Towards Player Preference Modeling for Drama Management in Interactive Stories

Manu Sharma, Santiago Ontañón, Christina Strong, Manish Mehta and **Ashwin Ram**
Cognitive Computing Lab (CCL)
College of Computing, Georgia Institute of Technology
Atlanta, Georgia, USA
{manus, santi, cstrong, mehtama1, ashwin}@cc.gatech.edu

Abstract

There is a growing interest in producing story based game experiences that do not follow fixed scripts predefined by the author, but change the experience based on actions performed by the player during his interaction. In order to achieve this objective, previous approaches have employed a drama management component that produces a narratively pleasing arc based on an author specified aesthetic value of a story, ignoring a player's personal preference for that story path. Furthermore, previous approaches have used a simulated player model to assess their approach, ignoring real human players interacting with the story based game.

This paper presents an approach that uses a case based player preference modeling component that predicts an interestingness value for a particular plot point within the story. These interestingness values are based on real human players' interactions with the story. We also present a drama manager that uses a search process (based on the expectimax algorithm) and combines the author specified aesthetic values with the player model.

Introduction

A typical problem in creating compelling story based games is to provide the player with a sense of agency during the interaction while simultaneously giving the whole experience an overall narrative structure. There is a growing interest in developing Drama Manager (DM) components that gently guide the player towards a story ending that exhibits a narrative arc. The goal of these components is to allow the player to have significant impact on what happens during the interaction, rather than following along with a pre-written script or being in control at only a few decision points. One of the approaches to drama management, Search Based Drama Management (SBDM) (Weyhrauch 1997), represents stories as a set of plot points and an evaluation function that models the interestingness of a particular sequence of plot points. The Drama Manager (DM) utilizes a set of drama manager actions to gently guide the player towards a story that would maximize the interestingness of the story. DM actions are typically things that can prevent access to certain parts of the world or start a conversation on a specific topic to provide the player with a hint to certain aspects of the story.

Previous approaches to DM development have used an author specified evaluation function to measure interestingness and narrative coherence of a particular story path (Weyhrauch 1997) and have employed a simulated player model to predict the next player action during the interaction (Nelson *et al.* 2006b). However, in experiential interactive systems, the player's preferences must be taken into account. Thus, a player model constructed based on real human players interacting with the game is needed.

In this paper, we present an approach to drama management that explores the link between an author defined evaluation of a given story arc and a player preference model that encodes the interestingness of a story from the player's perspective. To accomplish this, we propose the following:

- A case based approach for building a player preference model that allows the prediction of the plot points and story arcs that the player is going to enjoy during his interaction with the game.
- An approach to drama management based on the expectimax algorithm that is able to take into account the player preference model and the author specified aesthetic rules in order to guide the story towards narrative coherence (by deciding which DM actions to execute at each instant of time).

The rest of the paper is organized as follows. We first present a brief introduction to drama management in interactive stories. Then we present an overview of our proposed architecture. Next, we report a particular implementation of this architecture in the Anchorhead game and present initial empirical evidence of the system. The paper closes with future directions we plan to pursue.

Drama Management in Interactive Stories

Bates (1992) first proposed the idea of treating drama management as an optimization problem. The approach termed Search Based Drama Management (SBDM) was based on the fact that the drama manager chooses its best available action with expectation-calculating nodes and the player is typically assumed to act according to some probabilistic model. Peter Weyhrauch (1997) in his dissertation work further developed the idea of a SBDM with a game tree based search that used an author specified evaluation function to measure the interestingness value for a particular

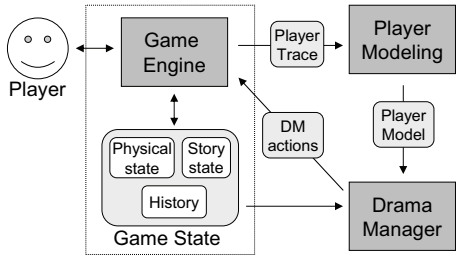

Figure 1: Basic scheme of the three modules that compose a game in the proposed approach.

