Skill-based Mission Generation: A Data-driven Temporal Player Modeling Approach

Alexander Zook, Stephen Lee-Urban, Michael R. Drinkwater, Mark O. Riedl
School of Interactive Computing, College of Computing
Georgia Institute of Technology
Atlanta, Georgia, USA
{a.zook, surban6, drinkwater, riedl}@gatech.edu

ABSTRACT

Games often interweave a story and series of skill-based events into a complete sequence—a mission. An automated mission generator for skill-based games is one way to synthesize designer requirements with player differences to create missions tailored to each player. We argue for the need for predictive, data-driven player models that meet the requirements of: (1) predictive power, (2) accounting for temporal changes in player abilities, (3) accuracy in the face of little or missing player data, (4) efficiency with large sets of data, and (5) sufficiency for algorithmic generation. We present a tensor factorization approach to modeling and predicting player performance on skill-based tasks that meets the above requirements and a combinatorial optimization approach to mission generation to interweave an author's preferred story structures and an author's preferred player performance over a mission—a kind of difficulty curve—with modeled player performance.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*Games*; K.8.0 [**Personal Computing**]: General—*Games*

General Terms

Algorithms, Measurement, Design, Human Factors

Keywords

Procedural Content Generation, Optimization, Player Modeling

1. INTRODUCTION

Games typically involve a sequence of skill-based tasks, such as combat or puzzle-solving, motivated by story. While games are often designed with a fixed progression of task difficulty, there have been calls for dynamic tailoring of difficulty on a per-player basis [13]. Dynamic difficulty adaptation is a challenging problem; given the current broad diversity of player background skills, preferences, and motivations, this matching is typically difficult or impossible to achieve with any single, fixed progression. The solution is *Procedural Content Generation* (PCG), in which a system automatically creates game content algorithmically, with or without the involvement of a human designer. *Just-in-time PCG*, in which the algorithm uses information about a player that cannot be known *a priori*, is well suited for customization of player experiences and dynamically creating numerous variations to promote replayability.

In this paper, we explore just-in-time content tailoring to dynamically adjust the difficulty of a game and provide motivating context for the adjustments provided. Specifically, we look to solve two related problems: *challenge tailoring* and *challenge contextualization*. Challenge tailoring is the problem of matching the difficulty of skill-based challenges over the course of a game to match player abilities. Challenge contextualization is the problem of providing appropriate motivating story context for the skill-based challenges. We motivate an approach to these problems with a simple role-playing game in which skill-based challenges manifest as combat with monsters (see Figure 1), which are contextualized through common role-playing game activities such as quests and interactions with non-player characters. The solution to the challenge tailoring and challenge contextualization problems in this domain is a *mission*, a sequence of events to occur in the game world, interleaving skill-based combat with story-based background motivation. Borrowing the term from Dormans [5], missions are brief "chapters" of gameplay that are composed of sequences of events that are a subset of the complete game plotline (see Figure 2).

Just-in-time content tailoring requires a model of the player and a means to use this model to adjust content to fit the player [23, 24]. We identify five criteria for player models for the purposes of content tailoring:

1. **Predictive power**—capturing expected player behavior or experience when given a known task or situation
2. **Accuracy**—correctly inferring player behavior or experience with high confidence, particularly when faced with partial/missing player data or small amounts of player data; transferring information between players is also desirable
3. **Efficiency**—scaling to large sets of data: fine-grained, long-term, and including many players

Foundations of Digital Games Procedural Content Generation Workshop 2012 Raleigh, NC, USA

Figure 1: Example battle between player team (right) and monsters (left).

...
enter(town)
arrest(guards, companion)
battle(guards)
reveal-history-foe(companion, wizard)
...
enter(palace)
search(castle, wizard)
guard(acid-monsters, wizard)
battle(acid-monsters)
capture(wizard)
...
meet(king)
ask-rescue(king, princess)
find-clue(princess)
enter(dungeon)
battle(ice-beasts)
free(princess)
...

Figure 2: Example mission.

