

Declarative Optimization-Based Drama Management (DODM) in the Interactive Fiction *Anchorhead*

Mark J. Nelson, Michael Mateas, David L. Roberts, and Charles L. Isbell, Jr.

Abstract—A drama manager guides a player through a story experience by modifying the experience in reaction to the player’s actions. Declarative optimization-based drama management (DODM) casts the drama-management problem as an optimization problem: The author declaratively specifies a set of plot points in a story, a set of actions the drama manager can take, and an evaluation function that rates a particular story. The drama manager then takes the actions in a way that attempts to maximize story quality. Peter Weyhrauch reported good results using a variant of game-tree search to optimize the use of drama-manager actions. We attempt to replicate these results on another story, *Anchorhead*, and show that search does not perform very well in general, especially on larger and more complex stories. However, we believe that this is a problem with the specific optimization method, not the general approach, and report some results demonstrating the plausibility of applying reinforcement-learning techniques to compute a policy instead of search.

I. INTRODUCTION

WE are interested in automatically guiding experiences in large, open-world *interactive dramas*: story-based experiences where a player interacts with and influences a story. Many modern computer games have (or would like to have) rich, non-linear plotlines with multiple endings, complex story branching and merging, and numerous subplots. As an example of an interactive drama, Figure 1 shows a screenshot from *Façade* [1], in which the player interacts with two college friends whose marriage is falling apart; the player’s actions influence the way events unfold.

Traditionally, story in games is guided by local triggers. Progress in a linear story depends solely on how much of the story has unfolded. In slightly more complex situations, the author can specify condition-action rules (e.g. “if the player is in the room and if the player is carrying a gun, then have the non-player character (NPC) hide behind the counter”). To avoid “holes” in the story, the author must specify believable rules for every combination of conditions a player might encounter, a tedious burden for stories of any complexity. Further, when just specifying local rules, the author will find it difficult to both allow the player significant control over the story’s direction and simultaneously keep it coherent and progressing along some sort of narrative arc.

A *drama manager* is one solution to this problem: A system that watches a story as it progresses, reconfiguring the world to fulfill the author’s goals. A drama manager might notice a player doing something that fits poorly with the current story, and attempt to dissuade him; this can be done by “soft” actions

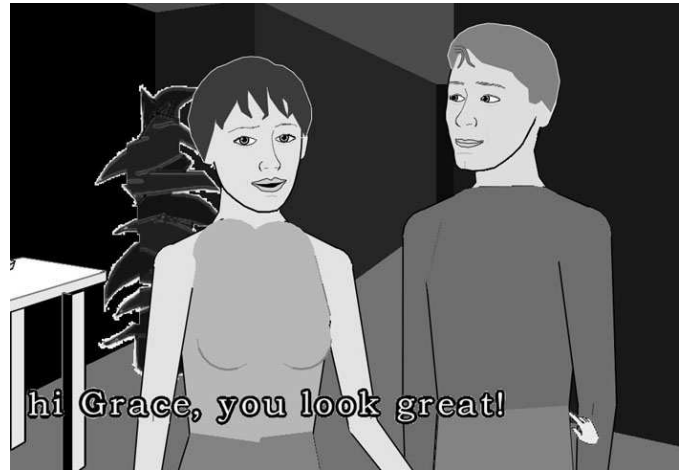


Fig. 1. The interactive drama *Façade*. The player speaks to Grace, an NPC. She will respond in a way that takes into account what the player says but fits with the story.

such as having an NPC start a conversation with a player to lure him to something else, or by more direct actions such as locking doors.

Declarative optimization-based drama management (DODM) guides the player by projecting possible future stories and reconfiguring the story world based on those projections. Stories are modeled as a set of *plot points* that can happen, and an author-specified evaluation function rates the quality of a particular sequence of plot points. The drama manager has a set of *drama manager actions* (DM actions) it can make to modify the world in order to guide the player towards a story that maximizes the evaluation function, taking into account any effect on evaluation the DM actions themselves may have. DM actions might include things such as causing a non-player character to bring up a particular conversation topic, causing certain parts of the world to become inaccessible, or leaving items where they’re likely to be found by the player. At each step, the drama manager chooses the DM action that maximizes projected story quality, subject to a model of what the player is likely to do. This all takes place in an abstract model, connected to the real game by passing messages back and forth, as illustrated in Figure 2: The game tells the drama manager when plot points have happened, and the drama manager tells the game when it wishes to take a DM action.