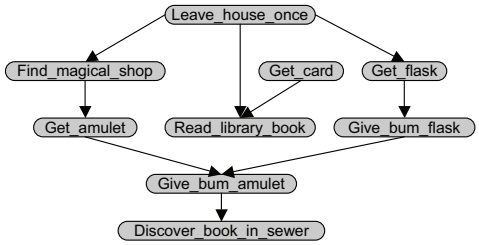

Figure 2: Subset of the plot points used in the story of Anchorhead, defined in (Nelson *et al.* 2006a).

story. However the DM employed was not connected to a concrete story world and the techniques were tested using simulated players rather than real human players. The approach furthermore ignored a player preference model capturing the players preference for a particular story. In another approach, Nelson et. al. (2006) define a Declarative Optimization based approach to Drama Management (DODM). The central premise of their technique is to provide the author with the ability to specify what constitutes a good story and use a reinforcement learning approach to optimize DM actions in response to player actions, given a set of plot points and the evaluation function. This approach also uses a simulated player model to predict the next player action. Furthermore, the approach ignores a player preference model to measure the interestingness of a story from the player's perspective.

In our approach to drama management, we construct a player preference model through real human player interaction with the game. Whereas previous approaches employed only an author based evaluation function for story interestingness, our approach essentially constructs an evaluation function that is a combination of both player and author.

Another approach to drama management is that of Façade (Mateas & Stern 2003), which employs a beat based management system suited towards tighter story structures where all the activity contributes towards the story. The Mimesis architecture (Young *et al.* 2004) proposes a story planning based approach for real-time virtual worlds. In this approach, the story plans are tagged with causal structure and the system handles player actions that might threaten the causal links through either replanning the story or disallowing the player the opportunity to carry out his action. In such an approach to drama management, however, only the author specifies the concrete goals that the planner should achieve; the approach doesn't incorporate a player preference model to guide the player experience during the game episode.

Drama Management Approach

Our approach to drama management consists of three modules (shown in Figure 1) namely: a *game engine*, responsible for actually running the game and interacting with the player; a *player modeling* module, responsible for analyzing the actions of the current player and developing a player model; and a *drama management* module, influencing the development of the game and making it more appealing to the player (represented as the player model). The remainder of this section presents each module in more detail.

Game Engine

The game engine performs three functions: a) handling player input (formalized as *player actions*), b) presenting the game state to the user, including information regarding the kind of actions the player can perform (via an audiovisual interface), and c) maintaining the current game state.

The game engine holds the following components corresponding to the game state:

- The *physical state*, which contains the status of the physical objects and characters in the world, i.e. it contains information such as "the character X is at (x,y,z) coordinates", or "the house's front door is closed".
- The *story state* is essentially represented as a set of *plot points* (Weyhrauch 1997). A plot point is an event that is relevant to the game story (e.g. "the player has discovered the existence of a hidden room" or "the librarian had a car accident"). Plot points are structured as a directed graph, where each plot point acts as a node in the graph and the arcs represent dependencies (a dependency states that, during the game, a particular plot point cannot happen unless another set of plot points have happened before). Figure 2 shows a particular example of a plot point dependency graph. The story state is simply a list that indicates which of the plot points in this graph have been visited. Initially, the story state is empty; indicating that no plot points in the story graph have been visited.
- The *history* contains information on the evolution of the game until the present state, i.e. a trace of player actions from the beginning of the game and other meaningful events during the game execution (e.g. relevant actions performed by other game characters).

Player Modeling Module

The Player Modeling Module (PMM) constructs player models based on the feedback provided by players at the end of each game. This feedback contains player opinions on the game, including the parts they enjoyed and those that were not interesting from their perspective. The goal is to capture the interestingness rating for the story elements encountered by the player during the game episode. At the end of each

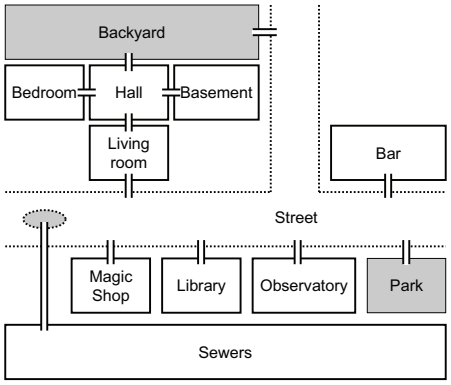

Figure 3: Map of locations in Anchorhead.

interaction, the PMM stores this player feedback along with the corresponding trace of player actions during the game.

During a particular game episode, the PMM utilizes this information to compare the ongoing player trace maintained by the game engine with player traces collected during previous interactions with different players. The feedback from players with the closest matching traces are combined to form an estimate of the stories that the current player is most likely to prefer. The underlying assumption behind this approach is that if the current player's actions follow a pattern that closely resembles those of certain players who have previously played this game, then their interestingness rating for stories would also closely match. The PMM uses this assumption to estimate the stories the current player is likely to enjoy. In general, the player model stored in the PMM contains information on a player's ratings for the locations, plot points or sequences of plot points in the game.

