Towards Challenge Balancing for Personalised Game Spaces

Sander Bakkes
University of Amsterdam
Intelligent Systems Laboratory Amsterdam
The Netherlands
s.c.j.bakkes@uva.nl

Shimon Whiteson
University of Amsterdam
Intelligent Systems Laboratory Amsterdam
The Netherlands
s.a.whiteson@uva.nl

ABSTRACT

This article focuses on games that can tailor the provided game experience to the individual player (personalised games), typically by effectively utilising player models. A particular challenge in this regard, is utilising player models for assessing online (i.e., while the game is being played) and unobtrusively which game adaptations are appropriate. In this article, we propose an approach for personalising the space in which a game is played (i.e., levels) – to the end of tailoring the experienced challenge to the individual player *during actual play* of the game. Our approach specifically considers two persisting design challenges, namely *implicit user feedback* and *high risk of user abandonment*. Our contribution to this end is proposing a clear separation between (intelligent) offline exploration and (safety-conscious) online exploitation. We are presently assessing the effectiveness of the developed approach in an actual video game: INFINITE MARIO BROS. [18]. To this end, we have enhanced the game such that its process for procedural-content generation allows the game spaces (i.e., levels) to be personalised during play of the game. We use data from intelligent offline exploration to determine both a model of experienced challenge as well as safe level design parameters for use on new players. Online, we use a gradient ascent algorithm with designer-specified domain knowledge to select the next set of level design parameters.

Keywords

Personalisation, game spaces, challenge balancing, video games, implicit user feedback, user abandonment

1. INTRODUCTION

Ideally, artificial intelligence (AI) in games would provide satisfactory and effective game experiences for players regardless of gender, age, capabilities, or experience [6]; ideally, it would allow for the creation of personalised games, where the game experience is continuously tailored to fit the individual player. Indeed, it is argued that we are now at a unique point where modern computer technology, simulation, and artificial intelligence (AI) have opened up the possibility that more can be done with regard to *on-demand and just-in-time personalisation* [20]. However, a prevailing limitation in the context of game personalisation, is that learning effective behaviour online (1) requires an inconveniently large number of trials, (2) is generally highly resource-intensive, and (3) usually generates many inferior solutions before coming up with a good one [2].

As such, achieving the ambition of creating personalised games requires the development of novel techniques for assessing online and unobtrusively which game adaptations are required for optimizing the individual player's experience. To this end, this article proposes an approach for online game personalisation that reasons about uncertainty about the player probabilistically, specifically using Gaussian Process optimisation. It considers game personalisation as an optimisation problem, and details design considerations for effectively personalising the space of a game, while the human user is interacting with it. Specifically, the proposed approach considers two persisting design challenges, namely *implicit user feedback* and *high risk of user abandonment*. Our goal is to demonstrate how to personalise game spaces *during play of the game*, such they that are appropriately challenging to the individual user. To this end, our approach is implemented in the actual video game INFINITE MARIO BROS.

2. GAME PERSONALISATION

Concisely speaking, of main relevance to the motivation for adopting personalised gaming methods, is its psychological foundation, concerning, among others, a significantly increased involvement and extensive cognitive elaboration when subjects are exposed to content of personal relevance [19]; they will exhibit stronger emotional reactions [7]. Particularly, a positive effect on player satisfaction is indicated - among others, research has shown that game personalisation raises player loyalty and enjoyment, which in turn can steer the gaming experience towards a (commercial) success [28]. Also, the contribution to game development practise is evident, as the perspective of AI researchers to increase the engagement and enjoyment of the player is one that is consistent with the perspective of game designers [20]. Finally, personalisation methods are regarded as instrumental for achieving industry ambitions [14].

Tailoring the game experience to the individual player particularly benefits from the use of player models, and requires components that use these models to adapt part of the game [4]. Though by no means an exhaustive list, a set of components that will allow the vast majority of video games to be personalised are: (1) space adaptation, (2) mission/task adaptation, (3) character adaptation, (4) game-mechanics adaptation, (5) narrative adaptation, (6) music/sound adaptation, (7) and player matching (multi-player). Where desired by the game designer, the components may be informed by (8) challenge-balancing techniques for adjusting the challenge level to the individual player.

