Temporal Game Challenge Tailoring

Alexander Zook, and Mark O. Riedl

***Abstract*—Digital games often center on a series of challenges designed to vary in difficulty over the course of the game. Designers, however, lack ways to ensure challenges are suitably tailored to the abilities of each game player, often resulting in player boredom or frustration. Challenge tailoring refers to the general problem of matching designer-intended challenges to player abilities. We present an approach to predict temporal player performance and select appropriate content to solve the challenge tailoring problem. Our temporal collaborative filtering approach—tensor factorization—captures similarities among players and the challenges they face to predict player performance on unseen, future challenges. Tensor factorization accounts for varying player abilities over time and is a generic approach capable of modeling many kinds of players. We use constraint solving to optimize content selection to match player skills to a designer-specified level of performance and present a model—performance curves—for designers to specify desired, temporally changing player behavior. We evaluate our approach in a role-playing game through two empirical studies of humans and one study using simulated agents. Our studies show tensor factorization scales in multiple game-relevant data dimensions, can be used for modestly effective game adaptation, and can predict divergent player learning trends.**

***Index Terms*—artificial intelligence, machine learning, games, procedural content generation, game adaptation, recommender systems, player modeling, constraint programming**

I. Introduction

DIGITAL games often set players against a series of challenges in the form of puzzles, combat with computer controlled enemies, strategic planning and execution, or reflex-based timed execution. Designers plan challenge difficulty to vary over the course of a game and choose content to deliver appropriate challenges. Players, however, have different skills and backgrounds and mismatches between player skills and challenges lead to player boredom or frustration. Automatically tailoring game content to a player's abilities can avoid these mismatches. Tailoring can also enable dynamic tutorials that train players only on the skills they lack or enable online selection of procedurally generated content appropriate to a player's abilities.

Challenge tailoring is the problem of selecting skill-based content to match a player's abilities. Tailoring applies to a variety of skill-based challenges: selecting the health, movement speed, and aggressiveness of enemies in action role-playing games (RPGs) such as *The Legend of Zelda*; selecting the types and number of obstacles in platformer games such as *Mario Bros.*; selecting the accuracy and aggressiveness of enemies in a first-person shooter such as *Halo*; or selecting the composition of waves of enemies in an arcade shoot-'em-up like *Space Invaders*. Challenge tailoring is similar to *dynamic difficulty adjustment*, which makes online, real-time changes to a game. Unlike dynamic difficulty adjustment, challenge tailoring allows offline game content optimization and applies to classes of challenges beyond difficulty. Challenge tailoring has been used for platformer game levels [1], action RPG levels [2], first-person shooter item drop rates [3], and puzzle game pieces [4].

In this paper we describe and evaluate GAMETAILOR—a temporal challenge tailoring system. Temporal challenge tailoring extends challenge tailoring to model expected player skills over time to account for player learning or forgetting. We describe a turn-based RPG game where GAMETAILOR models player ability to choose the right action to take against particular types of opponents. GAMETAILOR uses a temporal player model to select future game opponents that guide players toward a designer-specified trajectory of performance over time. We describe techniques for temporal player modeling and planning content for predicted player states. Specifically we use tensor factorization to model players and use constrained optimization using Answer Set Programming to plan content. Tensor factorization, a form of temporal collaborative filtering, models and predicts player performance against potential sets of opponents. Constrained optimization then selects opponents (content) to minimize the difference between predicted player performance and desired player performance for each point in time. Desired player performed is provided through a designer-authored performance curve.

Challenge tailoring adapts to new player behaviors and guides expected player behavior toward designer goals. Temporal player models allow proactive content selection to avoid future boredom, account for learning, and arrange content to create a trajectory of events—e.g. having players experience a challenge they later triumph over. Performance curves extend mixed-initiative design techniques to author desired player *behaviors* rather than authoring desired *content*. We make three contributions toward temporal challenge tailoring:

1) Using tensor factorization for temporal player models
2) Performing full-loop temporal player modeling and challenge tailoring in two human studies and one simulation
3) Developing performance curves for mixed-initiative design of temporal player behaviors

We describe related work on game adaptation and GAMETAILOR's temporal collaborative filtering technique—tensor factorization—and constrained optimization technique—Answer Set Programming. After reviewing our previous empirical human study of tensor factorization for predicting player behavior [5] we present a follow-up empirical human study of GAMETAILOR's full-loop adaptation process. We discuss a simulation to verify GAMETAILOR's ground truth efficacy. We close with a discussion of the strengths and limitations of temporal player modeling and future work to develop the techniques.

II. Related Work

Digital games are often designed with a planned progression of content difficulty or skill mastery [6]. Players, however, come from diverse backgrounds, making different content appropriate for each player to achieve the desired progression. Game adaptation addresses this problem by automatically selecting game content appropriate to player abilities or preferences. Game adaptation techniques: (1) model game players and then (2) select content to give players a designer's desired experience (or at least behavior).

Game adaptation techniques fall into two classes: providing designers with fine-grained control over content adjustments or high-level control of objectives for player subjective responses. Fine-grained control systems use feedback loop models or rule-based systems to specify how content should change based on player state. Player state models have used player inventory and combat effectiveness in a first-person shooter [3], player health in a RPG [7], or player health and combat effectiveness in a RPG [8]. Other systems have modeled players as vectors of training skills [9], character traits [10] or player types [11]. These player models enable reactive content selection based on current player and/or game state. Reactive responses have been encoded as feedback loops [3], [7], multi-agent systems [8], or rule-based systems [9]–[11]. System designers, however, must ensure the reactive responses move players toward the desired state.

High-level control systems have used evolutionary computing and machine learning player models and selected content by optimizing model parameters for designer-given goals. These methods bypass the need to explicitly author rules for how to update player models by learning to predict how content influences player responses. Evolutionary computing player models have used multi-layer perceptrons with gameplay features based on player movement and combat in a platformer [1], [12], physical activity in a playground game [13], weapon choice in a space shooting game [14], and a variety of psychophsiological measures [15]. Machine learning player models have used clustering and support-vector machines with combat effectiveness features in an action RPG [2], reinforcement learning with puzzle game state and choice alternatives [4], and Gaussian Processes with combat effectiveness features in a space shooter game [16]. Machine learning and evolutionary computing models predict a desired player metric and content is then selected to optimize that player metric. Evolutionary computing and machine learning have been used to optimize for player preference [14], subjective responses of boredom, frustration, or fun [1], [17], retention [4], or combat effectiveness [16].