We can distinguish two kinds of player models: the *player preference model* and the *player action model*.[1] The first tries to model the plot points or stories the player is going to enjoy, and the second is trying to model the actions that the player is likely to perform in a given situation. In the experiments reported in this paper, we have only modeled player preferences for different plot points encountered during the game episode.

Drama Manager Module

The input to the Drama Manager Module (DMM) is: 1) the player preference model built by the PMM, 2) the current game state, 3) a set of author specified story guidelines, and 4) a set of possible *drama manager actions*. With this information, the goal of the DMM is to select a drama manager action (if any) to perform at any instance of the game so that the story develops according to both the player model and the author-specified story guidelines.

In the same way that the game author specifies a set of actions that the player might take at each moment of time in the game (player actions), he also specifies a set of *drama manager actions* (DM actions). These actions represent the things that the drama manager can carry out to influence the game, e.g. "prevent the player from entering the library by locking the door" or "make the bar keeper start a conversation with the player about the suspicious person". Basically, the DM actions can be classified in two groups:

- *Causers*, which are designed to lead the player towards a particular direction in the story. Causers can be hints or direct causers, i.e. a hint to the player to go towards a particular direction, or directly make something happen so that the player cannot prevent from going in that particular direction.
- *Deniers*, which are designed to prevent the player to move towards a particular direction in the story. Again, deniers can be hints or direct deniers (the above example about locking the library's door is a direct denier).

Given the set of possible directions in which the story might unfold (as a function of the player's selected actions), the drama manager has to plan a set of DM actions (causers and/or deniers) to guide the player towards story directions that are likely to be more appealing to him, according to the player module and author specified story guidelines. Section *Expectimax-based Drama Manager* presents a particular implementation of the DMM which uses an expectation-maximization planning algorithm to decide DM actions.

To test our approach, we have used it in a interactive story named Anchorhead. We present this particular instantiation of our architecture in the next section.

Using the DM approach in Anchorhead

Overview of Anchorhead

Anchorhead is an interactive story game created by Michael S. Gentry. We have developed a text based interface for interaction with the player. Text based descriptions of the current scenario are presented to the player, who then enters commands in textual form to interact with the game. These commands are essentially phrases that encode the players intentions, e.g. "enter the mansion", encoding the desire to enter into the a new game location (mansion).

Anchorhead is a large game, with a complicated story and several plots and subplots that make it amenable for drama management studies. It has been previously used as a test bed for testing different drama management approaches (Nelson *et al.* 2006a). The story is divided in five days of action. For this paper we have focused on a subpart of the story, identified by Nelson et. al. (Nelson *et al.* 2006a) as interesting for evaluating drama manager approaches. The subpart is based on the second day of the original game. The resulting story has two main subplots, that are related but can lead to a different endings.

In order to test our approach to drama management, we have developed a concrete implementation for the Anchorhead game whose modules are presented next.

Game State Representation

The game state representation is composed of three parts: a physical state, a story state and a history. For Anchorhead, the physical state holds the current location of the player

[1] This action model is what is typically referred to as a "player model" in most drama management literature.

as he is the only active element during the interaction. Other game characters (the bum, the owner of the magic shop, etc.) essentially stay in the same location while waiting for the player to reach them. Figure 3 shows the map of locations in Anchorhead. At any given moment, the player could be, for instance, at the "magic shop" or in the "backyard" and the physical state would hold this information. The story state consists of the plot points that have been visited by the player. The history consists of the list of actions performed by the player till the current time instant, interleaved with the actions that the drama manager has performed.

The story is represented as a set of 32 plot points organized as a Directed Acyclic Graph (DAG). A subset of these plot points[2] are shown in Figure 2. Each plot point is associated with a set of prerequisites, expressed as a boolean formula over other plot points, representing the fact that a plot point cannot happen unless those plot points have been visited. In the Anchorhead game, for instance, the plot point *open_safe* can only happen if both *discover_safe* and *get_safe_combo* have been visited.