Our work follows the emerging trend of employing AI methods for adapting the game environment itself (as opposed to, more typically, adapting behaviour of the characters that operate in the game environment) [4]. In our investigation, we choose to focus on *personalising the game space* to the individual player with respect to experienced challenge[1]. Related work with regard to this scope is discussed below.

Challenge balancing
Challenge balancing concerns automatically adapting the challenge that a game poses to the skills of a human player [15, 26]. When applied to game dynamics, challenge balancing aims usually at achieving a 'balanced game', i.e., a game wherein the human player is neither challenged too little, nor challenged too much. In most games, the only implemented means of challenge balancing is typically provided by a *difficulty setting*, i.e., a discrete parameter that determines how difficult the game will be. Because the challenge provided by a game is typically multi-faceted, it is difficult for the player to estimate reliably the particular challenge level that is appropriate for herself. Furthermore, generally only a limited set of discrete difficulty settings is available (e.g., easy, normal, and hard). This entails that the available difficulty settings are not fine-tuned to be appropriate for each player. In recent years, researchers have developed advanced techniques for balancing the challenge level of games. Demasi and Cruz [8] used coevolutionary algorithms to train game characters that best fit the challenge level of a human player. Hunicke and Chapman [12] explored challenge balancing by controlling the game environment (i.e., controlling the number of weapons and power-ups available to a player). Spronck *et al.* [26] investigated three methods to adapt the difficulty of a game by automatically adjusting weights assigned to possible game strategies. Indeed, knowledge on the effect of certain game adaptations can be utilised to maintain a certain challenge level (e.g., [2]), and may be incorporated to steer the procedural generation of game content (e.g., [9]).

In our research, we follow the insight that (1) the challenge provided by a game is multi-faceted (i.e., a multi-dimensional action space), and (2) searching for an optimal policy in this space can be effectively guided by expert domain knowledge.

[1]Abstractly speaking, we consider the game space to be the representation of the game mechanics via which the user interacts with the game. Typically, in a video game setting, the game space will be the levels in which the game takes place. Game space personalisation, in this context, generally concerns the online learning of the next set of level design parameters.

Game space adaptation
Game space adaptation concerns allowing the space in which the game is played to adapt, ideally in response to the user experience [9]. Straightforward forms of game space adaptation have been incorporated in games such as Rogue, Diablo, Torchlight, Spore, and MineCraft. Game space adaptation in response to the actual user experience, generally in the context of procedurally generated games, is an active area of research [9, 16, 25, 30].

In our research, we will enhance a procedural process that generates levels of the Infinite Mario Bros. game such, that it is able to generate the upcoming parts of the level (i.e., level segments) online, *while the game is still in progress*. The idea here is to have assessments on the experienced player challenge impact the procedural process.

Player modelling
Player modelling is of increasing importance in modern video games [10]. The main reason is that player modelling is almost a necessity when the purpose of AI is 'entertaining the human player' rather than 'defeating the human player' [31]. A challenge for player modelling in video games is that models of the player have to be established (1) in game environments that generally are relatively complex, (2) with typically little time for observation, and (3) often with only partial observability of the environment [3]. A fairly recent development with regard to player modelling, is to automatically establish psychologically or sociologically verified player profiles [3]. Such models provide motives or explanations for observed behaviour. A solid profile can be used to, for instance, predict a player's affective state during play of the game. One contributor to this line of work is Yannakakis *et al.*, who investigated modelling the experience of game players, focusing particularly on utilising the models for the purpose of optimizing player satisfaction [32].

In our research, given these recent advances, we posit that adequately generated player models can be employed to automatically personalise the space of a video game, such that the experienced challenge is tailored to the individual player.