DODM is a generalization of search-based drama management (SBDM) [2]. SBDM specifically uses a modification of game-tree search to perform the projection, while DODM is agnostic about the projection method used. SBDM rests on

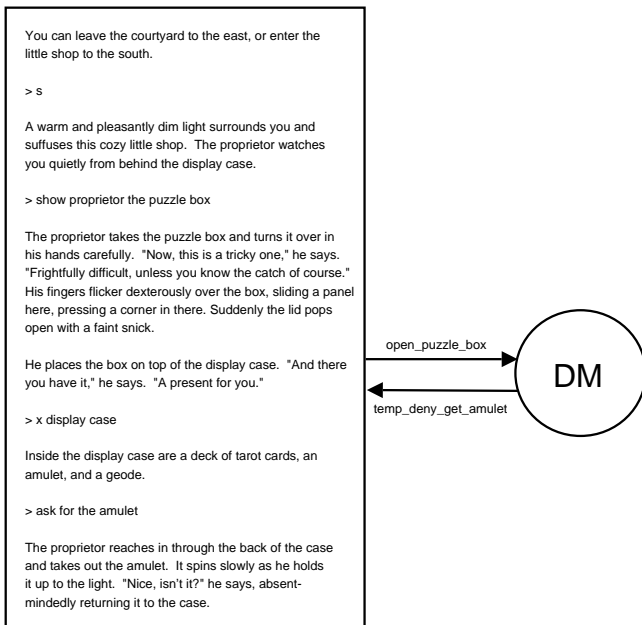


Fig. 2. An excerpt from *Anchorhead*, showing the relationships between concrete game-world actions and the abstract plot points and DM actions. When the proprietor opens the puzzle box, the game recognizes this as the plot point *open_puzzle_box* and tells the drama manager. The drama manager decides on the DM action *temp_deny_get_amulet* and sends it to the game, which implements it by not allowing the player to get the amulet.

two fundamental assumptions. The first, which also applies to DODM, is that an evaluation function can encode an author’s aesthetic; the second, which DODM doesn’t assume, is that search can be used to effectively guide a game’s plot progression to maximize this evaluation function. Weyhrauch [2] demonstrated a proof of concept for both assumptions in a small interactive fiction story, *Tea for Three*, but to what extent these results can be generalized, scaled, and extended isn’t clear.

We present work applying SBDM to the interactive fiction piece *Anchorhead*, in order to further investigate the algorithmic and authorship issues involved. We conclude that Weyhrauch’s results were too optimistic, and are not easily generalizable. In particular, search scales poorly to large stories, and the effectiveness of tractable sampling search depends heavily on the nature of the particular story.

However, we present some work demonstrating the plausibility of using offline temporal-difference learning to compute a policy instead; this approach scales and generalizes much better, and has the important benefit of using very little CPU time during actual gameplay. Given that replacing the search component means that the algorithmic issues appear tractable, we feel that this declarative style of drama management—in which the author specifies what a good story is, rather than procedurally how to achieve it—is an intuitive and authorially-scalable way of guiding player experiences in a wide range of games and interactive stories.

II. RELATED WORK

Search-based drama management was first proposed by Bates [3] and developed by Weyhrauch [2]; reviving the tech-

nique was proposed by Lamstein & Mateas [4]. Weyhrauch applied SBDM to a simplified version of the Infocom interactive fiction *Deadline*, named *Tea for Three*, achieving impressive results in an abstract story space with a simulated player.

The Mimesis architecture [5] constructs story plans for real-time virtual worlds. The generated plans are annotated with a rich causal structure, and the system monitors for player actions that might threaten causal links in the current story plan, either replanning or preventing player action if a threat is detected.

The Interactive Drama Architecture [6] takes a prewritten plot and tries to keep the player on the plot by taking corrective action according to a state-machine model of likely player behavior. DODM, by contrast, tries to incorporate player action into a quality plot rather than insisting on a prewritten plot.

Mateas & Stern [1] developed a beat-based drama manager for their interactive drama *Façade*, using the concept of a dramatic beat. Beats are the smallest unit of change in dramatic value, where dramatic values are character and story attributes such as love, trust, and tension; at each point in the story, a beat-based drama manager selects one of the available beat-level actions. We hypothesize that this style of management makes beat-based drama managers particularly suited to tight story structures, where ideally all the activity in the story world contributes to the story. DODM, on the other hand, lends itself to more open-ended story structures.

For a more detailed review of the drama-management literature (as of 1997), see Mateas’s survey paper [7].

III. ANCHORHEAD

*Anchorhead*¹ is an interactive fiction piece by Michael S. Gentry in the style of H. P. Lovecraft. As compared to the *Tea for Three* story that Weyhrauch investigated, *Anchorhead* has a much larger world, both in terms of the number of plot points and in terms of the size of the world itself (the locations and objects available to the player).

The full *Anchorhead* story is quite large, consisting of well over a hundred significant plot points, making it somewhat unwieldy for initial experiments. In addition, it lacked some features we wished to test, such as the ability for a player to end up in multiple endings based on their actions, and the mixing together of subplots. Fortunately, the story is broken into five relatively separate days of action, so we modified the original second day (the first is very short). We removed some subplots that only make sense in the light of subsequent days so that it would stand on its own, and moved up some events from later days to give a wider variety of potential experiences within the second day. The end result is a story with two main subplots, each potentially leading to an ending.

When the story starts, the player has just arrived in the town of Anchorhead, where her husband, Michael, recently inherited a mansion from a branch of his family, the Verlacs, that he hadn’t been in contact with. The player begins to find out strange things about the town and the Verlac family:

¹Z-machine executable in the Interactive Fiction Archive: <http://www.ifarchive.org/if-archive/games/zcode/anchor.z8>

Edward Verlac, Michael’s brother and previous occupier of the mansion, killed his family and later committed suicide in a mental institution; the townspeople are aloof and secretive; the real-estate agent who had overseen the inheritance is nowhere to be found; and so on. The story then progresses along two interleaved and somewhat related subplots.

