Automatic Game Parameter Tuning and Playtesting via Active Learning

anonymous

ABSTRACT

fill me

Game designers often use human playtesting to gather feedback about game design elements when iteratively improving a game. Playtesting, however, is expensive: human testers must be recruited, playtest results must be aggregated and interpreted, and changes to game designs must be extrapolated from these results. We argue that active learning techniques can be used to formalize and automate a subset of playtesting goals. Specifically, we focus on the low-level parameter tuning required to balance a game once the mechanics have been chosen. Through a case study on a shoot-'em-up game we demonstrate the efficacy of active learning to reduce the amount of playtesting needed to choose the optimal set of game parameters for a given (formal) design objective. This work opens the potential for additional methods to reduce the human burden of performing playtesting for a variety of relevant design concerns.

Categories and Subject Descriptors

pick

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

pick

Keywords

pick

1. INTRODUCTION

update top part w/FDG info

Iterative game development practices emphasize the centrality of playtesting to improving and refining a game design. Human playtesters provide valuable feedback and playtesting is often claimed to be "the single most important activity a designer engages in" [4]. Test data informs designs of how real players may react to the game in ways that self-testing, simulations, and design analysis may not. Playtesting, however, is expensive—developers must recruit players, devise design experiments, collect game play and subjective feedback data, and make design changes to meet design goals.

We ask the question: can we reduce the cost of the playtesting process and automate some of the more mundane aspects of playtesting? To address this problem we focus on a subset of playtesting questions focused on "parameter tuning." Parameter tuning involves making low-level changes to game mechanic settings such as character movement parameters, power-up item effects, or control sensitivity. Games based on careful timing and reflexes depend on well-tuned parameters, including racing, platforming, shoot-'em-up, and fighting game genres. Addressing the problem of parameter tuning requires a means to automatically select a set of potentially good parameter settings, test those settings with humans, evaluate the human results, and repeat the process until a pre-defined design goal is achieved.

Our central insight is to model playtesting as a form of active learning [16]. Active learning involves selecting among a set of possible inputs to get the best output while minimizing the number of inputs tested. We define the "best output" in terms of a parameter tuning design goal and treat a setting for a game design as an "input." This paper makes XX contributions toward machine-driven playtesting:

1. A formulation of efficient playtesting as an active learning problem
2. A definition of a set of playtesting goals in terms of active learning metrics
3. A case study of a shoot-'em-up game demonstrating the efficacy of active learning to reduce the number of playtests needed to optimize (1) difficulty-related and (2) control-related game parameters

Unlike prior work in dynamic difficulty and adaptive games we focus on the case of deciding on a fixed design for future use. Our approach can be applied to online game adjustments to more rapidly converge on the right set of game parameters. Unlike prior work on game design support tools using simulations or model-checking we focus on the problem of efficiently working with a set of human testers. Our ap-

Foundations of Digital Games ???

proach complements these tools for early-stage testing with late-stage refinement.

In the remainder of the paper we first compare our approach to related efforts at game design support. We next define a set of parameter tuning design goals and relate them to active learning approaches. Following this we describe a case study of automated playtesting using a shoot-'em-up game. After describing the game used we present results showing the efficacy of active learning to model a known simulated player model and human data collected from an online study. We conclude with a discussion of the limitations, applications, and potential extensions to this playtesting approach.

2. RELATED WORK

Two research areas are closely related to automated playtesting: online game adaptation and offline game design tools. Online game adaptation involves adjusting game elements in real time during play to meet given design goals. Example design goals are in-game performance ("dynamic difficulty adjustment") and subjective preference ratings ("preference learning"). Offline game design tools enable designers to explore possible game designs by defining a high-level space of games through a design language. Examples include support for generating or evaluating design in terms of hard constraints (what must be true) or soft optimization (what to achieve to the best degree possible).