3. DOMAIN DESCRIPTION

In our research we consider a typical video game: Infinite Mario Bros. [18]. The game is an open-source clone of the classic video game Super Mario Bros., which can be regarded an archetypal platform game; despite its relatively straightforward appearance it provides a diverse and challenging gameplay experience. The game and its mechanics are extensively described in [13]. In our research, we build upon a version of Infinite Mario Bros. that has been extended to be able to procedurally generate entire Mario levels. To our knowledge, these extensions have been made by Pedersen *et al.* [16, 17], Shaker *et al.* [23, 24], and Togelius *et al.* [29]. The extended versions of Infinite Mario Bros. have been been employed in several Mario AI Championships, e.g., [13, 22, 29].

We have further enhanced the Infinite Mario Bros. version that was used for the Mario AI Championship 2011, to be able to procedurally generate *segments* of Mario levels while the game is still in progress. One segment has a width of 50 game objects, and generally takes a player approxi-

mately 15 seconds to complete. This enhancement enables feedback on the observed player experience to rapidly impact the procedural process that generates the upcoming level segments. Specifically, in our enhanced version, the upcoming level segments are generated seamlessly, such that no screen tears occur when the user is transitioning from one segment to the next (i.e., before the next segment can be observed a short 'gap' block is injected in the game space). This process is schematically illustrated in Figure 1.

Here, the AI challenge is to (a) solely on the basis of in-game observations on the human player interacting with game space of the current level segment, generate (b) the next segment such that it is appropriately challenging to the observed player. Specifically, the agent that personalises the game space is input with a vector of 45 real-numbered features values – the standard logging capability of the game's data recorder – that describe abstractly the observed player behaviour (e.g., how many coins did the player pick up in the segment, how many turtles did the player kill, how much time did it take the player to complete the segment) [21]. The only action that the personalisation agent can take, is output a vector of six integers $\{1..5\}$ to the procedural process which in turn generates the next level segment. The output parameters reflect the probability of the game space containing (1) flat surfaces, (2) hills, (3) tubes, (4) jumps (of increasing difficulty), (5), cannons, and (6) enemies.

4. APPROACH

The goal of our approach is to online generate game spaces such that the spaces optimise player challenge for the individual player. Our contribution to this end – derived from the design criteria discussed below – is proposing a clear separation between (intelligent) offline exploration and (safety-conscious) online exploitation.

Our approach to challenge balancing for personalised game spaces consists of three phases. In Phase 1, we learn a *global safe policy* across users for initialising the game, while generating a set of training instances (offline). In Phase 2, we learn a *feedback model* across users from the generated set of training instances (offline). In Phase 3, we *personalise the game space* to the individual player (online).

Design criteria

Two main challenges persist for the task of challenge balancing in personalised game spaces, namely (1) only *implicit feedback* on the appropriateness of the personalisation actions are available, and (2) there is a *high risk of user abandonment* when inappropriate game personalisation actions are performed. We propose the following four design criteria to address these challenges.

Intelligently generate a set of training instances

To provide an effective basis for online personalisation, it is necessary to intelligently generate a set of training instances (Phase 1). Indeed, given the employed six-dimensional parameter space, it is infeasible to acquire training instances that cover the entire space of the investigated personalisation problem (particularly when employing human participants to generate the instances). As such we propose to sequentially acquire instance data from those data points in the parameter space that maximise the upper-confidence bound on the expected experienced challenge across users. This is achieved via the Gaussian Process Upper Confidence Bound (GP-UCB) algorithm [27], a Bayesian approach to global optimization that captures its uncertainty about the fitness function in the form of a distribution, represented using a Gaussian process, over the space of possible fitness functions. This distribution is used to select the next point to evaluate based on the principle of Upper Confidence Bounds [1], which focuses the search on points that are either highly promising or highly uncertain. The result of evaluating the selected point is then used to update the distribution using Bayes rule, and the process repeats.