Game adaptation systems have overlooked two problems: (1) acquiring knowledge of how to adapt content or how players behave and (2) adapting content to create a time-varying trajectory of player experiences. Medler [18] was the earliest to suggest recommender systems for game adaptation to address the first problem. Recommender systems, specifically collaborative filtering models, share content across users to avoid needing detailed knowledge of individual users. Sharing information among users while still personalizing to individual users offers greater robustness to missing data. Tracking user behavior over time allows recommender systems to capture patterns in how certain states lead to future states and can address the second problem. GameTailor uses a collaborative filtering technique—tensor factorization—to model temporally varying player state.

Educational data mining researchers have shown the power of recommender systems to model learner skills [19]. Desmarais and Naceur [20] use matrix factorization—a recommender system algorithm—to automatically choose questions based on learner skills; the technique outperformed human experts in constructing these models. Thai-Nghe, Horvath, and Schmidt-Thieme [21] used tensor factorization (a generalization of matrix factorization) to successfully model temporal changes in learner skills.

Recommender systems have also shown promise for game adaptation. Thurau, Kersting, and Bauckhage [22] used a sophisticated matrix factorization variant to mine game data patterns. Min et al. [23] found matrix factorization outperformed related techniques when personalizing story-centric narrative sequences in a learning environment. Yu and Riedl [24] describe a matrix factorization variant that compiles a history of preference ratings to predict future preference ratings, demonstrating the matrix factorization variant in a choose-your-own-adventure game. GameTailor uses tensor factorization for a temporal player model and selects content to achieve a desired sequence of player behaviors using constrained optimization. To our knowledge, GameTailor is the first system to combine a temporal player model—using time as an explicit dimension—with game content adaptation to achieve a trajectory of player states. Temporal adaptation allows game authors to specify trajectories of desired behavior, rather than a single player behavior.

III. Game Tailor

GameTailor adapts game content to guide players toward designer specifications for intended player behavior over time. GameTailor uses a closed loop process of: (1) collecting player data when engaged with content, (2) modeling the player, (3) predicting player responses to new content, and (4) selecting content so that predicted player responses meet design goals for player responses. We study this approach in a turn-based RPG domain, modeling player performance as skills in battling opponents.

GameTailor combines techniques from collaborative filtering to model players and constrained optimization to select content. Collaborative filtering is used to predict player performance and has the advantage of sharing information across players. Information sharing can bypass limitations of sparse player data by using information from similar players; e.g. new players or games where players are unlikely to experience all content. Constrained optimization—we use Answer Set Programming (ASP)—is used to select future opponents for a player subject to design constraints. GameTailor allows designers to specify time-varying constraints on desired player behavior using a *performance curve*. Performance curves describe designer intent for expected player behavior over time.

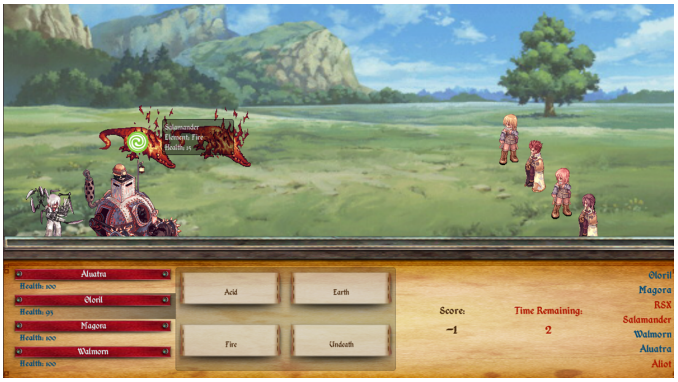

Fig. 1. A battle between monsters and the player team.

GAMETAILOR uses performance curves to bridge between designer goals for player behavior and specific content used to induce that behavior. Below we first present our game domain and performance curves, next explain our tensor factorization player modeling approach, and finally describe our ASP model to select game content according to design constraints.

A. Roleplaying Game Domain

To test GAMETAILOR we implemented a turn-based RPG and tailored the challenge of a sequence of RPG battles. The player leads a group of four characters through a sequence of spell-casting battles against groups of enemy monsters. See Figure 1 for the game's battle interface. Turns are limited to 10 seconds to require learning the spell system to the level of intuitive actions.

Players control four characters through a sequence of eleven battles against groups of four enemies in each battle. Each enemy is associated with one of eight spell types and player-controlled characters can attack with four of the eight possible spells. By forcing players to use different spell subsets we drove them to learn how to use all spell types, rather than specializing to a subset of spells. Casting a particular spell against an enemy of a particular type results in an attack being effective, ineffective, or super-effective, resulting in normal damage, no damage, or double damage against an enemy (Table I enumerates the combinations we implemented).

We intentionally created a spell system that was difficult to completely memorize. The spell system contains intuitive combinations—water spells are super-effective against fire enemies—and unintuitive combinations—undeath spells are super-effective against force enemies—ensuring that skill mastery could only be achieved by playing the game. Note that pairs of spells—e.g. fire and force—are repeated in Table I. This means there is a simpler underlying skill structure for players to learn; there are effectively only four spells. Player characters each have four different spells that provide a super-effective spell at all times.

Our scoring system was based on spell effectiveness to motivate players to learn; effective spells earn two points, ineffective spells earn zero points, and super-effective spells earn five points. Enemy attacks decrease player score by one. Player characters were assigned different spell sets, forcing players to learn the spell system. Together, these systems were designed to motivate players to learn the game systems while preventing players from succeeding by only using knowledge of genre conventions.

TABLE I
SPELL EFFECTIVENESS MATRIX

Attack ↓ Def. →	fire	water	acid	ice	light.	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

key: super-effective = 2, effective = 1, ineffective = 0

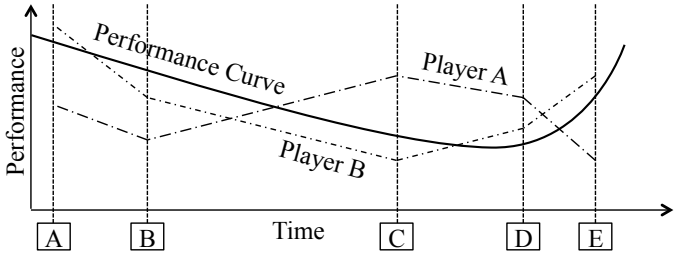

Fig. 2. Example performance curve. The solid line indicates designer-requested performance levels and the dotted lines show two player trajectories through the game.

B. Performance Curve

A performance curve describes a designer's intent for player performance over a series of events with measurable outcomes. In general, GAMETAILOR uses a designer-defined scoring method that quantifies player performance on an event. We used RPG battles as events and measured performance as the average effectiveness of all spells a player used against a particular type of enemy in a battle. Figure 2 shows an example performance curve indicating a desired gradual decrease in player performance before rising at the end of the game. Ideally players will experience this gradual decrease in performance as increasing difficulty in the game, even as the player masters earlier challenges.