2.1 Online Game Adaptation

Online game adaptation researchers have used both hand-crafted rules and data-driven techniques. Hunicke and Chapman [7] tracked the average and variance of player damage and inventory levels and employ a hand-crafted policy to adjust levels of enemies or powerups. Systems by Magerko et al. [11], El-Nasr [15], and Thue et al. [18] model players as vectors of skills, personality traits, or pre-defined "player types" and select content to fit players using hand-crafted rules. Hand-crafted rules enable designers to describe fine-tuned details of how to adjust a game toward design goals. However, designers must fully describe how to tune the game and rules are often sensitive to minor changes in game settings.

To bypass the brittleness of rules others have employed data-driven techniques that optimize game parameters toward design goals. Hastings et al. [6], Liapis et al. [9] and Yu and Riedl [21] model player preferences using neuroevolutionary or machine learning techniques, respectively, and optimize the output of these models to select potential game parameters. Harrison and Roberts [5] optimize player retention and Zook and Riedl [23] optimize game difficulty using similar techniques.

Automated playtesting extends these approaches by providing principled methods for guiding the process of designing hand-crafted rules or optimizing game parameters. When hand-crafting rules, automated playtesting informs the choice of which rule parameters to use. When optimizing models learned from data, automated playtesting informs the choice of which next set of parameters to test during the optimization process. We argue that research to date has ignored the problem of reducing "sample complexity"—the number of data points (human tests) needed to train a model. Active learning makes explicit the tradeoff in playtesting between "exploring" potentially valuable game design settings and "exploiting" known good solutions with small changes. Thus, active learning complement online game adaptation through reducing the number of mediocre or bad sets of game parameters players experience before arriving at good parameter settings without changing the underlying models used.

2.2 Offline Game Design Tools

Offline game design tools have evaluated game designs using simulated players and formal model-checking. Simulation-based tools use sampling techniques to test aspects of a game design for "playability," typically defined as the ability for a player to reach a given goal state with the current design parameters. Model-checking tools define game mechanics in a logical language to provide hard guarantees on the same kinds of playability tests.

Bauer et al. [1] and Cook et al. [**?**] use sampling methods to evaluate platformer game level playability. Shaker et al. [**?**] combine a rule-based reasoning approach with simulation to generate content for a physics-based game. Simulation approaches are valuable when design involves an intractably large space of possible parameters to test and can serve as input to optimization techniques. Active learning can enhance simulation approaches by guiding the sampling process toward spaces of parameters likely to be of greater value.

Model-checking approaches provide guarantees on generated designs having formally defined properties—typically at the cost of being limited to more coarse design parameter decisions. Smith et al. [**?**], Butler et al. [**?**], and Horswill and Foged [**?**] employ logic programming and constraint solving to generating levels or sets of levels meeting given design constraints. Jaffe et al. [**?**] employ game-theoretic analysis to understand the trade-offs of game design parameters for balance in a competitive game. Our approach to automated playtesting is intended to complement these approaches to high-level early design exploration with low-level optimization of game parameters and tuning. Further, we focus on a generic technique that applies to cases with human testers in the loop, crucial to tuning game controls or subjective features of games.

3. ACTIVE LEARNING

Our goal is to reduce the playtesting burden by efficiently choosing game designs to test on players. Active learning (AL) provides a generic set of techniques to guide the playtesting process of choosing a set of design parameters to test toward achieving a design goal. Playtesting typically involves tradeoffs between testing designs that are poorly understood (exploration) and refining designs that are known to be good but need minor changes (exploitation). Active learning captures this intuition through explicit models of the exploration-exploitation tradeoff. In the following sections we will define active learning in the playtesting context and provide the intuition behind several common active learning methods. Our aim is to characterize playtesting in terms of active learning; full mathematical treatments are available through the references.

