Interactive Bayesian Optimization for Game Mechanics

Abstract

fill me

Game design often involves a final phase of substantial fine-tuning of game mechanics. Paradigmatic examples include varying the settings of player character movement parameters, altering opponent combat statistics, or varying low-level parameters around movement and collision of game objects. Tuning is often a time-consuming and expensive process due to the need for human testing (rather than analytic solutions or simulation results) and the large space of possible parameter settings for a game. We ask: can an artificially intelligent (AI) system act as a designer focused on the problem of parameter tuning? Can such a system acquire design knowledge of low-level parameter settings that achieve design goals tailored to individual players?

Our central insight is to formulate an iterative design process as a kind of active learning with players responding to an AI-chosen game tuning. An interactive Bayesian Optimization model for active learning makes explicit the trade-offs between improving a design and testing alternatives, while developing a model of the design space of a given game for later (re)use. From this perspective, a design space is a multi-dimensional domain of possible game parameter values mapping to expected player behavior (and associated levels of uncertainty). A design space might specify how well players are expected to perform in a first-person shooter in response to varying levels of AI opponent accuracy or how players rate quests in an adventure game.

Developing this design space formalization with an AI system enables game designers to specify abstract objectives for a game design—achieving a given level of player performance in a game or getting good ratings for provided content—rather than remain tethered to manually making minor low-level changes. An AI abstraction affords better automation of the process and just-in-time tailoring of game settings to individual players. The formal design space models learned can support game designers in considering the realm of possible design configurations a given game affords through visualization. Further, designers can leverage our overall approach to develop formal models of other aspects—beyond low-level tuning—of the game design process.

A game design AI system synthesizes elements from several existing paradigms for the use of AI in game design. Procedural content generation examines how an AI can create content within a space of parameters. Dynamic difficulty adjustment concerns itself with adapting game mechanics in real time to changes in users. By contrast, we focus on the problem of learning a model of how game mechanics impact player behavior. Unlike player modeling, our goal is an AI system that actively adjusts game mechanics to explore the design space for that game both to achieve a given design objective and understand how that design space works. Rather than replace game designers, a low-level game design AI system allows for a new kind of design process where an AI system automatically player-tests games and intelligently searches for designs that meet a designer-specified goal. Building a model of the game design space allows the system to transfer learning from users across multiple design objectives.

We employ interactive Bayesian optimization in an AI system to choose parameter settings that are most informative about the design space while achieving given design goals for player behavior in an online human-in-the-loop iterative process (Brochu 2010). By explicitly modeling the trade-off between exploring very different designs and exploiting designs similar to existing ones, our approach reduces the need for extensive amounts of play-testing while also automating the tuning process. Employing non-parametric models of the game design space (here Gaussian Processes) we demonstrate the application of interactive Bayesian optimization to two cases studies of game design tuning in a shoot-'em-up game: (1) adjusting enemy design parameters to achieve a desired level of player success and (2) optimizing controls to player preferences.

We make four primary contributions:

1. formulating game design tuning as an active learning process
2. showing the improved performance of this approach over standard randomized ("A/B") testing methods in terms of better modeling with the same quantity of data
3. applying this approach to optimize enemy designs for a design objective on player performance

4. applying this approach to optimizing and learning player preferences for control settings

For enemy design optimization we show how a designer-specified objective function for player performance statistics can guide building a regression model from enemy parameter settings to desired player performance. To optimize controls we use preference learning to select control settings to test and evaluate against the previous set of controls. In both studies, Bayesian optimization affords automatic exploration-exploitation trade-offs that enable rapidly (globally) optimizing for the design objective (player performance or preference).

First, we discuss related work in game tailoring and adaptation. Second, we motivate and describe our interactive Bayesian optimization approach, detailing the Gaussian process regression and preference learning models. Third, we describe our shoot-'em-up game and two empirical human studies demonstrating the efficacy of our approach. We conclude by discussing extensions and the range of applications of this modeling approach.