4. **Generative sufficiency**—usable in generative models as parameters or evaluation methods; contrasted with abstract class labels or theoretical constructs

5. **Temporal**—capturing changes in player behavior and experience over the course of experiences; capable of forecasting several steps into the future and incorporating interactions among individual experiences along a trajectory

Many existing PCG techniques have used theory-driven models guided by designer intuition or a qualitative model of player experience, rather than data-driven models based on empirical data connecting the effect of content on player responses [21]. This has limited these efforts to domains where players are well-understood and cleanly fit into particular *a priori* known categories. When such categories and theories are lacking, we believe that tailored content generation systems require data-driven player models that incorporate bottom-up information from players into the system design. Accounting for temporal variations in player models (e.g. learning over time or changes in preferences) can further enhance generation by predicting player changes in the future and creating sequences of content tailored to these expected changes.

We propose a tensor factorization approach based on collaborative filtering as a data-driven means of modeling and predicting player performance on skill-based events such as computer game combat. This approach captures temporal variations in the relationship between player abilities and task features in influencing player performance, for building player skill models meeting the above five criteria. Our player model is incorporated into a mission generation algorithm, based on a genetic algorithm, that balances tailoring of content to players with designer-specified content requirements and preferences. Our system is thus capable of producing a variety of missions tailored for any given player while still meeting high-level designer intentions.

2. RELATED WORK

Research on generating or adapting games to players requires both generating content and modeling players. Procedural content generation (PCG) research has developed systems to construct or adapt content—including puzzles, platformer game levels, racetracks, and game stories—from a given set of domain content using a variety of algorithms (for reviews, see [21, 24]). For PCG in the role-playing game genre, work has been done to generate missions and levels [5, 8, 10, 18]. Most systems, however, have relied on theory-driven player models based on designer experience or qualitative theory that lack direct connections to the behavior of players during play.

Player modeling research has investigated techniques to extract player preferences or skills given their activities in a game (e.g. [1, 14, 22]). For example, Thue, et al. [20], Seif El-Nasr [6], and Magerko [11] model players of an interactive story using vectors of various archetypal player classes. Other researchers have applied data-driven techniques including evolutionary computing and machine learning methods to model players. Pederson et al. [14] collect preferences from players of a platformer game through a questionnaire and train a neural network to predict player emotional states based on player behavior and game features. Weber, Mateas, and Jhala [22] model player retention using an ensemble of regression algorithms and a ranking of features according to their individual impact on player retention. Harrison and Roberts [7] model the temporal relationships among acquiring achievements in a massively-multiplayer online role-playing game using correlations between achievements for different players at different times. Yu and Riedl [25] use a collaborative filtering approach to predict preferences for subsequences of a choose-your-own-adventure story. Our player modeling approach differs from those above in that it is both data-driven *and* incorporates temporal variations in player performance—that is, our technique, based on tensor factorization, allows us to model change in a player's skill over time.

Educational data mining (EDM) researchers have explored a variety of models to capture player skills and preferences in the context of learning tasks (for a review, see [4]). While we focus on games for entertainment, the fact that players learn a set of skills makes EDM relevant. Among EDM approaches to player modeling, collaborative filtering techniques are most relevant to the current discussion as they meet the five requirements of predictive power, accuracy with partial data, efficiency with large data sets, sufficiency for algorithmic input, and accounting for temporal change. Recent advances have extended these models to handle temporal factors [9, 19]. Mapping to standard collaborative filtering terminology, players are treated as users, particular tasks they execute (e.g. a mathematics problem) as items, and the performance on the task as the rating.

Mission generation shares much in common with story and quest generation, in which a system autonomously produces a linear or branching sequence of events to play out in the game world. Generation of sequential content has often been approached as a planning problem, in which the system searches for a sequence of operations to transform a given domain from an initial state to a goal state, with the final sequence being the generated content. Porteous, et al. [15] developed a system that organizes a set of given key elements of a story along a dramatic arc and subsequently fills gaps between these events using a planning system. Li and Riedl's [10] system uses partial order planning to adapt quest plans to a set of player-specified requirements for events to include or exclude from an initial quest.