3.1 Playtesting as Active Learning

Active learning techniques take a model of how a design works and choose how to change that design to achieve a design goal. Formally, the design goal is known as the *objective*

function and captures the relationship between input (game design parameters) and desired output (game design goals). An *acquisition function* takes information from the objective function and uses it to decide on the next set of inputs to test. Objective functions come in two forms: (1) regression and (2) classification models. *Regression* models capture continuous outputs—e.g. how the rate of enemy firing in a shoot-'em-up influences the number of times a player is hit or how the height of jumps in a platformer influences how long it takes players to complete a level. *Classification* models capture discrete outputs—e.g. whether a player preferred the current control sensitivity compared to the previous set of controls in a racing game or which choice a player made in a choose-your-own adventure. Acquisition functions differ for these two tasks and we will treat each in turn. Note that at this high level formulation a wide variety of design goals can be formulated as goals for playtesting—the primary challenge lies in defining a useful metric for measuring these goals through player feedback or in-game behavior.

3.2 Regression Models

We consider four acquisition functions for regression models: (1) variance, (2) probability of improvement, (3) expected improvement, and (4) upper-confidence bounds. The *variance* acquisition function picks designs based purely on the uncertainty of the design goal value [2]. Intuitively, variance encodes a playtesting strategy of attempting any design that is poorly understood until the entire design space is well-understood. Variance approaches are pure exploration and are useful when design information on many possibilities is needed.

Probability of improvement (PI) improves on uncertainty by selecting the design that is most likely to gain some improvement of the best design observed so far [2]. Intuitively, probability of improvement is a playtesting strategy of attempting the design that seems most likely to improve over what has been tried so far. Probability of improvement approaches are pure exploitation and are effective when performing minor tweaks to already-effective designs.

Expected improvement (EI) enhances PI by weighting the probability of each design improvement by the amount of improvement [2]. Unlike probability of improvement or variance, EI balances between exploration and exploitation. Intuitively, EI seeks designs that are likely to have large improvements over the existing design. By balancing exploration and exploitation EI is a more general-purpose technique useful for playtesting during many stages of the development cycle.

Upper Confidence Bound (UCB) methods select designs based on the upper bound on design quality according to a model of design quality and variability in that quality [**?**]. Designs that are expected to be high quality but are also poorly understood (and thus highly variable) are given priority. By testing these designs UCB narrows down the space of designs to those known to be good or unknown but likely to be bad. In the UCB approach cited above the importance of variability is gradually scaled down as more playtests are run. Intuitively, this capture the notion of initially exploring a variety of possible designs and gradually converging to exploit the most promising designs.

3.3 Classification Models

We consider three acquisition functions for classification

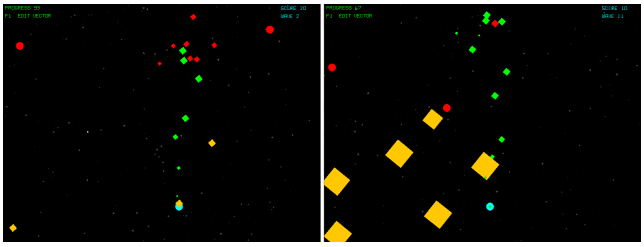

Figure 1: Experimental game interface illustrating player, enemies, and shots fired by both at two points along adaptation process. Preference learning experiments varied parameters related to player movement; enemy tuning involved the speed, size, and rate of fire of enemy bullets.[[Each image full-column-width, top and bottom]]

models: (1) entropy, (2) query-by-bagging voting, and (3) query-by-bagging probability. *Entropy* acquisition functions choose designs that are most uncertain according to entropy [16]. Entropy measures the amount of information needed to encode a distribution. Intuitively, this captures how uncertain a model is about the possible (discrete) outcomes. Entropy sampling acts similarly to variance-based sampling for regression.

Query-By-Bagging (QBB) approaches emphasize disagreement among potential models [16]. First, multiple copies of the same objective function are trained using random selections of the playtest data gathered so far. This captures the notion of guessing how the model might be if slightly different sets of playtests had occurred. Second, these models predict the outcome of new playtests. The final choice of design to test is the one these models most disagree on. Intuitively, this captures the idea of picking designs that are most likely to be poorly understood.