In one subplot, the player discovers a safe in which is hidden a puzzle box she’s unable to open. The owner of the town’s Magical Shoppe will helpfully open it, revealing an odd lens. When the lens is inserted into a telescope in the Verlac mansion’s hidden observatory, the player sees an evil god approaching Earth on a comet, reaching the climax of the subplot and a possible ending.

In the other subplot, the player discovers that giving a bum a flask of liquor makes him talkative. Through questioning, the player discovers the bum knows quite a bit about the Verlac family, including a terrible secret about a deformed child, William, who supposedly was killed soon after birth. The bum grows anxious and refuses to give more information until the player finds that William’s coffin contains an animal skeleton. Upon being shown the animal skull, the bum confesses that William is still alive, and confesses his role in the matter. The bum reveals who William is and some of the background of the Verlac family. Parallel to this progression, the bum is afraid for his life and desires a protective amulet the player can get from the shopkeeper; if the player gives it to him, he’ll in return give the player a key to the sewers, in which will be a book revealing the full background, and forming the other possible ending.

IV. MODELING A STORY WITH PLOT POINTS

The first authorial task when applying DODM is to abstract the contents of the story into discrete plot points, each of which represents some event in the story that the drama manager should know about. Sequences of these plot points form the abstract plot space in which optimization will take place.

The plot points are assigned ordering constraints, so that the drama manager only considers possible sequences that could actually happen; for example, the plot point *open_safe* can only happen after both the plot points *discover_safe* and *get_safe_combo* have already happened. (These ordering constraints specify only what *must* happen based on the actual mechanics of the game world—sequences that are undesirable but possible are another matter.)

Weyhrauch specifies these ordering constraints by placing all the plot points in a directed acyclic graph (DAG), with the edges specifying ordering. The possible sequences of plot points are then just the topological orderings of the DAG. We extend this representation by allowing constraints to be arbitrary boolean formulas over the other plot points. This is useful in *Anchorhead*, for example, because the plot point *talk_to_bum_about_william* can only happen once the player has been told of William’s existence, but there are three different plot points that can satisfy this requirement, for which two ORs in the constraint are necessary. (We didn’t use any NOT constraints in *Anchorhead*, but one might easily imagine their use in other stories.)

Figure 3 shows the 29 plot points we used to model *Anchorhead*’s Day 2, arranged into an AND-OR graph showing their constraints.

A. Level of Detail

A difficult question is that of how abstract or specific to make plot points: The more high-level and general they are, the more complex the bits of code that recognize them and signal to the drama manager that they’ve occurred will have to be. We envision those bits of code as very simple and easy-to-code triggers. Some small amount of complexity may be desirable, so that every possible variation in the way events can play out doesn’t need a distinct plot point. However, the amount of complexity needs to be kept quite small so that recognizing plot points does not turn into a problem requiring significant AI of its own.

A similar issue is the question of how fine-grained the plot points should be. For example, a conversation could be a single plot point, *conversation_happens*, or it could be a set of plot points for the major possible conversation topics, or in the extreme case there could be a plot point for every possible line of dialog. As might be expected, there are tradeoffs between fine and coarse modeling. The drama manager cannot make decisions about plot components not represented as plot points, so the more plot points, the more decisions the drama manager can make. However, each added plot point increases the complexity of the optimization problem. More importantly from an author’s perspective, including many relatively unimportant plot points tends to make evaluating plot sequences more error- and noise-prone, as the important plot points are obscured amongst the rest (barring a perfect evaluation function). This leads to the heuristic that any plot point you might conceivably want to change (i.e. cause to happen, prevent, or otherwise modify) with a DM action should be represented, as should any plot point that will have a significant impact on the quality of the story (and so should be visible to the evaluation function); all others should be omitted. This is of course a subjective judgment, and some experimentation is likely the best way to arrive at a reasonable level of detail.

B. Player Modeling

Finding the action that maximizes expected story quality requires some model of how likely it is that various plot points will happen at any given point in the story. We call this a *player model*, since fundamentally which plot points happen and in what order depends on what the player does.

We currently model what we consider a simple but reasonable player: One who has no particular knowledge of the story or the author’s goals, and so is acting in an essentially random manner to explore the story. Specifically, at any given point in the story, we assume that any of the possible plot points (i.e. those whose ordering constraints are satisfied) are equally likely. We also test a variant of this, in which we assume that the player at least sometimes listens to DM actions that hint at particular plot points (see Section V), therefore making those plot points more likely than the rest.²

²These are the same player models Weyhrauch uses.