Alternative approaches to content generation for stories have explored the use of machine learning and evolutionary computation techniques. Roberts et al. [16] used a reinforcement learning technique—targeted-trajectory distribution Markov Decision Processes—to direct an agent to construct appropriate stories from a space of possible stories given sequences of player actions. Sorenson and Pasquier [18] combine a feasible-infeasible 2-population genetic algorithm with constraint satisfaction techniques to generate challenge-based game levels. As with the planning systems above, these efforts are based on *a priori* known models of player experiences: TTD-MDP's rely on author specified distributions, and Sorenson and Pasquier's challenge metric is predefined by an author and tuned using level exemplars independent of player performance. We describe a combinatorial optimization approach—using a genetic algorithm—that integrates given author-specified evaluation criteria with player performance-derived criteria. We advance previous work by incorporating dynamic models of players into the generation process while also leveraging additional knowledge of how players learn over time.

3. TAILORING GAME MISSIONS

Our goal is to generate missions—sequences of skill-based challenge events and story events. This requires solving both the *challenge tailoring problem* and *challenge contextualization problem.* The challenge tailoring problem requires finding a sequence of skill-based challenge events that produces a given progression of predicted player performance. To determine whether a particular sequence of challenges is appropriate, the game designer specifies a *performance curve*—a progression of player performance over the course of a mission—to guide generation. For example, a performance curve describing a reduction of player performance over time can create the feeling of increasing challenge. Designers may specify any form of performance curve desired, representing arbitrary shapes such as fluctuating peaks and valleys or a rise and subsequent drop in performance.

In missions, skill-based challenge events, such as battles, do not exist in isolation from the other aspects of game play. The challenge contextualization problem is the creation of story content that motivates game play in between challenges, sets up challenges, and varies the game play to increase replayability. The fragments of a full mission shown in Figure 2 contain numerous story events that contextualize the battles. Challenge contextualization is a simplified form of story generation. Whereas story generation provides causally linked narratives that incorporate a sequence of required events and end with a given consequence, challenge contextualization provides interesting interactions with characters and short-term quests to motivate the player's activities without concern for a coherent overarching narrative. In challenge contextualization, contextualization events—non-skill-based events—are selected and instantiated from a set of known possible event types.

We demonstrate our challenge tailoring and challenge contextualization system on a custom game designed to test our algorithms. The game consists primarily of a sequence of battles in the model of traditional turn-based role-playing games (RPGs). Battles are interspersed with periods where players can interact with NPCs and go on quests that serve as opportunities to set up and contextualize the battles. Our game domain was constructed to focus on deliberative tasks during combat such that players are forced to make the best choice of action for the given situation while having limited time to consider the options available. Battles consist of a sequence of discrete turns taken by characters on a player team and enemy team. On each turn, the player (or enemy) character has a set of available actions—magical spells—that they may select among (see Table 1). The player's team consists of four characters each with a different subset of spells they may use; the player has 10 seconds to cast a spell before his or her turn is skipped. Each battle presents a different enemy team of four enemies.

Each spell is associated with a *type* that defines its effectiveness against opponents of a particular type. For example, a fire type spell is effective against an ice type character and ineffective against a water type character. We vary spell effectiveness into three levels: super-effective, effective, and ineffective. Table 1 presents the effectiveness matrix of spell types cast on particular character types. As an example, a player may battle an enemy ice beast. In this case, since the ice beast is of the ice type, casting a fire spell is super-effective, casting an earth spell is merely effective, and casting a lightning spell is ineffective. Each player character in the game was designed to have a partially overlapping set of spells, while each battle is generated with varying sets of enemy types. This situation forces players to make choices based on what is available to them and learn which spells are most effective in a given situation. Spell naming was intentionally obscure in order to require players to learn the effectiveness matrix over play, rather than using pre-existing knowledge of game conventions. The player's skill at the game is in the form of knowing which spell to choose for each of his or her teams' characters to perform against any given enemy.