Related Work

Automating low-level game design tuning relates to techniques for adapting content based on player behavior or preference information. Approaches to game tailoring and adaptation combine a player modeling technique with a content adaptation or generation method. Many early efforts employed a game-specific player model using vector of attributes (e.g. skills or use of content) and reactively selected new content to guide players toward a desired level of skill or intended level of performance. Hunicke and Chapman (2004) track the average and variance of player damage and inventory levels and employ a hand-crafted policy to adjust levels of enemies or powerups. (Magerko, Stensrud, and Holt 2006), (El-Nasr 2007), and (Thue et al. 2007) model players as vectors of skills, personality traits, or pre-defined "player types" and select content to fit players using rule-based approaches. In contrast, an interactive Bayesian optimization approach generalizes across games, automatically determines how complex the player model should be, and does not require designers to tune a set of rules (or other parameters) to generate content while retaining designer control over the algorithm objectives.

Evolutionary computing and machine learning provide frameworks for modeling players and optimizing game parameters to achieve adaptation goals. (Hastings, Guha, and Stanley 2009) track player use of weapons and use neuro-evolution to generate new weapons based on those expressed preferences. (Pedersen, Togelius, and Yannakakis 2009) employ neuro-evolution to model reported frustration, challenge, and fun with a multi-layer perceptron. (Shaker, Yannakakis, and Togelius 2010) use this model to generate platformer game levels through an exhaustive search process. (Yannakakis, Maragoudakis, and Hallam 2009) employs the same techniques in an augmented reality game setting, attempting to optimize player satisfaction. (Missura and Gärtner 2009) create player clusters from training data and train a linear regression models for game difficulty within each cluster in an arcade game. During adaptation, new player clusters are predicted from early play data using a support vector machine (SVM) and difficulty is set using the cluster-specific regression model. (Yu and Trawick 2011) perform a similar process of player modeling and adaptation by clustering players and optimizing game parameters employing a large-margin SVM approach that minimizes player boredom or frustration in a level-based action game.

Unlike these approaches we integrate feature selection into the model selection process, automatically balance model complexity with fit (rather than requiring parameter tuning), and automatically trade off exploration of possible parameter settings against exploitation (rather than only exploiting through optimization). Conceptually, the interactive Bayesian optimization approach captures a full design tuning process of testing many possible parameter settings, incrementally updating an understanding of the design space, and selectively seeking points in the space that will be most informative to improving the design space model while still meeting design objectives. We seek an approach that is intentionally attempting to acquire design knowledge relevant to designer-specified goals, without requiring a designer have deep familiarity with either the player modeling or generation algorithms. In (Khaled, Nelson, and Barr 2013)'s framework our system combines the analysis of gameplay data of a PCG EXPERT with solving design tasks (adaptation) as a DESIGNER ASSISTANT.

Game Domain

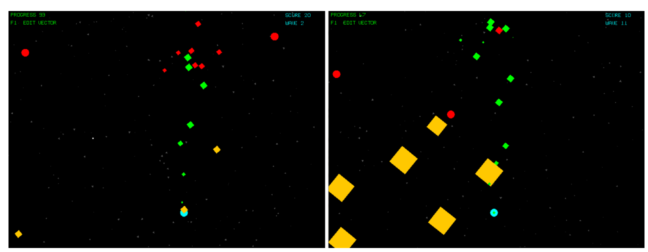

Figure 1: Experimental game interface illustrating player, enemies, and shots fired by both at two points along adaptation process. Preference learning experiments varied parameters related to player movement; enemy tuning involved the speed, size, and rate of fire of enemy bullets.