The representation for player actions consists of prerequisites and effects that respectively encode the situations under which the action is applicable and the changes in the game state once the action takes place. Specifically, we represent it as a tuple $a = [op, p_l, p_{pp}, e_l, e_{pp}]$, where op is the name of the action, p_l are the prerequisites related to player's location, p_{pp} are the prerequisites related to plot points, and e_l and e_{pp} represent the effects on player location and plot points respectively when the action is executed. In order to understand the representation, lets consider an example:

$$a_1 = \begin{bmatrix} op = use\_combo\_on\_safe \\ p_l = bedroom \\ p_{pp} = discover\_safe \wedge get\_safe\_combo \\ e_l = \emptyset \\ e_{pp} = open\_safe \end{bmatrix}$$

Here, a_1 encodes that the action $use\_combo\_on\_safe$ can only be applied if the player is present in the $bedroom$ and only when the plot points $discover\_safe$ and get_safe_combo have been visited. The action causes no effect on the player location, however, when the action is executed, the plot point $open\_safe$ is marked as visited.

In the next section, we present our current implementation of the PMM, using Case Based Reasoning (Aamodt & Plaza 1994).

Case Based Player Preference Modeling

The case-based player modeling module (shown in Figure 4) is used to predict the interestingness of sequences of plot point for the current player at every instance of the game. As part of its internal representation, it stores records of previous player interactions in the form of cases. These cases encode a combination of the history (i.e. the player trace) and an associated interestingness rating (explained below) elicited at the end of each game episode.

[2]We have used a slightly modified version of the plot points defined in (Nelson *et al.* 2006a).

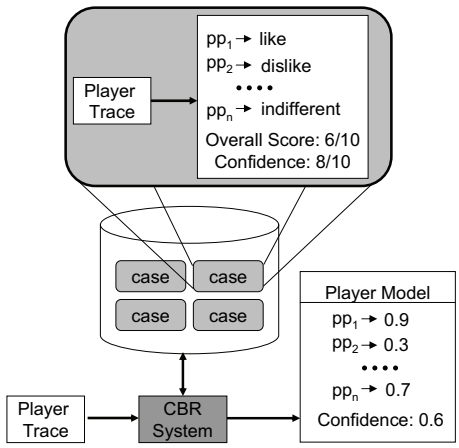

Figure 4: The Player Modeling Module uses a Case Based approach to build the player model.

As a particular player is playing the game, his current trace is compared to the traces within the different cases stored in the case base. In order to facilitate calculating the similarity between these player traces, we have categorized the player actions. For example, the actions to move from one location to another are labeled as *movement*, others that have no effect in the story (e.g. having a nap) are classified as *no_effector*. In order to retrieve cases that are applicable for the current player, his trace is compared with stored player traces. We have used a similarity function that compares the histogram of the different kinds of actions in the corresponding traces.

In our experiments we retrieve the 5 most similar cases and use them as the estimate of the current player's preferences. The interestingness value for each plot point is then computed by aggregating the interestingness values from those three cases by using the similarity metric to weight the individual interestingness values.

In order to develop cases for the player modeling module, players provide feedback at the end of each game. Specifically, we ask each player to fill out a short form. The player is presented with a sequence of the plot points visited in his game. From the list, the player is asked to select his preference of the plot points based on a 5 point scale classification: *strongly like*, *like*, *indifferent*, *dislike* and *strongly dislike*. In addition to this, he is asked to rank the game as a whole on a 5 point scale. The player also provides a confidence value on a 5 point scale. Notice that in our Anchorhead test system, the player model is a player preference model, and we are only modeling the interest of a particular player for each plot point. From each player feedback form, the system can build a case in the following way:

- Player annotations (like that of interest) for each plot point pp_j are converted to a number $\delta(pp_j)$ using the mapping: strongly dislike = -1, dislike = -0.5, indifferent = 0, like = 0.5 and strongly like = 1.
- The overall score provided by the player is converted to a number $s \in [-1, 1]$ by mapping 0 to -1 and 4 to 1.

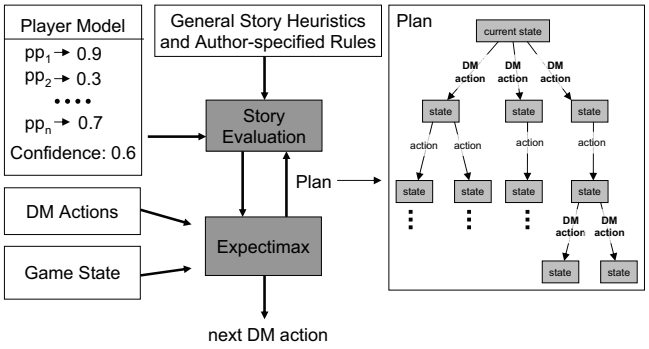

Figure 5: The Drama Manager Module consists of a planner and a story evaluation module.