We developed performance curves to enable designers to indicate plans for temporal patterns in player behavior over time. Most previous mixed-initiative game design tools have focused on enabling designers to explicitly control game *content*—e.g. platformer levels [25], world terrain [26], or game maps [27]. Performance curves extend these efforts by enabling designers to express desired player *behavior* over time, with the system responsible for content-level decisions. Complementing Smith et al. [28], [29] work to specify constraints on the space of all possible player behaviors, we specify constraints on expected human player behavior.

In adaptive games performance curves act as a proxy for a designer's intent for what a system should realize in the face of individual player differences. For example, a curve with a constant value (horizontal line) indicates designer intent to have player performance remain constant over time. An adaptation system is responsible for adjusting content to ensure

a player experiences a constant level of difficulty, even though the player may learn to play the game better. A different curve could specify a series of gradual rises and short drops to create a sense of mounting difficulty followed by brief breaks.

Performance curves apply to many measurable player behaviors beyond in-game performance. For example, player preferences for game events could be modeled using player choices when given alternative events to experience. Further, multi-dimensional performance curves are possible—e.g. adapting games for both player performance and preference. Our content adaptation approach (Section V) allows for multiple sources of design constraints and readily integrates multi-dimensional optimization criteria.

IV. Tensor Factorization

This section presents tensor factorization as a player modeling approach. We use tensor factorization to model player performance over time in games. We chose tensor factorization due to its favorable scaling properties, ability to cope with missing data, high accuracy, speed in generating predictions, and previous success in other applications [30]. While our experiments study a turn-based RPG, we believe our techniques generalize to other games that use skill-based events or measurable subjective player feedback.

Tensor factorization decomposes multidimensional measurements into sets of latent components that capture underlying features of the high-dimensional data. Tensors generalize matrices to higher dimensions: matrices have a two-dimensional structure; tensors have a three or more dimensional structure. Intuitively, the decomposition process searches for a set of low-dimensional vectors that, when combined, accurately recreate the values in the original tensor. The low-dimensional approximation drives the model to identify the main underlying dimensions of variability in the data. We expect player performance to depend on a few underlying aspects of skill, encoded by this approximation. Low-dimensional approximations have the additional virtue of constraining the model from over-fitting the training data to reduce low-accuracy predictions on new cases. We extend two-dimensional matrices of player performance on skill-based events with a third dimension for the time of the event.

Tensor factorization is an extension of matrix factorization, which has been used in collaborative filtering applications—such as the Netflix Prize data mining competition—to great success [31], [32]. We first discuss matrix factorization and next show how tensor factorization extends these principles. Matrix factorization uses information from a group of users that has experienced a set of content to predict what new users that have only been partially exposed to that content will do. Matrix factorization learns to predict new player behaviors as they appear, rather than requiring knowledge encoding these new cases and how to predict them. Matrix and tensor factorization handle missing information from players, achieve high accuracy, and can be trained efficiently even with large amounts of data [22], [31], [32].

In the standard collaborative filtering setting, user data is represented in a matrix $M = U \times I$ containing structured feedback such as user (U) preference ratings on items (I) or user (U) performance on test questions (I). Decomposition extracts latent factors relating to users and items. Latent factors describe underlying traits of the data: e.g. personality traits extracted from questionnaires or movie genre preferences from Netflix movie ratings. Prediction combines these factors to estimate user ratings on previously unrated items.

Tensor factorization extends matrix factorization as follows. As a running example we will use our application turn-based RPG game, predicting player performance against each type of enemy on each battle. Formally, we represent player data in a three-dimensional tensor $Z = U \times I \times T$ where U is the player ("user"), I is the spell type ("item") and T is the time of the performance recording. We use Canonical polyadic (CP) decomposition (a generalization of the singular value decomposition method used for matrices) to decompose the three-dimensional Z tensor into a weighted combination of three latent vector factors:

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

where $\circ$ is the vector outer product, λ_k are positive weights on the factors, w_k are player factors, h_k are spell type factors, and q_k are time factors. K is the number of components used for each factor as an approximation of the true structure, keeping the set of the K most important components found [30]. The decomposition can be computed by minimizing the root mean squared error between the result of the outer product above and true data values, iteratively fitting each of the factors while fixing values of the other factors until convergence. We employ the N-way toolbox [33] to perform this decomposition.

The outcomes of unseen events are predicted by combining the latent factors specific to the player, task, and time. Computationally, prediction is the inner product of the three latent factors, a computationally efficient process:

$$\hat{p}_{uit} = \sum_{k=1}^{K} \lambda_k w_{uk} h_{ik} q_{tk}$$

where $\hat{p}_{uit}$ is the predicted performance of player u on task i at time t, w_{uk} indexes the target player's player factor, h_{ik} indexes the target spell type, and q_{tk} indexes the target time. Intuitively, prediction combines the weighted strengths of the underlying factors describing the player, task, and predictive time point to estimate that particular performance value.

V. Content Adaptation

Challenge tailoring in our RPG is the selection of opponents in a battle to achieve a desired level of player performance. The designer-specified performance curve provides desired levels of player performance. GameTailor minimizes the difference between the performance curve and the predicted performance of a particular player against a set of enemies across all battles. In our second study we tailored opponents in last four battles of the game; we used a performance curve specifying a steady drop in performance, attempting to create a feeling of increased difficulty. Four enemies were selected

for each battle to have a predicted average performance value matching the desired performance.

GAMETAILOR selected enemy sets using constrained optimization implemented in Answer Set Programming (ASP) [34]. Answer Set Programming is a declarative programming language used for finite domain constraint solving using logic programming semantics. As a declarative programming language ASP allows design criteria to be specified in terms of constraints on the form of the final solution. Fast satisfiability solvers have been implemented to enforce hard constraints and optimize for soft criteria. Answer sets consist of logical variable values (if any exist) meeting the requested criteria. ASP allows design requirements to be expressed declaratively and efficiently solved without needing to formulate game-specific algorithms for how to select that content.

To use ASP we defined a representation for enemies and the performance curve. Enemies were represented in terms of: (a) their spell type, (b) the battle (and slot of four possible options in a battle) in the sequence, and (c) the predicted player performance against the enemy in that battle in the sequence. Predicted player performance on a battle was measured as the average player performance over a selection of four (not necessarily distinct) enemy types according to tensor factorization predictions for that battle. The performance curve consisted of a series of battles and the level of performance to achieve. ASP was constrained to find sets of enemies matching predicted player performance to performance curve values. All answers within an answer set are equally effective at matching the performance curve; differences only relate to the specific enemies chosen. We used the first generated answers among the answer sets for simplicity and allowed for multiple enemies of the same type. ASP can add inter-battle or inter-enemy constraints for greater design control; in our study we only matched performance without other constraints for simplicity.