Learn feedback model

To address the problem of implicit feedback, we propose to learn a *feedback model* from the intelligently generated set of training instances (Phase 2). Recall that in our problem domain, the only input which the personalisation agent receives, are observations on the player interacting with the generated game space (e.g., how successful is the player in killing enemies of a particular type). As a result, for game personalisation, a mapping from gameplay observations to player experience needs to be learned. To this end, while generating the set of training instances, we gather labels on the player's gameplay experience with regard to the dependent variable (DV) 'experienced challenge'[2]. On the basis the labelled training instances, we train a random forest decision-tree classifier [11] for classifying the experienced challenge of an observed gameplay observation O. The random forest classifier consists of a combination of tree classifiers where each classifier is generated using a random vector sampled independently from the input vector, and each tree casts a unit vote for the most popular class to classify an input vector [5]. The advantage of using random forest classification, is that it does not solely output a classification, but returns a probability distribution of the classification, which we will employ as a reward signal for game-space personalisation (Phase 3).

Smart cold-start initialisation

To address the problem of user abandonment in actual play of the game, we propose to initialise a new game with a challenge level that is most appropriate in expectation across users. We achieve smart cold-start initialisation by learning a global safe policy while intelligently generating the set of training instances, with the GP-UCB algorithm. The learned global safe policy is learned in Phase 1, and employed in Phase 3.

Guided exploration

Also, to address the problem of user abandonment in actual play of the game, we propose to guide state-space exploration by using *expert domain knowledge* and a *model of user abandonment* (Phase 3). With regard to a model of user abandonment, we consider a personalisation action's expected reward (derived from the feedback model's classification of appropriateness of the action to the individual user), as inversely correlated to the probability of the player abandoning the game. This design decision has as consequence that with a high probability of abandonment we will permit

[2]In accordance with the work of Shaker *et al.* [21], we also gather labels on the DV's engagement and frustration. These DV's will be employed for multi-objective learning in future work.

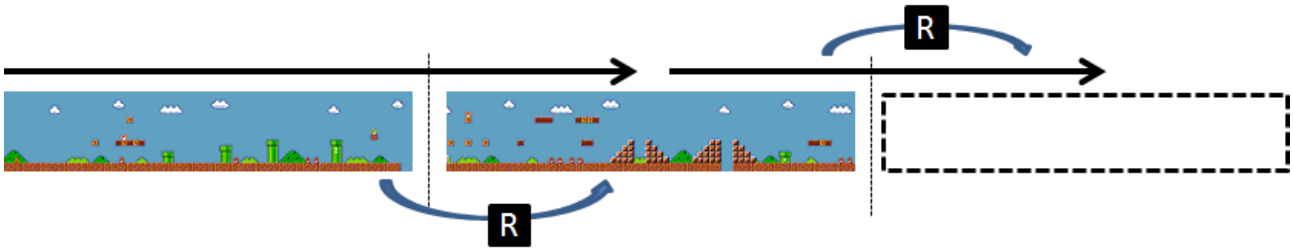


Figure 1: Our version of INFINITE MARIO BROS. It is enhanced such that during play of the game, it generates short *new level segments* of specific content *on-the-fly*, on the basis of an implicit reward signal (given by an off-line learned feedback model).

the algorithm to perform a relatively exploratory action (assuming that if the inappropriate state persists, the user will abandon in any case). On the other hand, with a low probability of abandonment we will restrict the algorithm to performing a relatively exploitative action (assuming that the algorithm may be approaching a global maximum). With regard to guiding the exploration process by domain knowledge, we acknowledge that if *a priori* it is known what effect certain action features have on the overall player experience, then this knowledge can be exploited intelligently. As such, we will let expert domain knowledge *guide the exploration* of the state-space so that it is much more effective. Specifically, the domain knowledge *dk* is formalised as a set of rules that define how, and under which conditions, a specific action feature should be adapted. Here, we posit that such expert domain knowledge is available by the designer of a game, or can be computationally derived from gamelogs.

Phase 1 – Learn the global safe policy across users, while intelligently generating a set of training instances (offline)
The procedure adopted for learning the global safe policy, while intelligently generating a set of training instances, is illustrated in Figure 2a. O indicates a gameplay observation (a vector 45 real-numbered features values), P indicates the parameters used for generating a level segment (a vector of six integers $\{1..5\}$), L indicates the user-provided label of the gameplay experience (one of five possible Likert classes), and R indicates the reward signal that can be derived from the the user-provided label L.