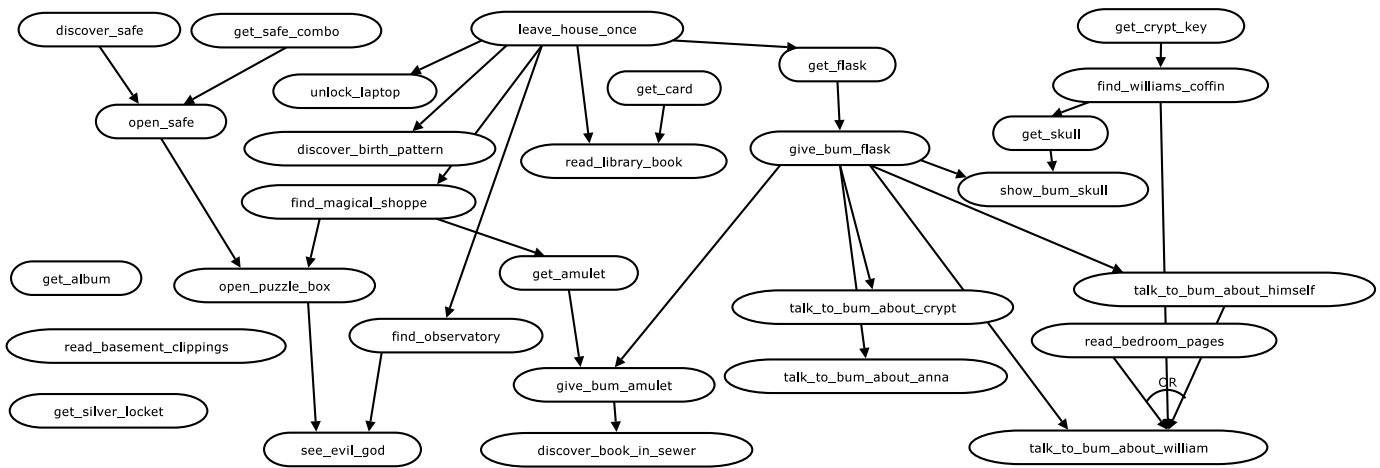


Fig. 3. Plot points modeling *Anchorhead*'s Day 2, with ordering constraints. A directed edge from a to b indicates that a must happen before b , unless multiple edges are joined with an OR arc.

Player modeling impacts how stories should be abstracted into plot points, and is thus important from an authorship perspective. With the current (close to) random-exploration model, things work better if the story is modeled at a fairly uniform level of detail; otherwise, parts of the story modeled in more detail (that is, by more plot points) will be considered more likely (as a whole) to happen.

Of course, there are many possible ways to improve the player model; Section XI discusses several.

V. CHOOSING A SET OF DM ACTIONS

The next authorial issue is choosing a set of DM actions, the “tools” the drama manager will have to work with. There are various types of conceivable actions: They could prevent things from happening; cause them to happen; give hints; and so on. Of course, an action should not simply make strange things happen in front of the player's eyes. If the player hasn't yet found the safe, for example, we can just make it disappear so they'll never find it, but if they've already seen it, we need to be more careful. How to design unintrusive DM actions depends a lot on the story world; one generalization is that it's much easier to do with plot points involving characters, since they can often be plausibly made to start conversations, perform actions, and so on.

A. Types of DM Actions

We're currently investigating five types of DM actions:

- *Permanent deniers* change the world so that a particular plot point becomes simply impossible for the duration of the game. For example, if *find_safe* hasn't happened yet, we can prevent it from ever happening by changing the bookcase with a loose book (behind which the safe is hidden) into just a normal bookcase.
- *Temporary deniers* also change the world so a particular plot point becomes impossible, but only temporarily: Each comes with a paired *undenier* (or *reenabler*) DM action that makes the plot point possible again. For

example, *find_safe* might be reenabled by hiding the safe in some other location the player hasn't yet been to.

- *Causers* simply make a plot point happen. For example, the bum in *Anchorhead* could volunteer information about his past, thereby causing *talk_to_bum_about_himself* to happen.
- *Hints* make a plot point more likely to happen, with an associated multiplier and duration. For example, if the bum tells the player that the crypt key is hidden in the basement of the house, it increases the chances that one of the next few plot points will be *get_crypt_key*.
- *Game endings* are a special type of DM action that ends the game. These are included so that stories can have multiple endings, which are triggered by the drama manager using the same criteria it uses for its other decisions.

In *Anchorhead*, we have 4 deniers, 5 temporary deniers, the 5 corresponding reenablers, 4 causers, 10 hints, and 2 game endings, for a total of 30 DM actions.

B. Issues in Specifying DM Actions

The first issue we ran into was that in a large world like *Anchorhead* not every DM action is appropriate at any given time. The *Tea for Three* world is fairly small, so this was a reasonable assumption, but in *Anchorhead*, it hardly makes sense for the DM to request, for example, that the bum bring up a particular topic in conversation when the player is not even remotely near the bum in the world. As a first step in remedying this, we've added two possible constraints on DM actions: *must-follow* and *must-follow-location*. A *must-follow* constraint allows a DM action to be chosen only immediately after a particular plot point; this is particularly convenient for endings, which usually only make sense to trigger at a specific point. A *must-follow-location* constraint allows a DM action to be chosen only immediately after a plot point that happens in a particular location; for example, we can constrain any DM actions that cause the bum to do something to be legal only following plot points that occur in the bum's vicinity.