Two varieties of QBB are relevant: QBB vote and QBB probability. In *QBB vote* each objective function model votes on outcomes, giving a single choice. When only two options are possible (e.g. controls were better or worse) the design with the greatest disagreement between two binary outcomes is chosen. When multiple options are possible (e.g. a multiway branch in a choose-your-own-adventure), the design with the greatest difference between the top two options is chosen. In *QBB probability* each objective function model estimates how probable each outcome is, giving a weight to all choices. The weights of all models are averaged and the point with lowest probability is chosen. [[**find citation**]]

find citation

4. GAME DOMAIN

We developed a simple shoot-'em-up game to do a case study of active learning for playtesting. Shoot-'em-up games emphasize reflexes and pattern recognition abilities as a player maneuvers a ship to dodge enemy shots and return fire. In general, arcade games serve as an ideal starting domain for low-level parameter tuning:

- There are a number of parameters that can potentially interfere with each other: size and speed of enemies and enemy bullets, rate of enemy fire, player speed, player rate of fire, etc.

- The game can be played in a series of waves, enabling our system to naturally test game parameter settings and gather player feedback.

- Action-oriented gameplay reduces the complexity of player long-term planning and strategizing.

- A scoring system makes gameplay goals and progress clear, unlike domains involving puzzle-solving or aesthetic enjoyment of a game world or setting.

In the case of shoot-'em-up games, we tested two different kinds of game design goals: player game play behavior goals—e.g. player success rates or score achieved—and subjective responses goals—e.g. getting good user ratings on the feel of the controls. To test active learning for regression we set a design goal of the player being hit exactly six times during each wave of enemies and tuned enemy parameters. To test active learning for classification we set a design goal of the player evaluating a set of controls as better than the previous set and tuned player control parameters.

We implemented a shoot-'em-up arcade game based on space ship combat, shown in Figure 1. During each wave a series of enemies appear that fire bullets at the player. For enemy tuning we varied the: size of enemy bullets, speed of enemy bullets, and rate that enemies fire bullets. Increasing bullet size requires the player to move more carefully to avoid bullets. Faster bullets require quicker player reflexes to dodge incoming fire. More rapid firing rates increase the volume of incoming fire. Together these three parameters govern how much players must move to dodge enemy attacks, in turn challenging player reflexes. Getting approximate settings for these parameters is easy, but fine-tuning them for a desired level of difficulty can be challenging.

For control tuning we varied two ship movement parameters: drag and thrust. Drag is the "friction" applied to a ship that decelerates the moving ship at a constant rate when it is moving—larger values cause the ship to stop drifting in motion sooner. Thrust is the "force" a player movement press applies to accelerate the ship—larger values cause the ship to move more rapidly when the player presses a key to move. Combinations of thrust and drag are easy to tune to rough ranges of playability. However, the precise values needed to ensure the player has the appropriate controls are difficult to find as player movement depends on how enemies attack and individual player preferences for control sensitivity (much like mouse movement sensitivity). After wave of enemies a menu asked players to indicate if the most recent controls were better, worse, or as good/bad as ("neither") the previous set of controls. We provided a fourth option of "no different" for when players could not distinguish the sets of controls, as opposed to "neither" where players felt controls differed but had no impact on their preferences.

5. EXPERIMENTS

Our experiments sought to test whether the active learning framework could reduce the number of human playtesters needed to tune design parameters. In the experiments we used the active learning acquisition functions given above for tuning enemy parameters or tuning player controls. Both experiments involved a preliminary test on simulated data followed by testing with human participants. The simulation experiments allowed us to verify that our methods would work in principle; humans studies show our method can apply to real-world contexts.