We employ the GP-UCB algorithm via the following general procedure. First, we start at a heuristically determined prior (the prior is represented by a Gaussian process, the Gaussian process, in turn, is parametrised by a length scale; in our six-dimensional parameter space of integers $\{1..5\}$, we adopted a length scale of 1). Second, we query GP-UCB for a new point P in the parameter space. Third, we procedurally generate a level segment on the basis of the selected parameter values P. Fourth, after a player has completed the level segment, we query the player on a label L of the experienced challenge, and use a label-to-reward mapping to calculate a reward R on the basis of the label L. Fifth, we store in the set of training instances the gameplay observations O, the parameters P that were used to generate the observed level segment, the label L on the challenge experienced by the user, and the reward R that was derived from the label L. Sixth, we feed the parameters P (it being the explored point in the parameter space) and the reward R into the Gaussian process in order to compute the posterior mean μ and standard deviation σ for some appropriately chosen constant β (where β is chosen automatically by the algorithm). Seventh, the posteriors are used to update the global safe policy GSP; it maximises the upper-confidence bound on the expected experienced challenge of a level segment. Finally, we repeat from the second step. After the desired number of iterations, we query the GP-UCB algorithm to return the data point that maximizes the posterior mean μ (i.e., the best result on average).

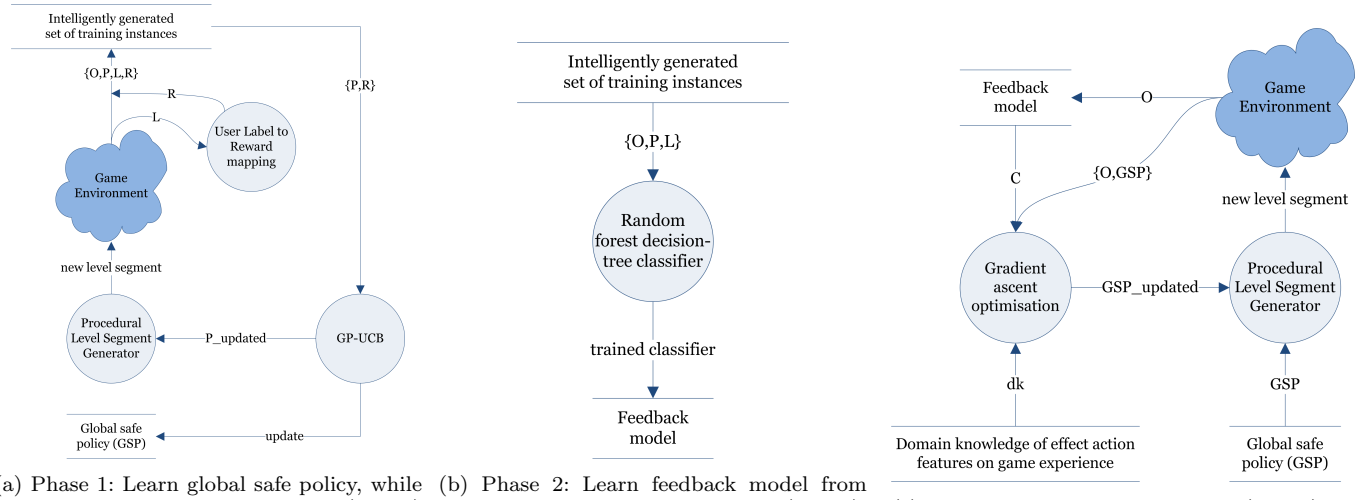
The user's label L on the experienced challenge concerns a 5-point Likert questionnaire scheme. The adopted scale for the DV challenge reads 1 = Extremely unchallenging for me, 2 = Somewhat unchallenging for me, 3 = Appropriately challenging for me, 4 = Somewhat challenging for me, and 5 = Extremely challenging for me. Given a user's label L on the experienced challenge, the reward R is calculated as follows: label 3 yields a reward of 1.0, label 2 and 4 yield a reward of 0.33, and label 1 and 5 yield a reward of 0.0.