VI. STUDY 1: PREDICTING PLAYER PERFORMANCE

Our first empirical study examined the efficacy of tensor factorization (TF) for predicting player performance without adaptation. Many of these results have been previously reported [5]; we review the methodology and results to contextualize our follow-up work evaluating the adaptation approach. Study 1 tested four hypotheses related to predicting player performance:

- H_1 TF player performance predictions improve with more per-player data.
- H_2 TF player performance predictions improve with more players.
- H_3 TF predictions of future player performance improve with more per-player performance history.
- H_4 TF predicts player performance better than related baseline matrix factorization (MF) models.

We quantified prediction quality as cross-validated *percent variance explained*—a standard evaluation metric for matrix and tensor decomposition evaluations [30], [32].[1] Percent variance explained (PVE) measures the proportion of the total variability in the data that is explained by a model. Larger percent variance explained (up to 100%) indicates a model fits given data better.

The first three hypotheses test different forms of model scaling in the amount of data used. If models achieve a high PVE they effectively explain the data; if models scale well they are robust to different forms of missing data. H_1 tests TF's robustness to sparse or randomly missing data; H_2 tests TF's scaling as the number of game players increases; and H_3 tests TF's scaling to predict the future as more player history is gathered. We hypothesized TF would improve with more data in all three cases and found an overall high PVE (over 80%). Confirming these hypotheses shows TF can capture patterns in player performance in a turn-based RPG battle system and that TF scales in game-relevant data dimensions.

H_4 compares TF to MF models as a baseline. Our TF model organizes player data in a three-dimensional tensor (player, enemy spell type, battle number), where each value in the tensor is the average performance of the player when attacking opponents of a specific spell type during a specific battle. We hypothesized TF would outperform MF as TF models can make time an extra dimension, while MF models must integrate temporal information into other dimensions, losing this additional information. We compared TF to two baseline MF models: MF using an unfolded tensor ("unfolded"), and MF averaging over time ("time-averaged"). The matrix unfolding of our tensor concatenates battles column-wise [30]. For example, a tensor can record data from 10 players using any of 8 spell types across 5 battles using a three-dimensional $10 \times 8 \times 5$ tensor. Unfolding this tensor concatenates battles and spells per player into a single dimension, producing a 10×40 matrix. Time-averaging the tensor instead takes the mean performance value over every time a spell type was used. Time-averaging produces a 10×8 matrix, losing information about changes in performance over time. We hypothesized TF would outperform MF on an unfolded tensor as TF retains more structural information about time in the data. We hypothesized TF would outperform MF on a time-averaged tensor as TF retains information on temporal variations in player behavior.

Game designers often plan for a desired player *experience*, rather than behavior or performance. We hypothesized there is an inverse relationship between objective, measurable player performance and subjective, self-reported difficulty:

- H_5 Player performance ratings inversely correlate with self-reported difficulty ratings.

Should this hypothesis hold it will verify that in the context of our turn-based RPG we can use skill performance as a proxy for difficulty. We evaluated H_5 using Kendall's rank correlation test to test the null hypothesis that player performance has no correlation with self-reported ordinal difficulty ratings.

Our results from study 1 show TF has a high PVE with large amounts of missing data (H_1) or fewer players (H_2). Further, TF has a high PVE when fill on future player performance (H_3) with a relatively short part of player history (the first 60% of the game). TF is moderately more effective than baseline MF models (H_4). There was a medium inverse correlation between player difficulty ratings and in-game performance

[1] H_4 used 60-fold cross-validation, the remaining hypotheses used 10-fold cross-validation.

(H_5). Taken together, the results of study 1 suggest TF robustly predicts player performance and scales in relevant data dimensions.

A. Study Methodology

For study 1 we recruited 32 people to play our game. Players had five minutes to review a document explaining the spell system and the game interface. Once familiar with the game players completed a sequence of 11 battles while we recorded player performance as the effectiveness of spells chosen against enemies. After each battle we asked players to report how difficult and enjoyable the battle was on a 5-point Likert-like scale.

We recorded each spell players cast on each enemy on each turn and battle in the game along with the associated performance value: 0 for ineffective, 1 for effective, and 2 for super-effective. Player behavior traces varied in the number of turns taken in a battle because spell effectiveness determined damage to enemies. We averaged performance for each spell type across all turns within a given battle, leaving the performance value missing for any unused spells.

B. Results

As a control we compared player performance across demographic attributes (age, gender, race, prior video game and RPG experience). An ANOVA found only prior RPG experience was a significant factor in player performance ($p < 0.01$), all other factors were not significant ($p > 0.05$). We omitted prior RPG experience from our models to emulate a use case where our system starts without prior player information.

Players showed substantial performance variability measured as the standard deviation of per-battle performance. Performance was on a [0,2] scale with per-battle performance standard deviations ranging from 0.62 to 0.76; the average value was 0.71. Performance variability indicates there were sufficient differences in the data collected that the MF and TF models needed to fit the data.

To test TF's robustness to sparse data (H_1) we removed a randomly selected subset of the data. Randomly missing values simulates more sparse data collection. Figure 3 shows TF captures most of the variability in the data with as little as 60% of the total dataset. These results support H_1: TF achieves a high PVE even when using small portions of the total data and improves with more data. Using more components creates more complex low-dimensional approximations. In our case player behaviors were explained by a simple model using only two components; more components likely created models that overfit the data and thus had a lower PVE.

To test TF's scaling in the number of players (H_2) we removed a randomly selected subset of players. Altering the number of players in the dataset simulates a growing number of game players. We randomly subsampled 6, 11, 16, 21, or 27 players from our full set of 32 players. Accuracy improved on the 2 component model from 81% PVE with 6 players to 87% with 27 players, while the 4 component model improved from 53% to 86%, respectively. Other numbers of components showed similar trends, converging to approximately 86% PVE

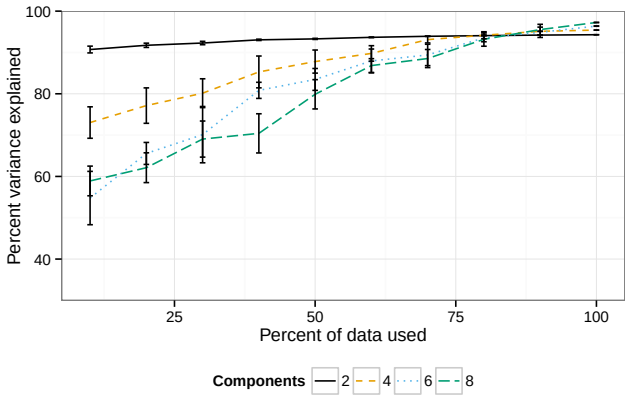

Fig. 3. TF PVE when using a random subset of data from study 1; bars indicate 95% confidence intervals.