Each mission consists of a sequence of 10 battles. Each battle yields data on each spell the player cast on each turn, enabling us to model player proficiency at selecting the best available spell in each situation. The battle interface is shown in Figure 1. Our game system allows us to record: which spells a player casts, how long they spend deciding to

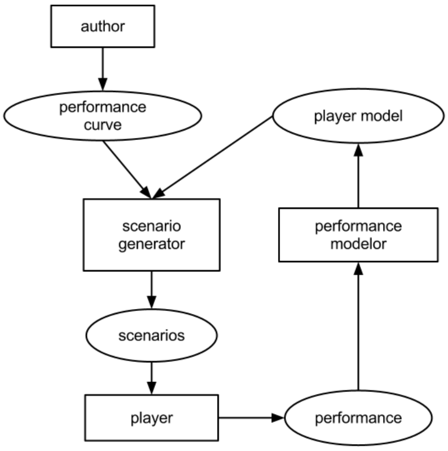

Figure 3: Framework for model training and mission generation.

cast a spell, which opponent they target, the battle and turn number in which the action was taken, and the effectiveness of that spell.

4. PLAYER MODEL

Solving the challenge tailoring problem requires a model of the player and a set of desired performance requirements. We focus on the player modeling task of predicting player performance in future battles given a player's history.

We approach this task using collaborative filtering, specifically tensor factorization. Collaborative filtering, which learns to make predictions from similarities across a large number of people, is a well-known technique for making accurate predictions with relatively little data about any one particular individual. It thus meets our first four player model requirements: predictive power, accuracy, efficiency, and generative sufficiency. A key strength of collaborative filtering techniques is combining data across users: the model is able to make predictions for players in battles they have not experienced, battles in the future that a smaller subset of players have experienced, and when given limited information about the player (e.g. due to the player recently starting). Our model can always make predictions for a player, with initial predictions based more heavily on task-dependent factors and later predictions incorporating player-specific information as it is gained.

Tensors generalize matrices to add additional dimensions to the matrix structure, moving from the two-dimensional arrangement of a matrix to higher orders. Tensors that add a time dimension extend collaborative filtering to make predictions about how a player is expected to change in the future, thus handling the fifth criteria, temporality. As a player modeling approach, we use tensor-based collaborative filtering to predict player performance for different possible parameterizations of battles across time, thus modeling skill improvement.

Our framework for combining player modeling and mission generation is presented in Figure 3. The mission generator takes as input both predicted player performance and a designer-specified performance curve, such as that shown in Figure 4. Battles with appropriate parameterizations are selected to match to the desired performance level at the desired point in time and sequenced to produce a full progression of player performance. Players then play through a desired mission, yielding performance ratings that the performance modeler uses to update the player model for the next round of mission generation.

While we employ our model's predictions to generate full missions, these results can also be employed for predicting player performance over single events and making adaptations to an existing mission or incrementally building a mission. For predictions of single events tensor factorization models temporal shifts in player performance over time and thus provides a more nuanced view of player performance than the static models build by traditional matrix factorization methods and related approaches. While our current approach employs offline training and adaptation recent work in machine learning has developed techniques for online training for matrix factorization that would apply to our problem [12]. We first present our matrix factorization approach to player performance modeling, then describe our genetic algorithm approach to mission generation.

4.1 Matrix Factorization

Matrix factorization techniques model player performance by considering the relationships between players and tasks with respect to the observed performance ratings. Performance ratings are decomposed into sets of latent factors describing underlying features of players and tasks. Task features describe a latent space of factors possessed by a task. Player features describe a latent space of capabilities of players at those task factors. Predictions of performance are made based on these latent factors using an inner product [19].

Formally, we have sets of players ("users") U, tasks ("items") I, and performance scores P. Data on player performance for a given task is collected in a $U \times I$ matrix with entries corresponding to the performance of player u on task i. In the time-varying case (three-dimensional tensor) we add a dimension to the matrix corresponding to the time of observations T. The resulting tensor $Z = U \times I \times T$ is a tensor of player u's performance on task i at time t. This tensor is decomposed into a set of factors according to:

$$Z \approx \sum_{k=1}^{K} \lambda_k w_k \circ h_k \circ q_k$$

where $\circ$ is the outer product, λ_k are positive weights on the factors, w_k are player factors, h_k are task factors, and q_k are time factors. K is the rank of approximation made by the decomposition, keeping the top set of most important components of the factors. Prediction on future tasks becomes:

$$\hat{p}_{uiT^*} = \sum_{k=1}^{K} w_{uk} h_{ik} \Phi_{T^*k}$$

where $\hat{p}_{uiT^*}$ is predicted performance of player u on task i at the current time T^* and:

$$\Phi_{T^*k} = \frac{\sum_{t=T^*-L}^{T^*-1} q_{tk} p_t}{L}$$

with L defining the number of previous time steps to use for performance. Φ_{T^*k} are averaged performances of players over the last L times performing this task, based on