An additional issue is that making DM actions too powerful can have negative consequences. This is particularly an issue with permanent deniers, since they force story choices of potentially major consequence that cannot then be undone. If a particular plot-point denial maximizes outcome in, say, 90% of cases, but the player’s playing causes the story to unfold into one of the other 10%, then there is little to do to recover and still push the story towards a reasonable conclusion. Therefore, temporary deniers are preferable, since they can always be undone if necessary; however, permanent deniers are still worth considering, as some potentially useful deniers are very difficult to make believably undoable.

VI. SPECIFYING AN EVALUATION FUNCTION

An evaluation function encodes the author’s story aesthetic declaratively, which is one of the main attractions of DODM (hence the name). The author simply specifies the criteria used to evaluate a given story, annotates plot points with any necessary information (such as their location or the subplot they advance), and the drama manager tries to guide the story towards one that scores well according to that function. In the process of doing so, it makes complex tradeoffs—difficult for an author to manually specify in advance—between possibly conflicting authorial goals (as specified by components of the evaluation function), taking into account the player’s actions and incorporating them into the developing story.

In order to ease authoring, there is a toolbox of features representing common authorial goals. To make weighting various goals straightforward, all features range from 0.0 to 1.0, so an author can specify an overall evaluation function as a weighted combination of the features. It’s worth noting that, at least at present, these generalized features view the plot-point space as “flat”, with each plot point being equally important. Therefore, portions of the story with more plot points will figure more heavily into the evaluation function. This might conceivably be addressed in the future by allowing either a hierarchical space of plot points, or importance values attached to each plot point.

We’re using seven features in our evaluation function for *Anchorhead*, all of which are designed to be applicable to any story where the goal the feature encodes would be desirable.

A. General features

Three features specify general properties we’d like our stories to have.

Location flow is a measure of spatial locality of action: The more pairs of plot points that occur in the same location, the higher the score. This feature is based on a judgment that wandering constantly around the world is undesirable.

Thought flow is calculated similarly to location flow, but measures continuity of the player’s (assumed) thoughts, as specified by an optional *thought* annotation on plot points. This feature can be seen as preferring very short snippets of coherent “sub-subplots”; for example, *get_safe_combo* and *discover_safe* are both annotated with the thought *safe*, so the thought-flow feature would prefer plots in which the player finds the safe and then looks for the combination (or

vice versa), rather than finding the safe, getting distracted by something else, and then finding the combination later.

Motivation is a measure of whether plot points simply happened out of nowhere, or happened after other plot points that motivated them in the player’s mind (this is a subjective determination by the author). For example, first finding the observatory (*find_observatory*) and then noticing that the telescope is missing a lens would make opening the puzzle box and finding a lens inside (*open_puzzle_box*) motivated, while opening the puzzle box without having found the observatory would make the discovery of the lens un-motivated.

B. Features for Stories with Multiple Endings

With multiple subplots leading to multiple potential endings, two additional features are needed to evaluate the interaction of the plots.

Plot mixing measures to what extent the initial part of the story includes plot points from multiple subplots. We’d like the player to explore the world in the beginning, rather than finding one of the plot sequences and going straight to one of the endings.

Plot homing measures to what extent the latter part of the story includes plot points uniformly from the same subplot. This is a counterpart to the plot mixing feature: While we don’t want the player to go straight to one subplot and finish the game right away, we do want them to do so eventually, rather than continually oscillating between subplots and then stumbling upon one of the endings.

C. Meta-features

The final two features are intended to rate the impact of drama management on a story rather than the story itself.

Choices is a measure of how much freedom the player has to affect what the next plot point will be. The goal is to allow the player as many choices of action at any given time as possible, rather than achieving a highly-rated story by forcing the player into one. Without this feature, a drama manager with access to a lot of causers and deniers would basically linearize the story, making the best story as judged by the other features the *only* possible story, which would defeat the entire purpose of an interactive experience. (This feature can be seen as a way of trading off just how much guidance the drama manager should give the player.)

Manipulativity is a measure of how manipulative the drama manager’s changes in the world are. The author specifies a manipulativity score for each DM action, encoding a judgment of how likely it is to be noticed by the player as something suspicious going on (subtle hints might be judged to be less manipulative than outright causers, for example).

VII. EVALUATION METHODOLOGY

The goal of drama management is to improve the quality of experiences over a whole range of possible player behavior. Evaluating drama management therefore requires applying the evaluation function to many stories induced by the player model, and comparing the distribution of story scores between

the drama-managed and non-drama-managed cases. Successful drama management should shift the distribution of story scores to the right as compared to the distribution with no drama management.

We test whether this is the case by generating and scoring random plots to construct the unmanaged distribution, and by running drama management with simulated players (see Section IV-B) to construct the managed distribution. In the experiments reported here, the non-drama-managed distributions were constructed from 10,000 samples each; the drama-managed distributions for search from 100 simulated runs each; and the drama-managed distributions for reinforcement learning from 2000 simulated runs each. All histograms use a bin width of 0.02.