Phase 2 – Learn the feedback model across users from the generated set of training instances (offline)
The procedure adopted for learning the feedback model is illustrated in Figure 2b. Given the intelligently generated set of training instances (Phase 1), a random forest decision-tree classifier [11] is built. The training instances concern tuples of gameplay observations O, the procedural parameters P that were used to generate the observed level segment, and the labels L of the user's experience when playing the concerning segment. We refer to the built random forest decision-tree classifier as a *feedback model*.

The feedback model can be queried to return a classification C, in the form of a probability distribution of the newly observed instance resulting from a challenging player-experience. Given that the feedback model is trained over five Likert-scale classes of challenge, the returned classification C reflects the probability of observed behaviour resulting from these five classes. We will employ this classification C as a reward signal when personalising the game space online (in Phase 3). Specifically, given the classification C, consisting of a vector $p_1..p_5$, the expectation of the reward R is defined as:

$$E[R] = \sum_{c=1}^{5} r_c p_c \quad (1)$$

where r_c is the reward value for challenge class c (the heuristic rewards defined in Phase 1, i.e., 0.0 for challenge class 1,



(a) Phase 1: Learn global safe policy, while generating set of training instances (offline)

(b) Phase 2: Learn feedback model from generated set of training instances (offline)

(c) Phase 3: Personalise game space (online)

Figure 2: Approach to game-space personalisation (legenda explained in the text).

etc.), and p_c is the probability of observed player behaviour resulting from challenge class c.

Phase 3 – Personalise the game space to the individual player (online)

The procedure adopted for game space personalisation is illustrated in Figure 2c (and listed in Algorithm 1). GSP indicates the global safe policy used for initialising the game in online play (as learned by the GP-UCB algorithm, and identical in structure to parameters P), C indicates the classification of the player experience (a probability distribution over the five Likert classes, as returned by the learned feedback model), and dk indicates the domain knowledge that defines the effect of action features on the game experience (as formalised in rules that mutate the parameter values).

The procedure is as follows. First, at the start of a game, the global safe policy GSP learned in Phase 1 is employed for generating the first level segment (it is appropriate in expectation across users). Just before completion of a level segment by the player (i.e., after having completed a segment length of 40 out of the total length of 50), gameplay observations O are fed into the feedback model. The resulting classification C by the feedback model, is input to the gradient ascent optimisation method, together with the actual gameplay observations O, the parameters GSP that were used to generate the level segment, and the domain knowledge dk on how to mutate which action feature. In the gradient ascent method, the parameters GSP are mutated into a direction that is more appropriate in expectation to the challenge experienced by the player[3]. The resulting updated GSP is fed into the procedural level segment generator. With a new level segment being generated, the process now repeats.

[3]In principle GP-UCB can be employed at this point too. However, in light of user abandonment, exploring the state-space in online gameplay is unusually expensive. As such, we adopt gradient ascent optimisation as a practical method for representing and exploiting prior domain knowledge for highly directed search in the action space.

We here discuss this procedure in some additional detail (*cf.* Algorithm 1). Given the classification C of the feedback model (Line 7), the expected reward is calculated via Equation 1 (Line 8). Using the expected reward R, together with observations O and parameters GSP, the gradient ascent optimization routine is called (Line 9). Our gradient ascent optimisation method is aimed at rapid convergence, though acknowledging potential noise in the reward signal, it probabilistically re-evaluates the champion solution (Line 17). Mutation of a solution takes place either via a regular exploratory step (Line 38), or a guided exploratory step (Line 34). Regular exploration employs a fixed step-size δ, a uniform mutation probability for all vector elements, and a uniformly randomised mutation direction. Guided exploration, however, adapts the step-size dependent on the expected reward R, and utilises domain knowledge dk to (1) specifically mutate certain vector elements more than others, and (2) reason on which vector elements should be mutated in which direction. For instance, in the case of Infinite Mario Bros., it is known that 'death by cannon bullet' is correlated to the 'number of cannons'. Consequently, one particular rule in the domain knowledge defines that when a player dies by a cannon bullet, the system should with a high probability decrease the number of cannons in the game space. As such, the guided exploration mechanism is expected to result in more rapid convergence.