Table 1: Spell Effectiveness Matrix

Attack ↓ Defense →	fire	water	acid	ice	lightning	earth	force	undeath
fire	1	0	1	2	1	2	1	0
water	2	1	0	1	0	1	2	1
acid	1	2	1	0	1	0	1	2
ice	0	1	2	1	2	1	0	1
lightning	1	2	1	0	1	0	1	2
earth	0	1	2	1	2	1	0	1
force	1	0	1	2	1	2	1	0
undeath	2	1	0	1	0	1	2	1

the factors describing the time of the task and the observed performance. Future performance is predicted by taking a weighted sum of the latent factors describing the player, task, and averaged performances. The weight factors are derived using root mean square error for optimization by stochastic gradient descent.

Prediction can be improved by including bias terms to account for baseline features of both the players and tasks. This alters the model to:

$$\hat{p}_{uiT^*} = \mu + b_u + b_i + \sum_{k=1}^{K} w_{uk} h_{ik} \Phi_{T^*k}$$

where μ is the global average performance on a task, b_u is a bias term encoding the proficiency of a player, and b_i is a bias term encoding the difficulty of a task. The player and task biases are computed as the averaged performance differences from the global average over all players or tasks, respectively.

In our game we collect player data on actions taken during combat. Each data point consists of a tuple (u, i, t, p) where u is the player, i is the task, t is the turn number in the full play trace, and p is the performance. i is recorded as a concatenation of: the battle in the full sequence, enemy being attacked, number of times that enemy has been fought, and the particular spell being cast. As an example, casting an earth spell on an ice beast fought in the third battle with this being the second ice beast encountered would be recorded as "3-ice_beast-2-earth". Performance ratings are based on the defined spell efficacy matrix above – super-effective, effective, and ineffective are mapped to 2, 1, and 0, respectively. For the above case, the full tuple may be (player01, 3-ice_beast-2-earth, 15, 1) when player01 performs the earth spell as her 15^{th} action and where casting earth is effective and thus scored 1. Given this data we model and predict player performance based on the tensor factorization techniques outlined above.

To bypass an early period of identical predictions across players we begin all players with a fixed initial mission that serves both as a tutorial and initial source of player data. The player model is trained using this data and existing data from all other players and then used to predict player performance and generate a mission. The model is subsequently updated after a player completes each mission and used to generate the next mission. We anticipate reducing training frequency when data sets become large to avoid substantial delays in mission generation. Reducing frequency of model updates is acceptable as larger data sets enable more information to be shared between players and thus enables more accurate predictions without additional data from the target player.

4.2 Performance Modeling

In order to generate a mission our system requires a set of skill-based challenge events (combat) associated with particular skill (spell) types and a designer-specified performance curve of desired player performance over the full mission. We model player *performance*, instead of game *difficulty*. Player performance is directly observable during play in the form of the choices made in given circumstances and evaluated by the game that directly "scores" these actions. In our game domain these scores come in the form of combat attacks being super-effective, effective, or ineffective; the notion generalizes to any actions that have some implications for how well a player is doing. In a shooter game performance may be measured through player firing accuracy, in a racing game performance may be measured by time to complete track segments, or in a puzzle game performance may relate the number of moves players require to complete particular parts of puzzles.

Content difficulty is not necessarily equivalent to performance as many factors impact player performance beyond difficulty alone, such as ambiguity in the context, level of player attention or fatigue, prior player knowledge, or familiarity with interface controls. Further, difficulty itself may be multidimensional, subdividing into terms based on speed of actions, precision of action timing and execution, or complexity of action composition. To bypass these complications we focus on the task of performance prediction, where we assume some of the latent factors may implicitly describe the notion of difficulty. Our system is able to accommodate multiple performance notions through training separate tensor factorizations for these separate dimensions.