VIII. SEARCH RESULTS

SBDM uses a variant of game-tree search after each plot point to project possible future stories and decide which DM action (if any) to take. The search projects possible future stories that alternate between the drama manager choosing its best action and the player choosing a random action. This is essentially the same as standard MINI-MAX game-tree search, except that the minimizing nodes are replaced with averaging nodes, since we model the player as acting randomly rather than adversarially (as discussed in Section IV-B).

The problem that immediately arises is that actually performing a complete search over all possible future combinations of DM actions and plot points is computationally infeasible, since the search space’s size grows exponentially with the size of the story.

In *Tea for Three*, Weyhrauch implemented a memoized full-depth search, taking advantage of symmetries in the search space to collapse the entire search tree into a table of approximately 2.7 million nodes. However, the memoized search is relatively difficult to author for, since the way to construct the table depends on the particular combination of evaluation features used, and would have to be recoded each time features changed (a process less appealing than specifying an declarative evaluation function to be used by an unchanging search process). In any case, as Weyhrauch notes, even the memoized search doesn’t scale well: In our model of *Anchorhead*’s day 2, the table would have hundreds of millions of entries at least, requiring gigabytes of memory.³

More promisingly, Weyhrauch reported surprisingly good results with SAS, a sampling search. SAS performs search to a specified depth (in our version, iteratively deepening until a time limit), and then obtains a score by averaging together a fixed number of samples of complete plots that could follow the cutoff point. SAS+ is a variant that allows temporarily-denied plot points to appear in the samples, under the assumption that they could be reenabled at some point in the future (necessary in order to prevent the possibility of stories in which no ending is reachable). In *Tea for Three*,

³Our model of *Anchorhead* has an average branching factor of around 4.5, giving approximately $4.5^{29} \approx 9 \times 10^{18}$ possible plot-point sequences. Some of these can be collapsed in the manner Weyhrauch does because they are identical from the point of view of the evaluation function, but the size is still far too large unless a very simple evaluation function is used.

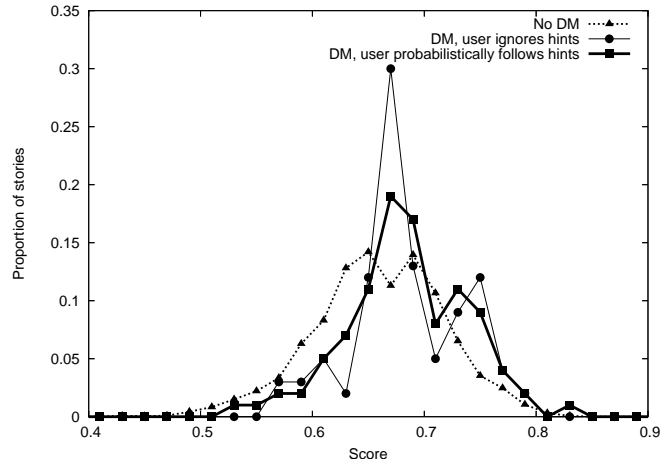


Fig. 4. Distribution of plot qualities with and without drama management. The drama-managed runs were done with SAS+ limited to 2 seconds per decision.

the mean quality of stories produced through SAS+ with a depth limit of 1 was at the 97th percentile of the unmanaged distribution, nearly equaling the 99th percentile obtained by the memoized full-depth search.⁴

The performance of SAS+ on our model of *Anchorhead*, on the other hand, is much less impressive. Figure 4 shows the distribution of plot scores in: an unmanaged story; a story managed by SAS+ with a simulated player ignoring hints; and a story managed by SAS+ with a simulated player probabilistically following hints as the drama manager expects. With the player ignoring hints, the mean score is at the 64th percentile and the median at the 59th; when the player follows hints probabilistically as expected, the mean is still at the 64th percentile, and the median at the 63rd. This is still successful (the overall curve is shifted to the right), but less impressively so than in *Tea for Three*, indicating that the SAS+ results from that story aren’t generalizable.

Indeed, in general, we wouldn’t expect SAS+ to achieve results anywhere near the 97th percentile reported by Weyhrauch: With shallow search depth, a sampling search of this sort is essentially doing local, “greedy” search, at each point choosing the DM action that maximizes the average future plot score under the assumption that no further DM actions will be taken. Since the entire point of DODM is to maximize score taking into account the possibility of future DM actions, this is a significant handicap. The limitation is particularly problematic for DM actions that need to be paired to be effective, such as temporary deniers and their corresponding reenablers.

To determine whether we saw worse results than Weyhrauch due to the set of DM actions we chose, we ran an experiment with causers, temporary deniers, and reenablers for each plot point. These “synthetic” DM actions (synthetic because only a subset are plausibly implementable in the real story world) ought to give the drama manager as complete control as

⁴Since the raw scores are arbitrary numbers, percentiles are reported as a useful relative measure.