We are presently performing experiments that validate the approach both in simulation studies and studies with human game players.

Acknowledgements. The authors gratefully acknowledge the contributions of George Viorel Vişniuc, Efstathios Charitos, Norbert Heijne, and Arjen Swellengrebel, who substantially contributed to the codebase of the present research project. Also, we would like to thank the following individuals for their valuable input on diverse subjects: Masrour Zoghi (GP-UCB), Anne Schuth (random forest classifiers), and both Abdo El Ali and Frank Nack (setting up user studies).

ALGORITHM 1. Game space personalisation

```
Data: GSP ← Global safe policy; O ← Observations on the player interacting
      with the game; α ← Exploitative gradient ascent step size; δ ←
      Exploratory gradient ascent step size; re ← Probability of re-evaluating
      champion; C ← Classification of user experience; E ← Expected reward
      given classification C; dk ← Domain knowledge, i.e., rules for vector
      mutation
function main()
begin
    while running do
        newLevelSegment ← generateNewLevelSegment(GSP);
        play newLevelSegment;
        if newLevelSegment has been played then
            C ← queryFeedbackModel(O);
            E ← getExpectedReward(C); (Equation 1)
            GSP ← gradientAscent(O, GSP, E);
        end
    end
end
function gradientAscent(GSP, E, O)
begin
    history.addChild(GSP, E);
    if re < random() && history.getParent() ≠ null then
        GSP_updated ← history.getChampion();
    else
        if history.getParent() ≠ null then
            if E < history.getParent.E then
                GSP ← history.getParent.GSP;
            else
                GSP ← history.getParent.GSP + α·abs(GSP -
                history.getParent.GSP);
                history.updateChild(GSP);
            end
        end
        GSP_updated ← mutate(GSP, dk, O, E, guidedExploration);
    end
    return GSP_updated;
end
function mutate(GSP, dk, O, E, guidedExploration)
begin
    if guidedExploration then
        m[] ← setSmartMutationProbabilityPerElement(dk, O);
        direction[] ← setSmartDirectionPerElement(dk, O);
        stepSize ← 1 - E;
    else
        m[] ← setUniformMutationProbability();
        direction[] ← setUniformRandomDirection();
        stepSize ← δ;
    end
    return GSP + step(m[], direction[]), stepSize;
end
```

5. REFERENCES

[1] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003.

[2] S. C. J. Bakkes, P. H. M. Spronck, and H. J. Van den Herik. Rapid and reliable adaptation of video game AI. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(2):93–104, 2009.

[3] S. C. J. Bakkes, P. H. M. Spronck, and G. van Lankveld. Player behavioural modelling for video games. *Entertainment Computing*, 3(3):71–79, 2012.

[4] S. C. J. Bakkes, C. T. Tan, and Y. Pisan. Personalised gaming. *Journal: Creative Technologies*, 3, 2012.

[5] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] D. Charles, M. McNeill, M. McAlister, M. Black, A. Moore, K. Stringer, J. Kucklich, and A. Kerr. Player-centered design: Player modeling and adaptive digital games. In *DiGRA*, pages 285–298, 2005.

[7] W. Darley and J. Lim. The effect of consumers' emotional reactions on behavioral intention: The moderating role of personal relevance and self-monitoring. *Psychology and Marketing*, 9(4):329–346, 1992.

[8] P. Demasi and A. J. de. O. Cruz. Online coevolution for action games. *International Journal of Intelligent Games and Simulation*, 2(3):80–88, 2002.

[9] J. Dormans and S. C. J. Bakkes. Generating missions and spaces for adaptable play experiences. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):216–228, sep 2011. Special Issue on Procedural Content Generation, DOI 10.1109/TCIAIG.2011.2149523.