In each study we employed three different objective functions—Gaussian Processes (GP), kernel support vector machines (KSVM), and optimized neural networks ("neuro-evolution", NE). We chose these objective functions as they cover several player modeling approaches used in this domain: GPs are a common model in active learning contexts [2], kernel methods (particularly KSVMs) are a popular machine learning technique previously used in player modeling [22], and optimized neural networks have been widely used in preference learning [**?**].[1] All three of these objective functions can be trained in regression or classification modes.

Our results show active learning can reduce the number of human playtests needed. For enemy paramter tuning (a regression problem) we find variance, UCB, and EI are all effective in simulation, but only UCB and EI are effective on human data. This is likely due to human players having more variability in playing behavior than our simulated models.

For control tuning (a classification problem) we find QBB vote and entropy have competitive performance in simulation. On human data GPs show improved performance using QBB probability with few samples while KSVMs show improved performance using entropy with greater samples; NE showed little difference among techniques used. Our control tuning results suggest a generic set of controls all players agree is good is difficult to obtain. Combined across experiments, these results show active learning has promise for helping to automate low-level parameter tuning.

In the following sections we detail our experimental design for both the simulated players and human participants. We then discuss the results of our tests in detail.

5.1 Simulation Methods

We devised two simple models of how players might respond to different design paramters to verify our active learning approach on ground truth information. Our regression model treats players as having an underlying set of skills related to each enemy tuning parameter along with a cross-skill rate of making errors. Greater differences between player skills and enemy parameters lead to larger differences from being hit at a base rate. Our classification model treats players as having preferences for each of the control tuning parameters along with a cross-parameter tolerance for differences from preference. Preference choices are based on the difference between the ideal set of parameters and design control settings.

Our regression model is a probabilistic model of player behavior in terms of underlying player skills and design parameters. Playtest players have three independent skills for enemy bullet size, bullet speed, and firing rate; all three skills are sampled from a normal distribution with a variance term capturing variability in skill. Taking the difference between the player-specific skills and the design parameters, then scaling by the error in player skills produces an estimated rate of being hit by enemies in our game. When training simulated models we generated a fixed ideal playtester and

[1]For computational reasons we employ a simple gradient-based optimization method, rather than the more common neuro-evolutionary approaches. We did not find any performance differences between the two optimization approaches in initial tests in our domain.

allowed the active learning model to choose sets of enemy parameters along a 10 x 10 x 10 grid.

Our classification model is a probabilistic model of player responses in terms of underlying control preferences and design parameters. Playtest players have two independent preferences for force and drag parameters; both are sampled from a normal distribution with a variance term capturing variability in preference. When given a set of design parameters the model performs a two-stage comparison process. First, each individual design parameter is compared to the desired parameter for the model by taking a cumulative normal distribution centered at the difference of the parameters and scaled by the player variance term. Differences below the player error threshold yield a "no difference" result; positive or negative differences above the threshold yield "better" or "worse" responses, respectively. Second, the individual parameter responses are combined into the final model response. If both responses are the same then that response is given. If one response is "no difference" then the other response is given. Otherwise the model resonds with "neither." When training simulated models we generated a fixed ideal playtester as before and varied sets of controls using a 5 x 5 x 5 x 5 grid. Each test point consisted of both current and previous wave control parameters.

Using these simulation models we performed 10-fold cross-validated experiments to measure how accurately an objective function could predict playtester output given a set of design parameters. In each cross-validation fold we generated a set of 30 initial randomly sampled design parameter settings and used the simulated models to generate playtester responses. These settings and responses formed the initial training data set. An additional set of 100 testing points was generated in the same way. We then created the grid of points the active learning model could sample, forming the training pool. Finally, we created a set of 100 randomly sampled design parameters and Within each fold we then repeated the following process:

1. Train the objective function on the training data set.
2. Test the objective function on the testing data set and measure the error.
3. Use the acquisition function to pick a point from the training pool to improve the objective function model.
4. Generate a player response for that training pool point and remove it from the training pool.
5. Return to the first step and repeat the process.