Figure 4 shows an example of a target performance curve, specifying that the designer wishes to have player performance decrease during combat over the course of the mission. The solid line depicts the author-specified desired levels of player performance over the course of the mission. This particular curve approximates a mission that will *appear* to steadily increase in difficulty. Other arcs may be used to approximate the shape of an Aristotelian dramatic arc, or create rhythms of alternating periods of low and high performance, or capture common design heuristics such as having high early performance, a middle period of low performance, and a final increase in performance that gives the player a sense of mastery. Dotted lines depict the actual performance of players over the mission and black boxes depict combat events within the mission, with intervening periods occupied by non-combat events. The particular parameterization of combat events in Figure 4 resulted in player B performing nearer the performance curve than player A.

5. MISSION GENERATION

Given both a set of predicted performance levels on skills

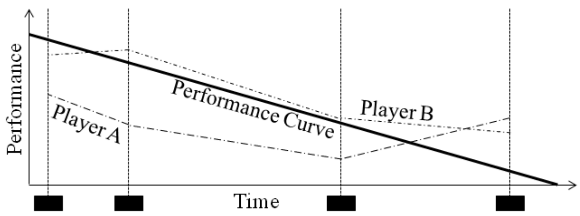

Figure 4: Illustration of desired and observed player performance. Boxes indicate skill-based (combat) events set at different times in the mission. The solid curve depicts desired performance over the course of the mission. Dashed curves depict performances for two different players on these events.

and a performance curve, the mission generator creates sequences of battles consisting of combinations of enemies meant to have players achieve particular levels of performance on particular skills. Challenge tailoring is thus the parameterization of the sequence of battles based on how closely a specific sequence matches the performance curve. Challenge tailoring involves solving the exact parameterization of each battle in the sequence and determining the timing of each battle.

Our mission generator is designed to apply to any domain where player performance can be assessed with respect to a set of concrete tasks used repeatedly in the course of a larger sequence. The full details of our mission generator are described by Zook et al. [26]. In this section we summarize the main points of how the mission generator works.

To solve the *challenge tailoring* problem a system must compose a sequence of battles that matches predicted player performance to a desired performance arc. To solve the *challenge contextualization* problem a system must intersperse story events within a given sequence of combat events, ensuring these events meet author requirements and preferences. Since challenge tailoring and challenge contextualization are highly interrelated—adding story events impacts timing of battles and thus impacts performance curve matching—we solve the two problems simultaneously.

When solving these problems we seek both to provide designers some control over generation and to create multiple distinct missions from a given set of input to meet the general PCG goal of enhancing replayability without designer effort. In this paper we focus on the task of fitting player performance to the performance arc expressing a designer's intended course of performance. Our previous work describes methods to incorporate designer control over domain content and the evaluation of story content [26].

The mission generator attempts to find the best sequence of events that incorporates the set of performance skills and performance curve, authored story content and evaluation, and player performance predictions. To meet these diverse requirements we use combinatorial optimization, specifically a Genetic Algorithm (GA). Genetic algorithms attempt to find one or more structures that maximize a given evaluation function. They are particularly suitable for problems where there are many soft requirements that describe ideal relationships between different aspects of the structure, but few binary requirements such as necessitated goal states.

A genetic algorithm starts with a population of randomly generated potential solutions—in this case missions—and attempts to modify and/or combine aspects of different members of the population to improve the fitness of the population according to the given evaluation function. Our GA performs evaluation using a combination of numerical evaluation functions with author-provided parameters and an author-provided grammar that expresses author preferences over events and compositions of events. Ensuring global properties of items is a difficult problem for GAs; we employ a planner to post-process the results produced by our GA to ensure this global coherence need, similar to the use of constraint-satisfaction approaches in other GA-based systems [18].

Challenge tailoring requires matching a designer-specified performance curve to predicted player performance. The generation algorithm takes as input a designer-specified performance curve, the set of possible elements to configure in battles and a set of player performance predictions for battle configurations when experienced at different times. From this knowledge the GA is able to evaluate each mission battle sequence created in terms of the distance between predicted player performance and designer-desired player performance in battles. The GA searches for the optimal fit between battles and desired performance by varying the parameterization and timing of battles over the mission. Scenarios are penalized using a Euclidean distance metric summing over all battles to encourage a smooth convergence toward battles meeting designer specifications.