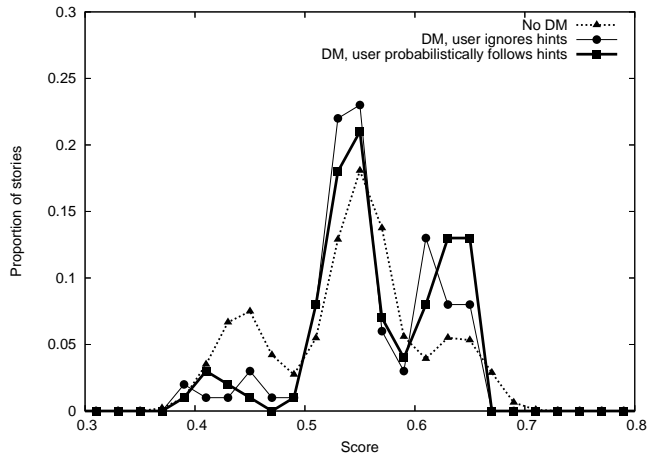
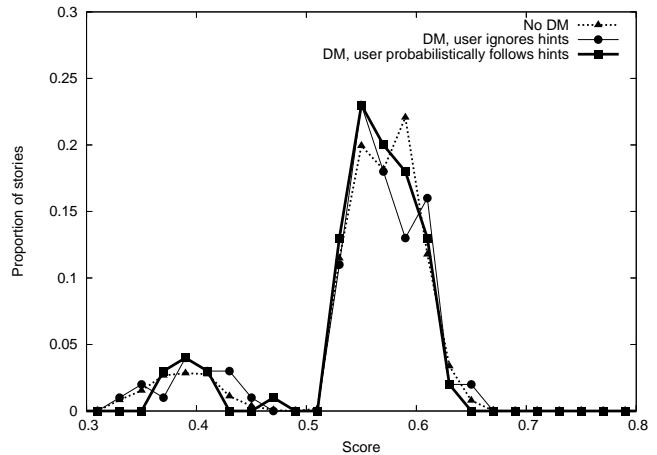
(a) *discover_book_in_sewer* ending.(b) *see_evil_god* ending.

Fig. 6. Distribution of plot qualities with and without drama management on the subplots considered separately. The drama-managed runs were done with SAS+ limited to 2 seconds per decision.

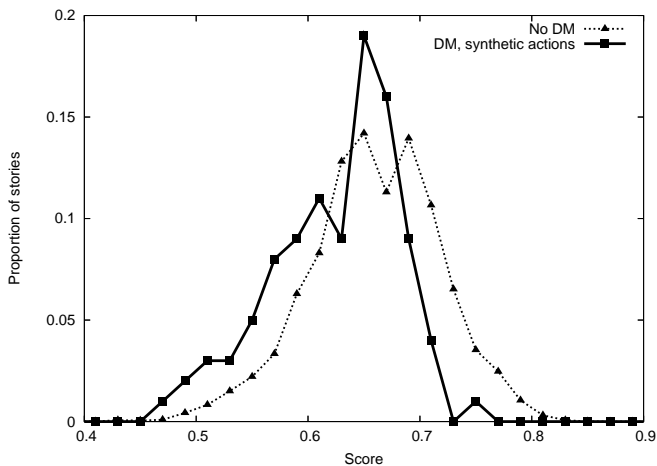


Fig. 5. Distribution of plot qualities with and without drama management, using synthetic DM actions for the drama-managed runs (see text). The drama-managed runs were done with SAS+ limited to 2 seconds per decision.

possible over the story. However, as figure 5 shows, the performance with this set of DM actions is actually *worse* than not using drama management at all! This would be impossible with a search reasonably approximating full-depth search, because even in the worst case the drama manager could avoid actually worsening a story by simply choosing to never take any action. Clearly, then, the difference in distribution quality is due to SAS+ being ineffective on our story.

In order to untangle the effects of multiple endings, we also tried each plot separately, turning off the plot-homing and plot-mixing features and keeping only the plot points and DM actions relevant to each subplot. The results in figure 6 show that the drama manager is much more successful at improving the quality of the stories with the *discover_book_in_sewer* ending than those with the *see_evil_god* ending. One possible

reason is that the storyline ending with *see_evil_god* mostly includes temporary deniers and reenablers as its DM moves, which local search is bad at using: Much like a chess program that searches only 3 moves into the future can't plan 5-move combinations, SAS-style search can't effectively plan using spaced-out pairs of temporary deniers and reenablers to delay plot points.

IX. REINFORCEMENT-LEARNING RESULTS

To address the shortcomings of search, we have been investigating using offline reinforcement learning to precompute a policy, rather than projecting future stories on the fly. There are orders of magnitude more CPU time available to run offline learning than to run online search during actual gameplay, so this allows much more computation to be performed on complex stories than time-limited shallow search does. As an added bonus, drama-manager responses during gameplay are nearly instant.