For regression we measured error as

check

the mean squared error from the desired rate of being hit. For classification we measured error as the F1 score—a combination of model accuracy and coverage of sample data.

5.2 Human Study Methods

deploy online N random samples for each criteria: min 10 waves total, only keep waves 1-10 for comparability

same test process as before but now using all data gathered as pool to sample from

6. RESULTS

overview results for fitting each model emphasis on AL, specific methods varied widely and likely to vary in other applications as well

6.1 Regression

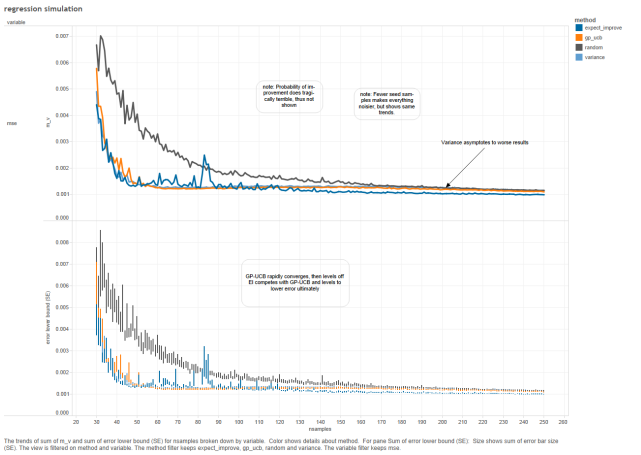

Figure 2: regression simulation; mse; down is good

all 3 methods do quite well EI asmpytotes to slightly better results

Figure 3: regression experiment; mse; down is good

UCB and EI are only ones to do well – UCB remains at consistently high performance – EI gradually gets worse over time why? – variance and PI do not capture exploration and thus have difficulty finding good results – UCB is able to temper exploration and exploitation over time, likely helping focus on best areas to test

6.2 Classification

GP random beats KSVM random -> GP is better able to model domain GP entropy shows best trend -> focus on finding out where problems are QBB models do moderately, but expensive to train -> best for high variance situations and model was not too high variance [cf Schein logreg]

GP random beats KSVM random -> GP is better able to model domain no strong winners, domain was very hard

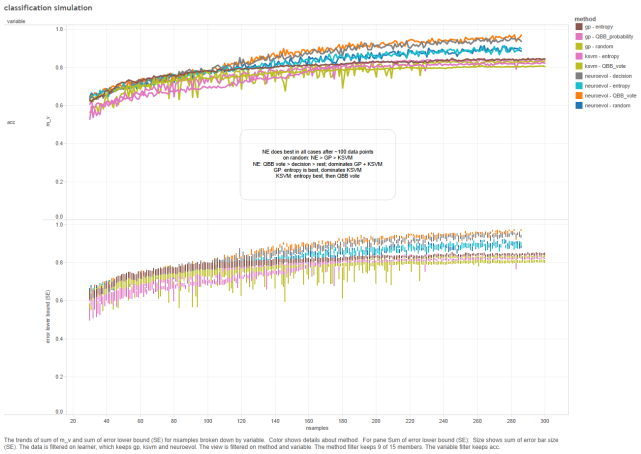

Figure 4: classification simulation; f1; up is good

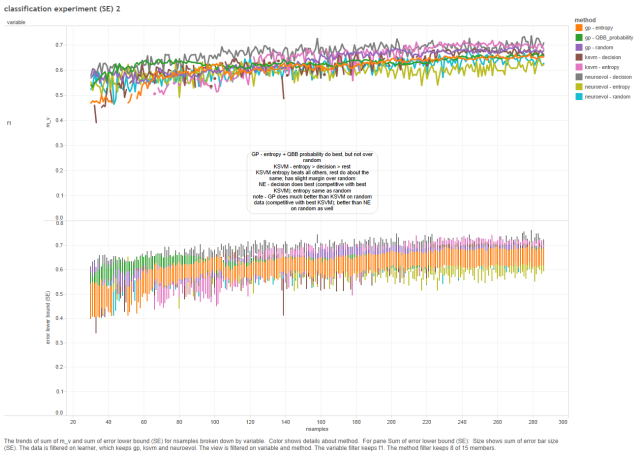

Figure 5: classification experiment; f1; up is good

KSVM entropy is slightly better QBB models don't perform –> QBB + KSVM results suggest consistent domain but complex underlying model