As an example, the system may be given the performance curve in Figure 4 that specifies decreasing player performance across four battles during the mission. In addition, the system is provided knowledge of ice king and fire fairy monsters. Given a player with a history of initially casting super-effective spells on the fire fairy but later mostly casting effective or ineffective spells on fire fairies, the system will generate predictions that over time player performance against fire fairies will decay. If the player also has a history of casting super-effective spells against the ice king the player model will predict the player to remain at a relatively high level of performance, assuming that most other players show a similar pattern of learning and mastering spell effectiveness. From this information the system would generate a combat sequence starting with a battle against an ice king and with a battle late in the sequence against a fire fairy. Once a player plays through one of the generated missions the system will subsequently update the player predictions to reflect the new model, before generating a new set of missions based on the updated player performance predictions.

6. TOWARD GENERATION OF FULLY REALIZED GAMES

Our future work involves evaluating the mission generation system. First, we intend to evaluate our tensor factorization player model to assess how accurately it can predict human player performance over time, and how many trials are required to train the model on any particular individual. Second, it is necessary to evaluate the extent to which our particular data-driven player modeling approach can affect noticeable change in a "closed-loop" system where missions are generated and played. As Cook, Colton, and Gow [3] note there can be differences between evaluation function ratings and human player ratings. As the primary function of performance evaluations is to tailor content to author requirements for player actions we avoid issues of player subjectivity by focusing on author goals. Third, we plan to examine the relationship between perceived difficulty, preference, and measured performance. Our study incorporates collecting player subjective ratings of perceived battle difficulty and enjoyment in addition to performance metrics to determine the extent to which our performance metrics provide useful correlations to player experience.

Currently, our game is limited to a fixed virtual environment. Future work with our system will explore the generation of spatial game content. Spatial layout of role-playing games has been previously explored [8, 18]. Hartsook et al. [8] especially advocate a pipeline approach where a mission is generated first, and then a space is constructed to support that mission. We believe it may be more beneficial to take spatial context into account when generating missions, as the time to navigate a large virtual world will affect game pacing and the way skill events align with a performance curve. We anticipate modifications to both our modeling and generation techniques due to the additional complexity added by spatial information. Modeling methods will need ways to appropriately abstract spatial location, potentially applying kernel learning techniques found successful for complex spatial modeling in the machine learning literature [17].

Additional advances to our mission generation approach may involve altering the dynamics of the game itself—for example, simplifying or complexifying the spell system with additional spell types or levels of spell effectiveness. These extensions could potentially alleviate boredom when players "master" a given game system or simplify a system that proves too complex for novice players. We anticipate building on previous work looking at evolving game systems while extending these efforts to alter complexity based on player performance (cf., [2, 3]). These efforts will also require ways to adapt NPC behavior appropriate to altered game rule sets.

7. CONCLUSIONS

We describe five criteria for data-driven player models that serve generation and demonstrate techniques to create such models and incorporate them into the process of generating game missions that combine combat and story events. Player models should predict player performance accurately, particularly when faced with large amounts of data that may be only partially filled for a given player. Temporal forecasting in player models is also a central requirement for predictions to account for the ways player performance or preferences fluctuate over time and to enable generation of sequential content. We employ tensor factorization to meet these diverse requirements in a domain-independent fashion.

Generating missions should involve requirements on sequential performance with designer preferences and knowledge about appropriate combinations of story content. A genetic algorithm enables a variety of competing factors to be incorporated into a single fitness function, meeting requirements from combat-related and story-related content. Temporal performance models combined with known combat content allows our mission generator to tailor sequences of battles to steer player performance toward an author-specified performance arc.

Procedural content generation techniques coupled with data-driven player modeling techniques hold great promise for automatically tailoring content to players' skill levels and preferences. We believe data-driven player modeling techniques meeting the five criteria we outline will help realize this potential and can expand efforts toward domain-independent PCG and adaptation solutions.

8. ACKNOWLEDGMENTS

The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

9. REFERENCES

[1] S. C. Bakkes, P. H. Spronck, and G. van Lankveld. Player Behavioural Modelling for Video Games. *Entertainment Computing*, pages 1–9, Jan. 2012.