We test the interactive Bayesian Optimization model in a shoot-'em-up arcade game (Figure 1). Shoot-'em-up games emphasize reflexes and pattern recognition abilities as a player maneuvers a ship to dodge enemy shots and return fire. An arcade-style game enables our system to use a series of waves to test game parameter settings and gather feedback from the player, iteratively refining the parameter settings to a given design objective. Further, the action-oriented gameplay reduces the complexities of player long-term planning and strategizing, serving as an ideal starting domain for low-level tuning. In our game players gain points for defeating enemies and lose points when hit by enemies. Players are encouraged to avoid enemy fire by dodging enemy attacks or destroying enemy ships to (temporarily) reduce the number of incoming attacks.

During each wave a series of enemies appear that fire bullets at the player. In our experiments, we varied three parameters governing enemy behavior: 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. As with controls, getting approximate settings for these parameters is easy, but fine-tuning them for a desired level of difficulty can be challenging. Our performance adaptation experiment had the system set enemy bullet size, speed, and firing rate to minimize the difference between player performance and designer-specified target performance.

Player controls are limited to shooting at enemies and moving their ship. Ship movement is governed by two controllable 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). Our preference-learning experiment had the system set drag and thrust for the player while attempting to maximize player preference for a set of controls.

Interactive Bayesian Optimization

We use interactive Bayesian Optimization (IBO) to model an iterative game design process informed by playtesting. IBO is itself a form of sequential Bayesian optimization where data points are player feedback gathered in interaction of AI with the user (Brochu 2010). Sequential Bayesian optimization optimizes a function through iteratively testing a sequence of points, each one selected by some algorithm based on previous points. Two functions are involved: (1) the *objective function* that maps inputs to outputs; and (2) the *acquisition function* that maps potential inputs to their value for optimizing the objective function output. In our application we employ Gaussian processes (GPs) for the objective function—the game design model from parameters to player behavior—and a modified expected improvement acquisition function—trading exploration of the design space against exploitation of designs known to achieve the design objective.

Gaussian processes are a non-parametric modeling technique able to capture complex non-linear relationships within a data set, automatically adjust model complexity to data, and integrate out parameters without user intervention (Rasmussen and Williams 2006). Intuitively, non-parametric models allow an infinite number of variables to account for the data before selecting only the subset needed to explain a given set of observations. In practice, this leads to models that automatically become more complex to fit a data set as needed. Bayesian formulations of GP regression and classification automatically trade off between model complexity and fit to a data set, avoiding over-fitting and poor generalization problems that occur with other optimization approaches. Bayesian model specifications allow parameters of the model to be integrated out, simplifying their use by requiring less user specification. We employ GPs to leverage the benefits of non-linear mapping from inputs to outputs, automatic complexity adjustment with data collection, and reduced or eliminated parameter specification from users.

Below we describe the standard formulations of GP regression and GP preference learning and then integrate GP models with active learning methods for a full IBO model. We demonstrate how the IBO approach can use existing GP models to automate design tuning and learning a design space model. Gaussian process regression enables automatic difficulty adjustment by modeling player performance in a game as a non-linear function of game parameters. Gaussian process preference learning enables optimization of game parameters (here controls) to player preferences by modeling player preferences for a set of game parameters as a non-linear function of game parameters that is forced to pairwise choices between alternatives. Active learning uses a GP objective function to identify the next parameter settings to test, guided by a designer-specified acquisition function—here expected improvement—for parameter adjustment. For game performance, designers specify a goal of achieving a given level of in-game performance; for controls, designers specify optimal player preference.

Gaussian Process Regression

Gaussian processes are formally defined as "a collection of random variables, any finite number of which have a joint Gaussian distribution" (Rasmussen and Williams 2006). While allowing an infinite number of variables to be used, any GP model can be computed through a multivariate Gaussian distribution based on the input and output values. Gaussian processes are specified by their mean function ($m(x)$) and covariance function ($k(x, x')$):

$$f(x) \sim GP(m(x), k(x, x'))$$

Intuitively, GP regression learns a model predicting that similar inputs—according to the covariance function—should yield similar outputs. Different covariance functions define different notions of similarity. In our work we employ the automatic relevancy detection (ARD) version of a squared exponential distance:

$$k(x, x') = exp\big(-\frac{1}{2}\Sigma_{l=1}^{d}\kappa^{l}(x^{l} - x'^{l})^{2}\big)$$

where $\kappa^l > 0$ is the ARD parameter for the l-th feature of a d-dimensional data set, serving to control the contribution of this feature to the model. Automatic relevancy detection allows us to optimize model parameters during the fitting process, automatically scaling input dimensions to minimize the impact of irrelevant aspects of the data. Mathematical properties of the GP mean that an initially zero valued mean function will taken on non-zero values after fitting data, allowing the model to be initialized with zero as the mean value (see (Rasmussen and Williams 2006) for additional details on GP regression). In our case we use such a zero-mean GP.

For our performance regression model we predict player performance (number of times hit) from game parameters controlling enemy attacks (speed and size of bullets along with firing rate). We fit a GP regression model to player data and optimize the covariance function ARD parameters using stochastic gradient descent after each training point received. Since GP regression has a closed-form solution for learning and prediction the primary computational bottleneck is optimizing the ARD parameters (needing 13 seconds for 10 training points on a standard desktop machine).

Gaussian Process Preference Learning

We employ a pairwise preference learning model rather than using preference rating scales due to human biases. Sequential numeric ratings are subject to a cognitive anchoring bias where earlier numeric ratings influence choices on subsequent ratings (Tversky and Kahneman 1974). We thus employ a model that generalizes information gained from pairwise rankings to the underlying preference of users for different instances (here game parameter settings). Games can only be played sequentially during comparisons, motivating an approach of pairwise preference ratings comparing each new instance to the previous one.

Gaussian process preference learning models user choices in a two-part model: (1) a GP regression model specifying the underlying (unobserved) value of a single instance; (2) a probit model of how a choice is generated based on two instances being compared (Chu and Ghahramani 2005). The GP model allows a flexible specification of how users value a given instance specified in terms of its parameters. The probit model—known in economics as the Thurstone-Mosteller law of comparative judgment—converts a pair of latent values into a comparison judgment according to the function:

$$P(x_i \succ x_j | f(x_i), f(x_j) = \Phi\left(\frac{f(x_j) - f_(x_j)}{\sqrt{2}\sigma_{noise}}\right)$$

where x_i, x_j are two instances, $f(x_i)$ is the GP latent value of an instance, Φ is the cumulative normal distribution, and σ_{noise} is the inherent noisiness of comparative judgments. Intuitively, the probit model encodes preference judgments as based on the difference in underlying value of two instances, allowing for noise in preference ratings.

Due to the non-linear probit model used GP preference learning has no analytic learning model. Instead, we follow work by (Chu and Ghahramani 2005) and use a Laplace approximation to learn the underlying GP model's parameters. We employ a GP with zero mean and the ARD covariance function and optimize its parameters along with the selection of σ_{noise} using a grid search over the space of possible parameters. These nested optimization processes are possible using off-the-shelf solvers, requiring 9 seconds for 10 training points. We note that optimization may be performed using any-time algorithms (such as DIRECT (Jones, Perttunen, and Stuckman 1993)), allowing optimization of parameters to occur while the player plays a new option. In our experiments we optimize parameter values and force players to wait in order to test the best-case performance of our approach.

Active Learning

Active learning (AL) is an approach to machine learning problems with a large set of unlabeled instances where a computer asks a human to provide information about given instances to learn a model of the instances as a whole (Settles 2012). AL is well suited to our application where the space of game parameterizations is very large and information can only be gained through the expensive process of having a human play and provide feedback about a game instance. Acquisition functions specify how a given AL algorithm weights potential instances to test based on a goal of optimizing the objective function. In our case, the GP regression model seeks to minimize the difference between desired and actual player performance and the GP preference model seeks to find the highest latent value instance.