7. DISCUSSION

8. ACKNOWLEDGMENTS

9. REFERENCES

[1] A. W. Bauer, S. Cooper, and Z. Popovic. Automated redesign of local playspace properties. In *8th International Conference on the Foundations of Digital Games*, 2013.

[2] E. Brochu. *Interactive Bayesian optimization: learning user preferences for graphics and animation*. PhD thesis, University of British Columbia, 2010.

[3] W. Chu and Z. Ghahramani. Preference learning with gaussian processes. In *Proceedings of the 22nd International Conference on Machine learning*, pages 137–144. ACM, 2005.

[4] T. Fullerton, C. Swain, and S. Hoffman. *Game design workshop: a playcentric approach to creating innovative games*. Morgan Kaufmann, 2008.

[5] B. Harrison and D. L. Roberts. Analytics-driven dynamic game adaption for player retention in scrabble. In *IEEE Conference on Computational Intelligence in Games*, pages 1–8. IEEE, 2013.

[6] E. Hastings, R. K. Guha, and K. Stanley. Automatic content generation in the galactic arms race video game. *IEEE Transactions on Computational Intelligence and AI in Games*, 1:245–263, 2009.

[7] R. Hunicke and V. Chapman. AI for dynamic difficulty adjustment in games. In *AAAI Workshop on Challenges in Game Artificial Intelligence*, 2004.

[8] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

[9] A. Liapis, H. P. Martınez, J. Togelius, and G. N. Yannakakis. Adaptive game level creation through rank-based interactive evolution. In *IEEE Conference on Computational Intelligence in Games*, volume 4, pages 71–78. Springer, 2013.

[10] D. J. Lizotte. *Practical bayesian optimization*. PhD thesis, University of Alberta, 2008.

[11] B. Magerko, B. Stensrud, and L. Holt. Bringing the schoolhouse inside the box - a tool for engaging, individualized training. In *25th Army Science Conference*, 2006.

[12] O. Missura and T. Gärtner. Player modeling for intelligent difficulty adjustment. In *Discovery Science*, pages 197–211. Springer, 2009.

[13] C. Pedersen, J. Togelius, and G. N. Yannakakis. Modeling player experience in Super Mario Bros. In *IEEE Symposium on Computational Intelligence and Games*, 2009.

[14] C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, MA, 2006.

[15] M. Seif El-Nasr. Interaction, narrative, and drama: Creating an adaptive interactive narrative using

performance arts theories. *Interaction Studies*, 8(2):209–240, 2007.

[16] B. Settles. *Active learning*, volume 6. Morgan & Claypool Publishers, 2012.

[17] N. Shaker, G. N. Yannakakis, and J. Togelius. Towards automatic personalized content generation for platform games. In *6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2010.

[18] D. Thue, V. Bulitko, M. Spetch, and E. Wasylishen. Interactive storytelling: A player modelling approach. In *3rd AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2007.

[19] A. Tversky and D. Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131, 1974.

[20] 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.

[21] H. Yu and M. O. Riedl. Personalized interactive narratives via sequential recommendation of plot points. *IEEE Trans. Computational Intelligence and AI in Games*, forthcoming, 2013.

[22] H. Yu and T. Trawick. Personalized procedural content generation to minimize frustration and boredom based on ranking algorithm. In *7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2011.

[23] A. Zook and M. O. Riedl. A temporal data-driven player model for dynamic difficulty adjustment. In *8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.