[2] C. Browne and F. Maire. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2:1–16, 2010.

[3] M. Cook, S. Colton, and J. Gow. Initial results from co-operative co-evolution for automated platformer design. In *Proceedings of EvoGames 2012*, 2012.

[4] M. C. Desmarais and R. S. J. D. Baker. A review of recent advances in learner and skill modeling in intelligent learning environments. In *User Modeling and User-Adapted Interaction*, Oct. 2011.

[5] J. Dormans. Adventures in level design: generating missions and spaces for action adventure games. In *Proceedings of the FDG 2010 Workshop on Procedural Content Generation in Games*, 2010.

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

[7] B. Harrison and D. L. Roberts. Using Sequential Observations to Model and Predict Player Behavior. In *Proceedings of the 2011 Foundations of Digital Games Conference*, 2011.

[8] K. Hartsook, A. Zook, S. Das, and M. Riedl. Toward supporting storytellers with procedurally generated game worlds. In *Proceedings of the 2011 IEEE Conference on Computational Intelligence in Games*, 2011.

[9] Y. Koren and R. Bell. Advances in Collaborative Filtering. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 145–186. Springer US, Boston, MA, 2011.

[10] B. Li and M. O. Riedl. Planning for Individualized Experiences with Quest-Centric Game Adaptation. In *ICAPS 2010 Workshop on Planning in Games*, 2010.

[11] B. Magerko. Bringing the schoolhouse inside the box-a tool for engaging, individualized training. Technical report, DTIC Document, 2006.

[12] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[13] B. Medler. Using recommendation systems to adapt gameplay. *International Journal of Gaming and Computer-Mediated Simulations*, 1(3):2009, 2009.

[14] C. Pedersen, J. Togelius, and G. N. Yannakakis. Modeling player experience in Super Mario Bros. *2009 IEEE Symposium on Computational Intelligence and Games*, pages 132–139, Sept. 2009.

[15] J. Porteous, J. Teutenberg, D. Pizzi, and M. Cavazza. Visual Programming of Plan Dynamics using Constraints and Landmarks. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling*, pages 186–193, 2011.

[16] D. L. Roberts, M. Nelson, C. Isbell, M. Mateas, and M. Littman. Targeting specific distributions of trajectories in MDPs. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.

[17] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[18] N. Sorenson, P. Pasquier, and S. DiPaola. A Generic Approach to Challenge Modeling for the Procedural Creation of Video Game Levels. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(3):229–244, 2011.

[19] N. Thai-Nghe, T. Horvath, and L. Schmidt-Thieme. In *Proceedings of the 4th International Conference on Educational Data Mining (EDM 2011)*, 2011.

[20] D. Thue, V. Bulitko, M. Spetch, and E. Wasylishen. Interactive Storytelling: A Player Modelling Approach. In *Proceedings of the 3rd Conference on Artificial Intelligence and Interactive Digital Entertainment*, number July, 2007.

[21] J. Togelius, G. Yannakakis, K. Stanley, and C. Browne. Search-based Procedural Content Generation: A Taxonomy and Survey. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(99):1–1, 2011.

[22] B. G. Weber, M. Mateas, and A. H. Jhala. Using Data Mining to Model Player Experience. In *EPEX 2011*, 2011.

[23] G. N. Yannakakis and J. Hallam. Real-time game adaptation for optimizing player satisfaction. *Computational Intelligence and AI in Games, IEEE Transactions on*, 1(2):121–133, 2009.

[24] G. N. Yannakakis and J. Togelius. Experience-Driven Procedural Content Generation. *IEEE Transactions on Affective Computing*, pages 1–16, Mar. 2011.

[25] H. Yu and M. O. Riedl. A Sequential Recommendation Approach for Interactive Personalized Story Generation. *Proceedings of the 11th International Conference on Autonomous Agents and Multi Agent Systems*, 2012.

[26] A. Zook, M. O. Riedl, H. K. Holden, R. A. Sottilare, and K. W. Brawner. Automated scenario generation: Toward tailored and optimized military training in virtual environments. In *Proceedings of the 7th International Conference on the Foundations of Digital Games*, 2012.