Many possible acquisition functions exist, varying in how the functions balance the exploration-exploitation trade-off guiding how locally-tethered the search for large objective function values is (Settles 2012). Expected improvement (EI) is an acquisition function that balances the value of unseen instances against the uncertainty regarding their values. EI integrates over all possible results to get an average-case estimate of the result, rather than seeking a best-case (or worst-case) scenario. We employ a modified EI function that incorporates a slack parameter ($\xi \geq 0$) to control the relative weighting of exploration and exploitation goals (Lizotte 2008):

$$EI(x) = \begin{cases} (f(x) - f(x^+) - \xi\Phi(Z) + & \\ \sigma(x)\phi(Z)) & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases}$$

where $f(x)$ is the function value at x, x^+ is the instance with the current greatest function value, σx is the uncertainty in the value of the instance, $\phi(Z)$ is the Gaussian distribution density at Z and Z is defined as:

$$Z = \begin{cases} \frac{f(x) - f(x^+) - \xi}{\sigma(x)} & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases}$$

Intuitively, Z is the noise-scaled difference between the test point x and the current best point x^+, and the expected improvement takes a weighted combination of this gain against the uncertainty of the point. Points that are more uncertain and expected to have higher values are preferred to those with lower values or high values that are highly certain. ξ allows an explicit specification of how heavily to emphasize exploration

Experiment

In this section we describe two empirical human studies we conducted. The first study examines the use of GP regression to find and continually adjust game parameter settings to achieve a designer-specified level of performance. The second study examines the use of GP preference modeling to optimize controls to player preferences. We show our approach, compared to random sampling: improves a player model's predictive power with the same sampling budget, does equally well or better at achieving a design objective, and improves the ratio of positive to negative responses to sample points in preference learning. Random sampling is the common standard used to evaluate the efficacy of AL models for improving model fits for a given data budget and is similar to A/B testing approaches that capture large amounts of data before acting on the results (Settles 2012).

Methods

We used 73 sessions of performance optimization (38 active learning and 35 random sampling) and 28 sessions (14 and 14) of control optimization data. Players logged into the game online and played the game against a series of waves of enemies. Players were assigned in a balanced factorial design to either a control case that sampled random possible game parameter values or a test case that selected parameter values using interactive Bayesian optimization. We recorded player performance in terms of number of times the player was hit and all parameter settings for each wave. Player preferences were recorded in a four option forced choice pane, selecting whether the control settings of the current wave compared to the previous wave were: better, worse, no different, or different but of equal quality. For analysis we examined only results from players who completed at least ten waves and only examine data from those ten waves. Preference responses only used better or worse choices.

In the performance optimization study a GP regression model was fit to predict the difference between the actual number of times the player was hit and a desired designer-specified value. We specified a designer target of players being hit 6 times over the course of a 20 second wave of combat. Between waves the model and covariance function parameters were all optimized and EI used to select the next test point. Learning had a one wave lag—players were allowed to play a wave while the model was fit on a server machine to all data but the most recent wave. This enabled the game to play continuously without pausing for learning to occur at the cost of a model that was unaware of the most recent test point results.

In the control optimization study a GP preference model was fit to predict the underlying preference players had for different control settings. After each wave players were prompted to indicate whether the most recent wave had better or worse controls than the wave before that. Model fitting used the same timing as above. In this case EI selected an instance to test in comparison to the last tested instance—every comparison was the next point that would be most useful compared to the last level the player completed.

wave	AL	random	p-value
1	13.18	8.89	0.19
2	**8.68**	**12.57**	**0.02**
3	**3.89**	**10.86**	**0.01**
4	7.71	5.37	0.25
5	7.34	5.63	0.96
6	**13.84**	**6.80**	**0.01**
7	9.79	4.60	0.07
8	10.92	6.57	0.12
9	**12.18**	**6.00**	**0.00**
10	10.37	7.17	0.46

Table 1: Error rates of active learning (AL) and random sampling (random) for achieving target performance rate of 6 hits per wave.