- The confidence provided by the player is converted to a number $c \in [0, 1]$ by mapping 0 to 0 and 4 to 1.
- The interestingness of each plot point pp_j is computed as $ppi(pp_j) = \frac{\delta(pp_j)+s}{2}$, i.e. the average between the overall score and the particular annotation for that plot point.
- A new case consists of the player trace, the interestingness values for each plot point $ppi(pp_j)$, and the confidence c.

The output of the PMM is a player model that consists of the predicted interestingness of each plot point for the current player and also a confidence on this player model, as shown in Figure 4.

Expectimax-based Drama Manager

The drama management module (DMM) reads the current state of the game and uses the player model to create a more interesting experience for the player. It uses the interestingness values encoded in the player model to increase the probability of visiting certain plot points. In addition, the DMM has a list of general author defined aesthetic rules that it tries to maximize. We have used the *story flow*, *thought flow* and *manipulation* rules defined in (Weyhrauch 1997) for our evaluation.

The DMM has a set of DM actions that it can perform. As explained earlier, a DM action can be either a causer or denier. To illustrate the kind of DM actions available in Anchorhead, consider the following example:

$$a_3 = \left[\begin{array}{l} op = bum\_hints\_crypt\_key\_is\_in\_basement \\ p_l = park \\ p_{pp} = true \\ e_l = \emptyset \\ e_{pp} = \emptyset \\ h_a = \{obtain\_crypt\_key\} \end{array}\right]$$

This DM action a_3 causes one of the characters in the game, the bum, to tell the player that there is a key hidden in the basement. It is important for the player to find this key (hidden in the basement) to advance in one of the subplots. Specifically, this action is a *hint*, and h_a represents a set of player actions at which this DM action hints; i.e. after this DM action has been executed during the game, the player is more likely to choose an action from this set. As the game is going on, the DM will choose to execute this action if it realizes that by providing the key to the player, it will potentially cause the player to reach plot points that he will enjoy.

In order to decide the DM action to executed at each moment of time, the DMM uses an *expectimax* method (Michie 1966). The starting node of the search tree is the current game state (see Figure 5). In the odd plys of the tree, each branch consists of a DM action (including a branch for doing no action). In the even plys of the tree, each branch consists of a player action. In our evaluation, we have kept a fixed depth of 5 plys so that the time required by the DMM to search is not appreciable by the player. For each leaf, l_j, of the tree, we compute an interestingness value $nodei(l_j) = c \times p(l_j) + (1 - c) \times a(l_j)$, where $p(l_j)$ is the average interestingness of the visited plot points in l_j (computed using the player model i.e. using $ppi(pp_k)$ from the cases closest to the state represented by l_j and contains plot points pp_k), $a(l_j)$ is the interestingness obtained using the author defined rules, and c is the confidence suggested by the player model. This essentially aids us in combining the author specified aesthetic values with those of the player model while searching for relevant DM actions.

The interestingness values are propagated up in the tree by selecting the maximum interestingness in the plys with DM actions and the average interestingness in the plys with player actions. Averaging is a good approximation as we do not have a concrete player action selection model. Moreover, if a hint DM action has been executed, then the subtree below it assumes that the hinted actions by that DM action has double the probability of being executed (another approximation due to lack of a player action selection model), and thus that is taken into account when averaging interestingness values in the player action plys. In the end, each of the DM actions in the first ply has an interestingness value (obtained by propagating up the interestingness as described above), and the DMM executes the DM action with maximum interestingness. If the maximum interestingness is given to the branch with no DM action, then no DM action will be executed. The result of this process is that the DMM selects the action that leads the story in a direction that would be more interesting for the player.

Initial Game Runs

We have conducted some initial runs of the game to perform tests on our approach and populate the case base of the PMM. We have run the game with and without the usage of the drama manager. The initial runs provide us with anecdotal evidence on the benefits of using the drama manager approach with the Anchorhead game.

One observation from our initial runs suggests that without the DM, the players spend most of their time exploring the game in a random fashion. The DM with its set of hint actions suggests the player to carry out a certain set of player actions. We observed that players tend to choose the action suggested by the DM and change from an exploratory mode to a goal driven one.