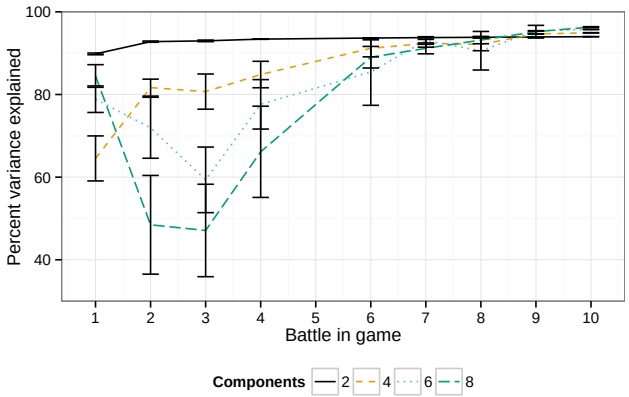

Fig. 4. TF PVE when using data for a player limited to the initial portion of the game using study 1 data; bars indicate 95% confidence intervals.

with 27 players. These results support H_2: TF reaches a high PVE even with few players and improves with data from more players.

To test TF's scaling in the length of player history available (H_3) we removed a subset of future battles from the data of a target group of players. Removing future data emulates predicting future player performance from the initial history of player actions. Figure 4 shows TF reaches a high PVE with 60% of a player's total gameplay trace (7 out of 11 battles). These results support H_3: TF achieves high PVE even when using only an initial subset of a player's history and improves with more player history.

To compare TF to baseline MF models (H_4) we trained each of the models on the full dataset. TF outperformed the unfolded and time-averaged MF models (Table II). Differences among the models were small overall, with TF performing best with few components. The unfolded model shows negative PVE with more components—it fails to explain player behavior when using more flexible models of player abilities. Conversely, the time-averaged model gradually improves with larger numbers of components, indicating the need for a more complex model to capture player underlying abilities. Overall,

TABLE II
COMPARISON OF PERCENT VARIANCE EXPLAINED BY DIFFERENT MODELS USING VARYING NUMBERS OF FACTORS. COMPARES TENSOR MODEL (TENSOR), MATRIX MODEL CREATED BY UNFOLDING THE TENSOR (UNFOLDED) AND MATRIX MODEL AVERAGING OVER ALL TIME POINTS (TIME-AVERAGED).

components	tensor	unfolded	time-averaged
2	92	90	-19
3	87	86	15
4	73	45	42
5	69	28	89
6	65	46	90
7	58	-91	91
8	70	-152	91

these results only weakly support H_4: TF models player behavior as well as related MF models with a relatively simple (in terms of number of components) model of player abilities.

To test the relationship between objective performance and self-reported difficulty (H_5) we tested the correlation between mean per-battle performance and ordinal difficulty ratings. As hypothesized, players reported difficulty had a significant, medium negative correlation (Kendall's rank correlation test, $p < 0.001$ and $\tau = -0.35$) with in-game performance. Enjoyment also had a significant, small ($p < 0.05$ and $\tau = -0.09$) negative correlation with performance. Thus, in our game lower objective performance correlates with greater perceived difficulty and slightly greater enjoyment.

VII. STUDY 2: ADAPTING GAME CONTENT

Our first study showed TF can predict player performance, but did not address whether these predictions allow for game adaptation. Study 2 tested adaptation using ASP to select opponents for the last four battles of the study 1 game. We hypothesized that adaptation could guide player performance:

H_6 Player performance on adapted game content will match TF player performance predictions.

Adaptation quality was quantified as the mean absolute error (MAE) between desired performance and observed player behavior on a piece of adapted content—here a battle. Our study results partially refuted H_6: adaptation never achieved non-zero error, though in half of the cases the error was small.

Based on the results of testing H_6 we conducted a post-hoc analysis of data from both studies. We believed that adaptation in study 2 introduced out-of-sample behavior—behaviors that were not observed in the original game used as training data—that led to poor prediction quality. Repeatedly exposing players to their weaknesses taught them to overcome these weaknesses. We hypothesized TF could predict this new behavior when using the new data showing this behavior:

H_7 TF has high quality predictions using player performance data from on adapted content.

TF prediction quality was quantified using PVE and achieved similar PVE to the results from study 1, supporting H_7. We also hypothesized TF's scaling properties and comparative effectiveness (tested as H_1 through H_4 in study 1) would hold in the context of the combined dataset from both studies. Using the combined dataset from both studies we found support for these hypotheses (H_1' through H_4'). Together these results suggest temporal adaptation may initially fail due to player learning, but can model new player behaviors once elicited.

A. Study Methodology

Study 2 used the same game as study 1 with 30 new players. Players completed the same initial seven battles from study 1. Upon completing battle 7 a new player's data was added to the database of the 32 players from study 1. TF predicted the new player's performance against each type of opponent for the remaining four battles. ASP then selected content (based on TF performance prediction values) to match a performance curve that described decreasing player performance over the last four battles (average performance across enemies of 1.5, 1.0, 0.5, 0.5 on a [0,2] scale).

B. Results

To test game adaptation (H_6) we compared desired performance from the performance curve and observed performance on the four adapted battles using the MAE as an error metric. Player performance was near desired performance for the first two battles before deviating over the last two battles. All battles showed significant (non-zero) mean absolute error (t-test with Bonferroni correction $p < 0.001$), with values of 0.24, 0.65, 1.21, and 1.09 over the last four battles, respectively. Non-zero, but sometimes small, errors refute H_6: GAMETAILOR's adaptation fails to perfectly drive player performance to desired levels, though the gap between desired and actual performance is sometimes small.

Qualitative feedback from players revealed the reason for these prediction errors—forcing players to fight many enemies testing the same skill rapidly trained players in that skill. Players reported the final battles being initially challenging before learning how to defeat opponents through practice. The study design of gradually decreasing performance led players to face multiple of a single type of enemy, giving players more opportunities to learn. These considerations motivated a post-hoc analysis of whether TF could learn to predict the new behavior (H_7).

