to find level design parameters which will produce desired player experiences (e.g. maximum fun value) in particular players. This method will of course need to be verified with studies on real players, using an experimental paradigm similar to that which was used for the initial data collection (see also the experimental protocol used in [21]). Of particular interest will be whether we can use the nonlinear nature of our predictors to elicit combinations of experiences which are in themselves non-correlated or negatively correlated, for example levels that are both fun and frustrating or challenging but not frustrating.

IX. Conclusions

The work reported in this paper introduces data-driven computational models that predict several dimensions of player experience based on both level design features and gameplay metrics. The work also constitutes the first player experience study in the context of a platform game; follow-up experiments will be the first where game levels are generated based on quantitative player experience models. While our experiments were successful in the sense that the predictors achieved high accuracy on core aspects of player experience using 3-fold cross-validation, there are potentially ways to further increase those models' performance. As noted above, gathering data from more gameplay sessions could assist in the process of improving these cognitive models of player experience. However, given the challenges of accurately capturing the user's affective state reported in the literature [26], [7] we deem the models found so far good enough for use in optimizing level design parameters for player experience.

Acknowledgments

The authors would like to thank Aki Järvinen and Markus Persson for insightful discussions, the anonymous reviewers for useful comments and all subjects that participated in the experiments. Additionally, we thank Ricardo Lopes for solving the mystery concerning that players reported lower challenge when ducking. The research was supported in part by the Danish Research Agency, Ministry of Science, Technology and Innovation; project name: *AGameComIn*; project number: 274-09-0083.

References

[1] C. Bateman and R. Boon, *21st Century Game Design*. Charles River Media, 2005.

[2] K. Isbister and N. Schaffer, *Game Usability: Advancing the Player Experience*. Morgan Kaufman, 2008.

[3] M. Csikszentmihalyi, *Flow: the Psychology of Optimal Experience*. Harper Collins, 1990.

[4] R. Koster, *A theory of fun for game design*. Paraglyph press, 2005.

[5] J. Juul, *Half-real*. MIT Press, 2005.

[6] G. N. Yannakakis and J. Hallam, "Towards optimizing entertainment in computer games," *Applied Artificial Intelligence*, vol. 21, pp. 933–971, 2007.

[7] ——, "Entertainment modeling through physiology in physical play," *International Journal of Human-Computer Studies*, vol. 66, pp. 741–755, 2008.

[8] J. Togelius, R. De Nardi, and S. M. Lucas, "Making racing fun through player modeling and track evolution," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2006.

[9] ——, "Towards automatic personalised content creation in racing games," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2007.

[10] K. Compton and M. Mateas, "Procedural level design for platform games," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2006.

[11] G. Smith, M. Treanor, J. Whitehead, and M. Mateas, "Rhythm-based level generation for 2d platformers," in *Proceedings Foundations of Digital Games*, 2009.

[12] M. J. Nelson, C. Ashmore, and M. Mateas, "Authoring an interactive narrative with declarative optimization-based drama management," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2006.

[13] M. O. Riedl and N. Sugandh, "Story planning with vignettes: Toward overcoming the content production bottleneck," in *Proceedings of the 1st Joint International Conference on Interactive Digital Storytelling*, Erfurt, Germany, 2008, pp. 168–179.

[14] C. Browne, "Automatic generation and evaluation of recombination games," Ph.D. dissertation, Queensland University of Technology, Brisbane, Australia, 2008.

[15] J. Marks and V. Hom, "Automatic design of balanced board games," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2007, pp. 25–30.

[16] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2008.

[17] H. P. Martinez, A. Jhala, and G. N. Yannakakis, "Analyzing the Impact of Camera Viewpoint on Player Psychophysiology," in *Proceedings of the Int. Conf. on Affective Computing and Intelligent Interaction*. Amsterdam, The Netherlands: IEEE, September 2009, pp. 394–399.

[18] M. Schwartz, H. P. Martinez, G. N. Yannakakis, and A. Jhala, "Investigating the Interplay between Camera Viewpoints, Game Information, and Challenge," in *Proceedings of Artificial Intelligence and Interactive Digital Entertainment (AIIDE'09)*. Palo Alto, CA: AAAI Press, October 2009.

[19] A. S. Gertner, C. Conati, and K. VanLehn, "Procedural help in Andes: Generating hints using a Bayesian network student model," in *Proceedings of AAAI-98*, 1998.