Another observation that appears from our initial runs suggests that the addition of the DM seem to make the char-

acters appear more active. During the game, some of actions that the DM performs cause it to operate on behalf of the game characters. For example, if the player goes to the park (one of the locations in the map) and has the amulet, one of the actions of the DM would cause a game character, the bum, to ask the player to give him the amulet. In the original Anchorhead game, as the game characters are passive and wait for the player to interact with them, the bum would wait for the player to offer him the amulet.

The initial runs seems to suggest that the drama manager tries to follow the author defined aesthetic rules. For instance, trying to provide a hint before a direct causer, as there is an author defined rule that penalizes the causers and deniers. This is because they are limiting to the liberty of the player and should only be used when there is no other way to make the player follow the desired path.

As future work we plan to evaluate our approach with a large player base to test the hypothesis raised during our initial observations.

Conclusion and Future Work

In this paper, we have proposed an architecture to deal with the drama management problem in interactive stories. In particular, our approach combined case-based player modeling and a search-based drama manager and thus incorporated the player model with a set of predefined author aesthetic rules to guide the story in a direction that would be more pleasing for the player. We presented a particular implementation of this approach in the Anchorhead game, and performed an initial evaluation that proves the viability of the approach.

Our main contributions have been (1) the inclusion of an explicit player preference model, that is learned through the interaction with real players, (2) the combination of the player model with a set of author defined aesthetic rules, and (3) the connection of the drama management module to a real game that allows us to perform real player evaluation (to the best of our knowledge, there are no evaluations with real players when using a search-based drama manager).

As part of our future efforts, we plan to perform extensive player evaluations to validate the case based player modeling module and the drama manager module. Since the player's experience is important, we plan to use rich qualitative methods to conduct our future evaluation. Furthermore, we plan to conduct systematic observations to measure a player's patterns of preference. It would allow us to complement and contrast the player model created by the case based player modeling module with humans. Consider an interesting scenario where the current player is given hints based on the player model that was most suited to the player. This hint might cause the player to be able to solve a puzzle in the game with relative ease as compared to the previous players in the player model and this might not be an equally interesting experience. This scenario challenges the assumption that the players with similar game playing behavior will have similar interestingness rating for plot points. We plan to address such issues by making the drama manager provide hints only when the system believes that the player is stuck in the game; thereby aiding the player to shift from an exploratory mode to a goal directed mode.

Finally, in our Anchorhead implementation, we have only used a player preference model. We plan to expand our case based player modeling module to generate player action models that can predict the actions a particular player is likely to make in given situations. Such models will help drama manager to have additional knowledge about the player and can be used to prune the search tree and enhance the maximum depth at which the drama manager can search.

As part of our future efforts, we plan to move from the text-based game Anchorhead to a real time 3D system, where the complexity of drama manager is increased considerably. The real time nature of the domain and the combinatoric explosion of the possible player and DM actions in such a domain would necessitate more advanced planning techniques than the current search-based techniques we have used. Furthermore, we plan to study the development of richer story models, by using a hierarchical plot point representation. It will allow hierarchical planning techniques to be applied, thus allowing the drama manager to see further in its search space and plan a better story for the given user.

Acknowledgements

The authors would like to thank Mark J. Nelson for providing all the information related to the Anchorhead game, and to David L. Roberts and Charles L. Isbell for the feedback and interesting discussions about drama management.

References

Aamodt, A., and Plaza, E. 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications* 7(1):39–59.

Mateas, M., and Stern, A. 2003. Integrating plot, character, and natural language processing in the interactive drama faade. In *Proceedings of the 1st International Conference on Technologies for Interactive Digical Storytelling and Entertainment*.

Michie, J. 1966. Game-playing and game-learning automata. In Fox, L., ed., *Advances in Programming and Non-Numerical Computation*, 183–200.

Nelson, M.; Mateas, M.; Roberts, D.; and Isbell, C. 2006a. Declarative optimization-based drama management in interactive fiction. *IEEE Computer Graphics and Applications* 26(3):33–41.

Nelson, M.; Roberts, D.; Isbell, C.; and Mateas, M. 2006b. Reinforcement learning for declarative optimization-based drama management. In *Proceedings of AAMAS 2005*.

Weyhrauch, P. 1997. *Guiding Interactive Drama*. Ph.D. Dissertation, Carnagie Mellon University.

Young, R.; Riedl, M.; Branly, M.; Jhala, A.; Martin, R.; and Sagretto, C. 2004. An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development* 1(1).