To test whether TF could adapt to the new learning behavior (H_7) we trained TF on the full dataset comprising the players from both studies. We also retested our hypotheses for TF's scaling (H_1 through H_3) and effectiveness compared to baseline models (H_4) using this new dataset (labeling these hypotheses H_1' through H_4'). TF scaled in dataset size with the combined dataset. TF fit player behavior (80% PVE) when hiding 30% of data randomly (H_1'), and scaled smoothly with increasing amounts of information on players (up to 96% PVE) (Figure 5). Even with adaptation introducing out-of-sample behavior, TF effectively fit data from the full 62 player dataset (H_7) with a slightly better fit than when using only 32 players (H_2': from 92% PVE to 96% PVE). TF achieved a good fit (over 90% PVE) for predicting future player behavior (H_3') with only 50-60% of the game completed (Figure 6). As in the first experiment, models with few components had the best performance while models with more components achieved similar quality with more data. Together these results show

TABLE III
COMPARISON OF PERCENT VARIANCE EXPLAINED BY DIFFERENT MODELS USING VARYING NUMBERS OF FACTORS.

components	tensor	unfolded	time-averaged
2	93	85	92
3	92	82	93
4	91	79	92
5	91	83	92
6	90	77	92
7	88	72	92
8	88	70	92

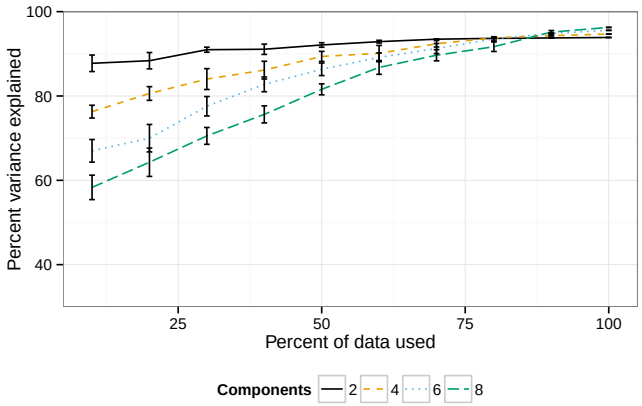

Fig. 5. TF PVE when using a random subset of data from both studies; bars indicate 95% confidence intervals.

TF can learn from anomalous behavior during adaptation to improve as more player behavior is observed.

TF and the time-averaged model show comparable high performance (H_4'), while the unfolded model shows comparatively poor performance (Table III). As in study 1, TF degrades with more components, likely due to overfitting the data. The time-averaged model shows similar performance across the varying number of components, indicating a stable model that gains little from additional flexibility but is less likely to overfit. The unfolded model did worst—likely caused by modeling spell types as independent across battles due to the concatenation used in unfolding.

We performed a post-hoc analysis to examine why the time-averaged model performed well. A linear regression of player performance against the battle they were on in the sequence found progression across battles has a significant, but small, impact on player performance ($\beta = 0.014$, $p < 0.001$). That is, for every 10 battles we could expect players to improve 0.1 points on a [0,2] scale. Performance in our game was highly variable: per-battle standard deviations in the full data set ranged from 0.64 to 0.79 with an average value of 0.72. Per-player *changes* in performance, however, were small: poor players remained poor while skilled players remained skilled. With little player learning occurring, TF had limited advantages over the time-averaged model that ignores this additional information and estimates player performance based on their average performance.

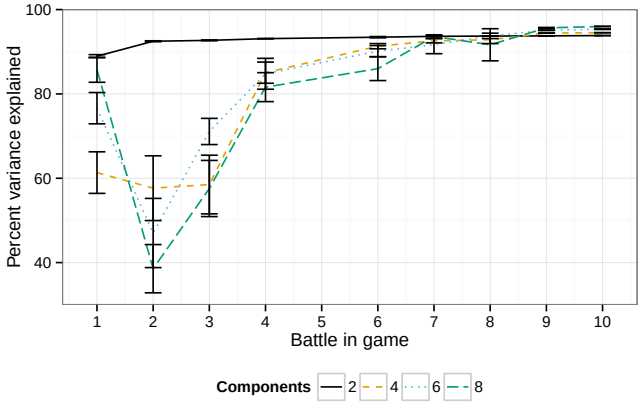

Fig. 6. TF PVE when using data for a player limited to the initial portion of the game using data from both studies; bars indicate 95% confidence intervals.

VIII. SIMULATION STUDY

As players showed little temporal variation in our two empirical studies we performed a simulation study with hard-coded learning trends. Our simulations generated data with differing rates of learning across the eight spell types in our game. We again compared TF to baseline models (H_4'') and found TF fits temporally varying data as well as the matrix alternatives. We also retested and found support for all scaling properties (H_1'' through H_3''). Together these simulation results support the conclusion that TF can model player behavior with learning trends even with variations across players and across individual player abilities.

A. Study Methodology

Our simulation study generated data for 30 players with varying learning rates on each of the 8 spell types across 11 battles. For each player we synthesized a series of performance values when facing a given spell type by: uniformly randomly generating a starting and ending performance level in the [0,2] range, sorting these to ensure increasing performance over time, and interpolating performance to linearly increase from the initial to the final value. We added a moderate amount of Gaussian noise (mean 0, standard deviation 0.5) to all performance values and clamped values to fall within [0,2] after adding noise. The 8 spell performance levels were generated independently to create the 8×11 matrix of an individual player's performance. We repeated the synthesis process for 30 total players.

B. Results

We repeated our baseline comparison and scaling tests from the previous two studies using only data from the simulated players (H_1'' through H_4''). TF fit the simulation data as well as the unfolded or time-averaged models (Table IV). Performance for all models was best with eight components—this is expected as the underlying generation process created eight independent learning trends for the models to fit. TF's superior performance supports the claim that tensors can

TABLE IV
COMPARISON OF TF TO BASELINE MF MODELS ON SIMULATED LEARNING DATA.

components	tensor	unfolded	time-averaged
2	87	86	84
3	89	87	87
4	91	90	89
5	94	92	91
6	96	95	92
7	97	96	93
8	98	97	94

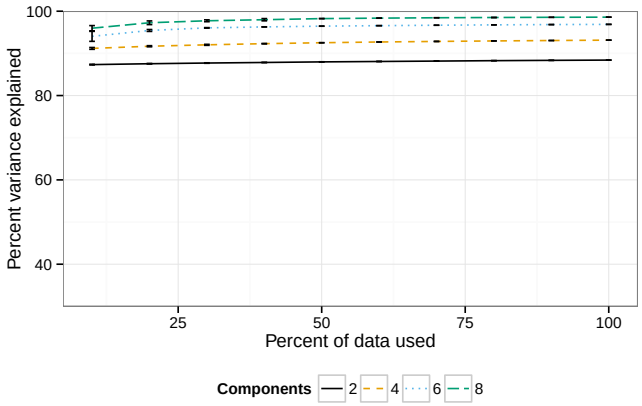

Fig. 7. TF PVE when using a random subset of data from simulations; bars indicate 95% confidence intervals.

model temporal trends in player behavior that related baseline models less effectively predict (H_4'').