[10] J. Fürnkranz. Recent advances in machine learning and game playing. *ÖGAI-Journal*, 26(2):19–28, 2007.

[11] T. K. Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

[12] R. Hunicke and V. Chapman. AI for dynamic difficulty adjustment in games. In *AAAI Workshop on Challenges in Game Artificial Intelligence*, pages 91–96. AAAI Press, Menlo Park, California, USA, 2004.

[13] S. Karakovskiy and J. Togelius. The Mario AI benchmark and competitions. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(1):55–67, 2012.

[14] P. D. Molyneux. The future of game AI - Lecture. Imperial College London, London, UK, 2006. October 4, 2006.

[15] J. K. Olesen, G. N. Yannakakis, and J. Hallam. Real-time challenge balance in an RTS game using rtNEAT. In P. Hingston and L. Barone, editors, *Proceedings of the IEEE 2008 Symposium on Computational Intelligence and Games (CIG'08)*, pages 87–94, 2008.

[16] C. Pedersen, J. Togelius, and G. N. Yannakakis. Modeling Player Experience for Content Creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(2):121–133, 2009.

[17] C. Pedersen, J. Togelius, and G. N. Yannakakis. Modeling player experience in super mario bros. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 132–139. IEEE, 2009.

[18] M. Persson. Infinite mario, 2009. [Online] Available: http://www.mojang.com/notch/mario.

[19] R. Petty and J. Cacioppo. Issue involvement can increase or decrease persuasion by enhancing message-relevant cognitive responses. *Journal of personality and social psychology*, 37(10):1915, 1979.

[20] M. Riedl. Scalable personalization of interactive experiences through creative automation. *Computers in Entertainment (CIE)*, 8(4):26, 2010.

[21] N. Shaker, S. Asteriadis, G. N. Yannakakis, and K. Karpouzis. A game-based corpus for analysing the interplay between game context and player experience. In *Affective Computing and Intelligent Interaction*, pages 547–556. Springer, 2011.

[22] N. Shaker, J. Togelius, and S. Karakovskiy. 2010 Mario AI competition, 2010. [Online] Available: http://www.marioai.org/.

[23] N. Shaker, G. N. Yannakakis, and J. Togelius. Crowd-sourcing the aesthetics of platform games. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012.

[24] N. Shaker, G. N. Yannakakis, and J. Togelius. Digging deeper into platform game level design: session size and sequential features. In *Applications of Evolutionary Computation*, pages 275–284. Springer, 2012.

[25] G. Smith, M. Treanor, J. Whitehead, and M. Mateas. Rhythm-based level generation for 2D platformers. In *Proceedings of the 2009 International Conference on the Foundations of Digital Games, Orlando, FL*, pages 175–182, 2009.

[26] P. H. M. Spronck, I. G. Sprinkhuizen-Kuyper, and E. O. Postma. Difficulty scaling of game AI. In A. E. Rhalibi and D. Van Welden, editors, *Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAMEON'2004)*, pages 33–37. EUROSIS-ETI, Ghent University, Ghent, Belgium, 2004.

[27] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[28] C. Teng. Customization, immersion satisfaction, and online gamer loyalty. *Computers in Human Behavior*, 26(6):1547–1554, 2010.

[29] J. Togelius, S. Karakovskiy, and R. Baumgarten. The 2009 mario ai competition. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[30] J. Togelius, M. Preuss, and G. N. Yannakakis. Towards multiobjective procedural map generation. In *Proceedings of the 2010 International Conference on the Foundations of Digital Games, Monterey, CA*, page #3, 2010.

[31] H. J. Van den Herik, H. H. L. M. Donkers, and P. H. M. Spronck. Opponent modelling and commercial games. In G. Kendall and S. Lucas, editors, *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (CIG'05)*, pages 15–25, 2005.

[32] G. N. Yannakakis, M. Maragoudakis, and J. Hallam. Preference learning for cognitive modeling: a case study on entertainment preferences. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 39(6):1165–1175, 2009.