Results

We evaluated our IBO model for two capabilities: (1) achieving a given design objective, and (2) modeling the design space. Achieving a design objective meant either enforcing the target player performance goal or optimizing player preferences. We evaluated this as error from the target level of performance and proportion of better or worse choices from player subjective feedback. Modeling the design space meant building a model with better generalization to new data. We evaluated this using 10-fold cross-validation—training a model on nine tenths of our data and testing it on the remaining one tenth.

For performance optimization we found our IBO approach performs equivalently to a random sampling strategy on a wave-by-wave basis (Table 1). We evaluated error as the difference between the number of times a player was hit and the target rate of 6 hits for each wave. We compared error rates between the two models using a Kruskal-Wallis rank sum test, a non-parametric alternative to the standard t-test used when data is not normally distributed (as is ours and most human subject data). These results showed the IBO approach had significantly ($p < 0.05$) smaller error on two waves, higher error on two waves, and no difference on the remaining waves. Both methods were unable to achieve the design objective, likely due to the difficulty of enforcing such a low rate of hitting players. Alternative acquisition functions may also allow for better performance on design objectives by putting heavier emphasis on exploitation over attempting new parameter values.

For preference optimization we found the IBO model had a statistically significantly greater proportion of "better" responses (and lower "worse" responses) on a wave-by-wave basis compared to the random sampling approach (χ^2 test, $p < 0.05$ for both). Players also showed significantly greater proportion of "no difference" responses (χ^2 test, $p < 0.05$) on a wave-by-wave basis. There were too few "equal quality" responses (n=44 over 10 waves) to achieve statistical significance. Together, these results show the active learning model is able to capture aspects of player preference and attempt to optimize for them.

Building a good design space model entails exploring the space in a way that generalizes to new parameter settings.

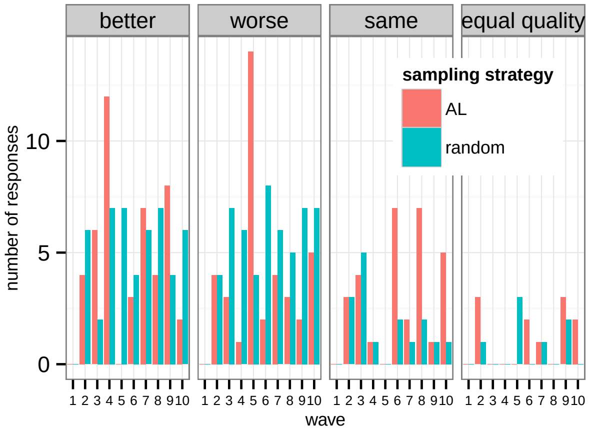

Figure 2: Count of player subjective responses to control settings in different sampling techniques.

	better		worse	
wave	AL	random	AL	random
2	4	6	4	4
3	6	2	3	7
4	12	7	1	6
5	0	7	14	4
6	3	4	2	8
7	7	6	4	6
8	4	7	3	5
9	8	4	2	7
10	2	6	5	7

Table 2: Comparison of frequency of preference ratings by sampling method.

We evaluated this by training an active learning model or random sampling strategy on a subset of nine tenths of the data before testing on the remaining one tenth of the data. We used the following standard method for evaluating active learning models: (1) set aside a random selection of 10% of our data for testing; (2) train either random sampling or active learning on 100 sample points of the 657 points in the remaining 90% of our data; (3) test the trained model on the test 10%; (4) repeat the above for 10 repetitions (Settles 2012). In random sampling step (2) randomly selected training points. In active learning step (2) was seeded with 10 random points, then allowed to iteratively select one new training point to sample from the remaining points in the 657 point pool and retrained with the new set of data until 100 points were being used in the AL model. To examine the efficiency of the two approaches we compared the error (or accuracy) of the models as the number of samples increased.