As in the previous studies TF showed favorable scaling in randomly missing data (H_1'', Figure 7) and missing future data (H_3'', Figure 8). Increasing the number of players increased the PVE of the model (H_2''). The 2 component model increased from 81% to 88% PVE when increasing from 5 to 25 players; the 8 component model increased from 87% to 92% over the same range. Unlike the empirical studies, TF performance improved with more components up to the 8 used. This aligns with our simulation data generation process that used 8 independent learning trends. Thus, even with learning trends TF retains its favorable scaling properties. Together these results support the claim that TF can be used for temporal player modeling.

IX. DISCUSSION AND FUTURE WORK

GAMETAILOR addresses the temporal challenge tailoring problem—selecting content to guide players to designer-specified behavior while accounting for how players change over time. A performance curve allows high-level authoring of desired gameplay behavior trajectories. Tensor factorization models temporal player behavior and constrained optimization can select content to guide player behavior toward designer intent using the performance curve. Our experiments with human players show tensor factorization scales with the density of data collection (H_1), number of game players (H_2), and length of player gameplay history available (H_3) and can adapt to errors due to unexpected player behaviors (H_7). Tensor

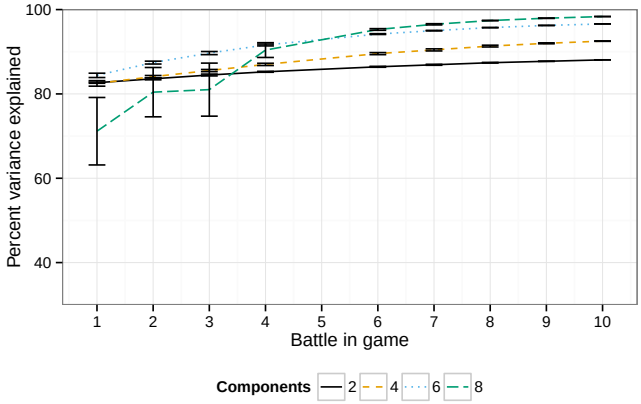

Fig. 8. TF PVE when using data for a player limited to the initial portion of the game using data from simulations; bars indicate 95% confidence intervals.

factorization has similar performance to baseline models when using data where players show variability but little learning (H_4 and H_4') and with simulated players that have hard-coded learning patterns (H_4''). GAMETAILOR has some capacity to guide player behavior through adaptation (H_6), though it fails to capture out-of-sample behavior induced by novel content combinations. Together, our results show tensor factorization has promise for temporal player modeling and can be used for (limited) temporal challenge tailoring.

We combine a temporal player model with game adaptation to model how players vary in ability over time as they experience game content. By anticipating temporal variation in player behaviors, we can proactively generate future content, avoiding the pitfalls of reactive content selection. Tensor factorization successfully modeled the small set of independent skills in our game with relatively little data. We hypothesize that games using tensor factorization or related collaborative filtering approaches to challenge tailoring will require only a small amount of training data from any given player. In game development contexts an initial scripted tutorial should be sufficient to elicit this data. We believe our techniques generalize to games with larger numbers of skills, though this will likely require more data than we used (30-60 players over 11 events).

Our techniques, however, require further development before they can be used in existing commercial games. Developers must choose the appropriate prediction model (e.g. tensor factorization or the time-averaged matrix) and number of model components to use. We found both tensor factorization and the time-averaged matrix factorization models performed well, making these preferable choices. Developers could make a small-scale comparison using randomly sampled player data from an existing game (or alpha/beta game testing) to evaluate these models and inform the choice of model to use for a game. Should developers change their desired performance curve they would likely need to retrain their prediction model on player behavior on new content settings before using the model's predictions for adaptation. Our study 2 results show tensor factorization can predict new behaviors but requires that

these new situations are first tested to gather examples of the new behaviors.

Our experiments also open several additional research questions. Does tensor factorization accurately predict human player learning? As players showed little learning in our experimental game additional work should modify the testbed game to help learning through more detailed feedback (e.g. suggesting the correct spell to use) or allowing a longer period of play. Our simulation studies show promise for predicting player behavior when learning and a follow-up study should test whether these results hold for humans. In addition, follow-up work should examine the relationship between adaptation and player experience to test whether adaptation improves player experience.

Several factors were important to success in our application domain of turn-based RPGs: action outcomes were easily attributed to individual skills, skills were largely independent, and players faced opponents with fixed behaviors. Games, however, often involve features that confound the clear expression of skill: outcomes may have an element of chance, actions may depend on multiple player skills, or actions may strongly depend on context. Noisy outcomes—e.g. due to game system randomization—require more data to discern the effect of player skill, but are amenable to the same factorization methods.

Skills often have inter-dependencies—e.g. the ability to effectively issue commands in a strategy game is required before a player can successfully execute more complex tactics or strategies. Skills may require mastering a full procedure with potential branching or looping over a sequence of actions [35], choosing a set of actions together to achieve a result [29], or combining strategic and tactical reasoning in a dynamic setting [36]. Extending tensor factorization to more complex skill dependencies may require additional techniques to model the relationships among the tensor components (e.g. using graphical models to introduce hierarchies [37]) or dimensions to account for skill dependencies. Future work will need to investigate when increasing the number of components is effective compared to adding new dimensions to the tensor (as we added time) or adding latent dimensions. Many important aspects of balancing among computational speed, data scaling, and system accuracy will need to be studied.

Developing appropriate constraints for adaptation in a closed loop—where a system observes player behavior, predicts subsequent behavior, and modifies the game for the player—is also important to ensure adapted content encourages desired player behavior. In our second empirical study we observed that adaptation selected multiple enemies of a single type, ultimately over-training players against this particular enemy. Constraining the number of any given type of enemy in a battle would address this problem and illustrates the value of more sophisticated adaptation constraints. In general, we see developing these systems and cross-domain approaches to solving challenge tailoring as pressing open problems with broad applicability to entertainment, education, and training.

Challenge tailoring is only one facet of controlling game content that ignores the non-skill-based elements of a game. Games often use fictional context between challenges to motivate the challenges presented and immerse players in a game world. *Challenge contextualization* is the problem of providing non-challenge content—e.g. cutscenes, dialog with characters, quests, or plot exposition—that motivate challenges. Contextualization answers the question "why am I, the player, fighting these enemies?" Techniques for story generation, quest generation, or drama management can potentially provide this content to increase the scope of a game adaptation system [38]. Tailoring and contextualizing challenges extends the ability of designers to shape game experiences.