[20] B. Magerko, C. Heeter, B. Medler, and J. Fitzgerald, "Intelligent Adaptation of Digital Game-Based Learning," in *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, Toronto, 2008, pp. 200–203.

[21] G. N. Yannakakis and J. Hallam, "Real-time Game Adaptation for Optimizing Player Satisfaction," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 2, pp. 121–133, June 2009.

[22] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Optimization of platform game levels for player experience," in *Proceedings of Artificial Intelligence and Interactive Digital Entertainment (AIIDE'09)*. Palo Alto, CA: AAAI Press, October 2009.

[23] ——, "Modeling Player Experience in Super Mario Bros," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. Milan, Italy: IEEE, September 2009.

[24] T. W. Malone, "What makes computer games fun?" *Byte*, vol. 6, pp. 258–277, 1981.

[25] R. L. Mandryk and M. S. Atkins, "A Fuzzy Physiological Approach for Continuously Modeling Emotion During Interaction with Play Environments," *International Journal of Human-Computer Studies*, vol. 65, pp. 329–347, 2007.

[26] R. W. Picard, *Affective Computing*. Cambridge, MA: The MIT Press, 1997.

[27] G. N. Yannakakis and J. Hallam, "Game and Player Feature Selection for Entertainment Capture," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. Hawaii, USA: IEEE, April 2007, pp. 244–251.

[28] G. N. Yannakakis, M. Maragoudakis, and J. Hallam, "Preference Learning for Cognitive Modeling: A Case Study on Entertainment Preferences," *IEEE Systems, Man and Cybernetics; Part A: Systems and Humans*, vol. 39, no. 6, pp. 1165–1175, November 2009.

[29] J. Togelius, T. Schaul, J. Schmidhuber, and F. Gomez, "Countering poisonous inputs with memetic neuroevolution," in *Parallel Problem Solving From Nature 10*, 2008.

[30] A. K. Jain and D. Zongker, "Feature-Selection: Evaluation, Application, and Small Sample Performance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 153–158, 1997.

[31] M. Mejia-Lavalle and G. Arroyo-Figueroa, "Power System Database Feature Selection Using a Relaxed Perceptron Paradigm," in *Proceedings of 5th Mexican International Conference on Artificial Intelligence, LNCS*. Springer Berlin/Heidelberg, 2006, pp. 522–531.

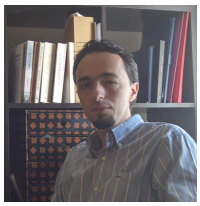
[32] P. Sweetser, "An emergent approach to game design — development and play," Ph.D. dissertation, University of Queensland, Australia, 2006.

[33] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems*, 2009.

Christopher Pedersen completed his MSc in Media Technology and Games at the IT University of Copenhagen in 2009. He received a BA in Computer Science and Business Economics from Copenhagen Business School in 2007. His research interests include cognitive modeling and player experience modeling.

Julian Togelius (S'05–M'07) is Assistant Professor at the IT University of Copenhagen (ITU). He received a BA in Philosophy from Lund University in 2002, an MSc in Evolutionary and Adaptive Systems from University of Sussex in 2003 and a PhD in Computer Science from University of Essex in 2007. Before joining the ITU he was a postdoctoral researcher at IDSIA in Lugano.

His research interests include applications of computational intelligence in games, procedural content generation, automatic game design, evolutionary computation and reinforcement learning. He is an Associate Editor of TCIAIG and a vice chair of IEEE CIS Games Technical Committee.



Georgios N. Yannakakis (S'04–M'05) is Associate Professor at the IT University of Copenhagen. He received both the 5-year Diploma (1999) in Production Engineering and Management and the M.Sc. (2001) degree in Financial Engineering from the Technical University of Crete and the Ph.D. degree in Informatics from the University of Edinburgh in 2005. Prior to joining the Center for Computer Games Research, IT University of Copenhagen in 2007, he was a postdoctoral researcher at the Mærsk Mc-Kinney Møller Institute, University of Southern Denmark.

His research interests include user modeling, neuro-evolution, computational intelligence in computer games, cognitive modeling and affective computing, emergent cooperation and artificial life. He has published around 40 journal and international conference papers in the aforementioned fields. He is an Associate Editor of IEEE Transactions of Affective Computing and the chair of the IEEE CIS Task Force on Player Satisfaction Modeling.