For both performance and control optimization IBO more effectively finds the right parameter settings to test to build a useful model of the design space. We compared mean squared error of model predictions on the held-out test data using the Kruskal-Wallis rank sum test. For performance optimization we found the mean squared error of predictions against actual performance levels of IBO to be significantly less ($p <$???) than that of random sampling when comparing on a wave-by-wave basis. fill For control optimization we found the IBO model accuracy for predicting player choices was significantly higher ($p <$???) on a wave-by-wave basis. fill Thus, IBO uses human playtesting more effectively to automatically model a design space.

While difficult to quantify, IBO shows clear trends of exploring design alternatives (e.g., Figures 3, 4). During performance optimization IBO gradually shifts the values of enemy parameters tested, distinct from the uniform sampling approach from random sampling. During control optimization IBO shows a similar trend of trying alternatives for ship drag and thrust. These results suggest IBO is both intentionally exploring the design space and uncovering general patterns of player responses across the design space that may go unnoticed unless designers are careful to try the full range of options.

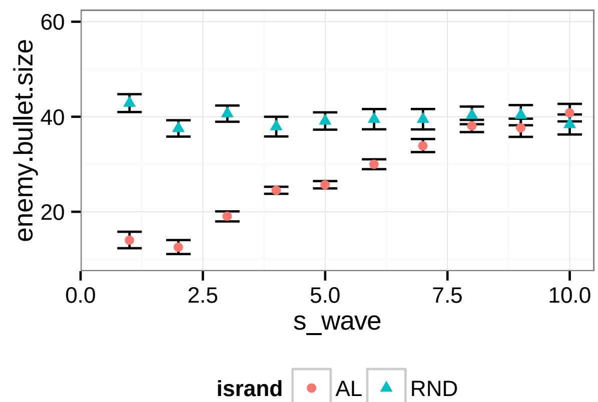

Figure 3:

Discussion

Acknowledgments

References

[Brochu 2010] Brochu, E. 2010. *Interactive Bayesian optimization: learning user preferences for graphics and animation.* Ph.D. Dissertation, University of British Columbia.

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

[El-Nasr 2007] El-Nasr, M. 2007. Interaction, narrative, and drama: Creating an adaptive interactive narrative using performance arts theories. *Interaction Studies* 8(2):209–240.

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

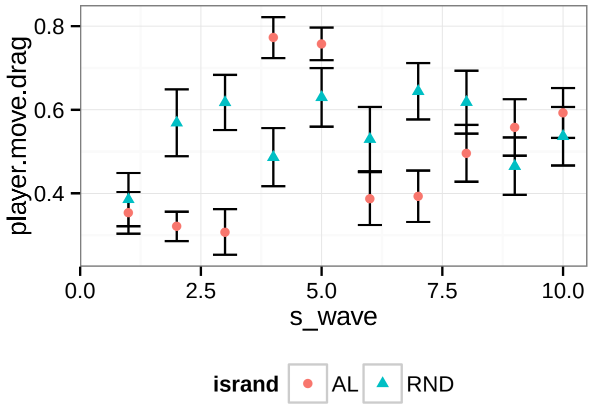

Figure 4:

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

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

[Khaled, Nelson, and Barr 2013] Khaled, R.; Nelson, M. J.; and Barr, P. 2013. Design metaphors for procedural content generation in games. In *ACM SIGCHI Conference on Human Factors in Computing Systems*.

[Lizotte 2008] Lizotte, D. J. 2008. *Practical bayesian optimization*. Ph.D. Dissertation, University of Alberta.

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

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

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

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

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

[Shaker, Yannakakis, and Togelius 2010] Shaker, N.; Yannakakis, G. N.; and Togelius, J. 2010. Towards automatic personalized content generation for platform games. In *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.

[Thue et al. 2007] Thue, D.; Bulitko, V.; Spetch, M.; and Wasylishen, E. 2007. Interactive storytelling: A player modelling approach. In *Proceedings of the Third AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

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

[Yannakakis, Maragoudakis, and Hallam 2009] Yannakakis, G. N.; Maragoudakis, M.; and Hallam, J. 2009. 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.

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