GAMETAILOR's combination of temporal player modeling and optimization allows game designers to abstractly specify the "feel" of their game with respect to difficulty and allows a system to fill in the content details that best suit an individual player. Challenge tailoring makes it possible for a single game to cater to broader demographics of potential game players with widely varying skills and backgrounds. Tailoring games ultimately enables the same game to reach a broader audience, allowing games for entertainment, training, or education to reach ever-diversifying player demographics with minimal additional design work.

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. Special thanks to Michael Drinkwater for assistance with development of the GAMETAILOR game.

REFERENCES

[1] 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.

[2] 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.

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

[4] B. Harrison and D. L. Roberts, "Analytics-driven dynamic game adaption for player retention in Scrabble," in *IEEE Conference on Computational Intelligence in Games*. IEEE, 2013.

[5] 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.

[6] G. Elias, R. Garfield, and K. Gutschera, *Characteristics of Games*. MIT Press, 2012.

[7] G. van Lankveld, P. Spronck, and M. Rauterberg, "Difficulty scaling through incongruity," in *4th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2008.

[8] L. Pons, C. Bernon, and P. Glize, "Scenario control for (serious) games using self-organizing multi-agent systems," in *International Conference on Complex Systems*, 2012, pp. 1–6.

[9] 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.

[10] M. Seif El-Nasr, "Interaction, narrative, and drama: Creating an adaptive interactive narrative using performance arts theories," *Interaction Studies*, vol. 8, no. 2, pp. 209–240, 2007.

[11] 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.

[12] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience in Super Mario Bros," in *IEEE Symposium on Computational Intelligence and Games*, 2009.
[13] G. Yannakakis, H. Lund, and J. Hallam, "Modeling children's entertainment in the playware playground," in *2nd IEEE Symposium on Computational Intelligenceand Games*, 2006.
[14] 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*, vol. 1, pp. 245–263, 2009.
[15] G. Yannakakis and J. Togelius, "Experience-driven procedural content generation," *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011.
[16] A. Zook, E. Fruchter, and M. O. Riedl, "Automatic playtesting for game parameter tuning via active learning," in *9th International Conference on the Foundations of Digital Games*, 2014.
[17] L. Yu and J. V. Nickerson, "Cooks or cobblers?: crowd creativity through combination," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2011, pp. 1393–1402.
[18] B. Medler, "Using recommendation systems to adapt gameplay," *International Journal of Gaming and Computer-Mediated Simulations*, vol. 1, no. 3, pp. 68–80, 2009.
[19] M. C. Desmarais and R. S. J. D. Baker, "A review of recent advances in learner and skill modeling in intelligent learning environments," *User Modeling and User-Adapted Interaction*, vol. 22, no. 1-2, pp. 9–38, 2012.
[20] M. C. Desmarais and R. Naceur, "A matrix factorization method for mapping items to skills and for enhancing expert-based q-matrices." in *Artificial Intelligence in Education*, 2013.
[21] N. Thai-Nghe, T. Horvath, and L. Schmidt-Thieme, "Factorization models for forecasting student performance," in *4th International Conference on Educational Data Mining*, 2011.
[22] C. Thurau, K. Kersting, and C. Bauckhage, "Convex non-negative matrix factorization in the wild," in *9th IEEE Conference on Data Mining*, 2009.
[23] W. Min, J. P. Rowe, B. W. Mott, and J. C. Lester, "Personalizing embedded assesment sequences in narrative-centered learning environments: A collaborative filtering app," in *Artificial Intelligence in Education*, 2013.
[24] H. Yu and M. O. Riedl, "A sequential recommendation approach for interactive personalized story generation," in *11th International Conference on Autonomous Agents and Multiagent Systems*, 2012, pp. 71–78.
[25] G. Smith, J. Whitehead, and M. Mateas, "Tanagra: Reactive planning and constraint solving for mixed-initiative level design," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 201–215, 2011.
[26] R. M. Smelik, T. Tutenel, K. J. de Kraker, and R. Bidarra, "A declarative approach to procedural modeling of virtual worlds," *Computers & Graphics*, vol. 35, no. 2, pp. 352–363, 2011.
[27] A. Liapis, G. N. Yannakakis, and J. Togelius, "Sentient sketchbook: Computer-aided game level authoring," in *8th International Conference on the Foundations of Digital Games*, 2013.
[28] A. Smith and M. Mateas, "Answer set programming for procedural content generation: A design space approach," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 187–200, 2011.
[29] A. M. Smith, E. Butler, and Z. Popović, "Quantifying over play: Constraining undesirable solutions in puzzle design," in *8th International Conference on the Foundations of Digital Games*, 2013.
[30] T. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.
[31] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Boston, MA: Springer, 2011, pp. 145–186.
[32] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, pp. 1–19, 2009.
[33] C. Andersson and R. Bro, "The N-way toolbox for MATLAB," *Chemometrics and Intelligent Laboratory Systems*, vol. 52, no. 1, pp. 1–4, 2000.
[34] C. Baral, *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.
[35] E. Andersen, S. Gulwani, and Z. Popović, "A trace-based framework for analyzing and synthesizing educational progressions," in *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2013.
[36] M. Buro and D. Churchill, "Real-time strategy game competitions," *AI Magazine*, vol. 33, no. 3, pp. 106–108, 2012.
[37] L. Song, M. Ishteva, A. Parikh, E. Xing, and H. Park, "Hierarchical tensor decomposition of latent tree graphical models," in *30th International Conference on Machine Learning*, 2013.
[38] M. O. Riedl and V. Bulitko, "Interactive narrative: An intelligent systems approach," *AI Magazine*, vol. 34, no. 1, pp. 67–77, 2013.

Alexander Zook Alexander Zook is a Ph.D. candidate in Human-Centered Computing at the Georgia Institute of Technology School of Interactive Computing. His research focuses on AI for game design and development, understanding how AI systems can augment or automate human design practices. The results of this research have implications for games used for entertainment, education, and training purposes. Alexander has worked with Blizzard Entertainment and Bioware to improve the design and ongoing development of analytic tools for major massively multiplayer online role-playing games.

Mark Riedl Mark Riedl is an Associate Professor at the Georgia Institute of Technology School of Interactive Computing. Mark's research resides at the intersection of artificial intelligence, storytelling, and virtual worlds, focusing on how intelligent systems can autonomously create engaging experiences for users in virtual worlds. His research has implications for entertainment computing, virtual educational environments, and virtual training. He has published over 80 scientific articles on artificial intelligence, story generation, interactive virtual worlds, and adaptive computer games.