We train the policy using temporal-difference (TD) learning [8], specifically taking as a starting point the approach used by Tesauro [9] to learn a policy for playing backgammon. Essentially, this consists of running many thousands of simulated games to estimate the values of story states—that is, the value that full-depth search would find for each state (a story state is a description of a partly- or fully-completed story). A full description of TD learning is beyond the scope of this paper, but the general idea is that as a simulation run progresses, the value of each state encountered is estimated based on the current estimate of its successor; since at the end of each story we can use the evaluation function to get a “real” value rather than an estimate for the last state in the sequence, the real values propagate backwards, and all states’

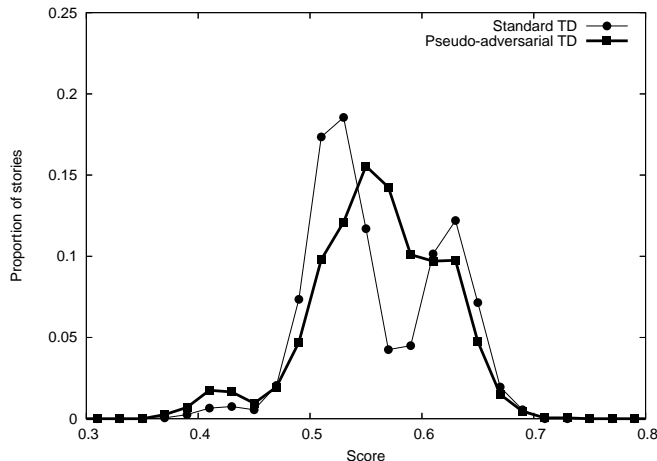


Fig. 7. Distribution of plot qualities using a policy trained with standard TD learning versus one trained with our pseudo-adversarial TD training procedure, on the *discover_book_in_sewer* ending.

estimates should eventually converge to the true values.⁵ The policy is then to simply pick whatever DM action leads to a higher-value state. Since there are far too many possible story states to store the value estimates in a lookup table, we follow Tesauro in training a neural network as a function approximator to estimate a function from states to their values.

One major difference between backgammon and drama management is that there is an adversary in backgammon; some people have hypothesized that this allows the reinforcement-learning agent to explore the edges of and weaknesses in its strategy, since these are constantly being exploited by the adversary (although precisely how this works is controversial). With drama management, the player isn't modeled as actively working against the drama manager, but, as discussed in Section IV-B, as more or less acting randomly. In our experiments, this makes training more difficult, and the resultant policies tend to be relatively mediocre and not very robust.

To address this problem, we train as if the drama-management problem were adversarial: The training runs are done with a simulated player that's actively trying to work against the drama manager. Even when the resultant policy is evaluated against a random player, it is quite good, and significantly better than the policies trained directly against the random player, as shown in Figure 7. This is somewhat surprising, since the standard practice in machine learning is to assume that training samples should be drawn from the same distribution as the test samples. We hypothesize that we get better performance by turning the problem into a pseudo-adversarial one because the adversarial player finds flaws in the drama manager's policy which the drama manager then learns to correct. In theory this could result in a policy specially tuned for the case of an adversarial player, but in practice the positive results even when evaluated with a random player

⁵When not using a function approximator, TD-learning provably converges to the true values (with probability 1) [10]; the neural-network function approximator complicates convergence guarantees, but empirically appears to work well (e.g. in Tesauro's work).

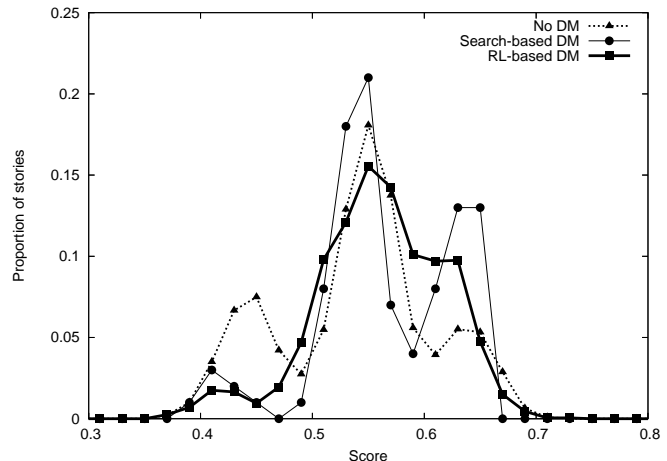


Fig. 8. Distribution of plot qualities with search-based, RL-based, and no drama management, on the *discover_book_in_sewer* ending.

Method	Mean	Median
Search	66.40%	57.62%
RL	72.21%	73.80%

TABLE I

SUMMARY OF RESULTS FOR RL AND SEARCH, IN PERCENTILES.

suggest that many of the policy improvements learned from training against the adversarial player must also improve the policy's performance in the case of a random player.

Since the reinforcement-learning results reported here are still preliminary, we have concentrated on a proof of concept using the *discover_book_in_sewer* subplot. Figure 8 shows story-quality distributions for stories that are unguided, guided by search (as in Figure 6(a)), and guided by a reinforcement-learned policy (using our modified pseudo-adversarial training procedure). As can be seen, the reinforcement-learned policy performs more consistently well than search, avoiding the large gap that is present in the search results, although its upper peak is not as tall. In terms of percentiles, Table I shows that the reinforcement-learned policy has a better mean score, and a significantly better median score. More importantly, it does this without online (game-time) optimization, so we are confident that it is much more easily scalable to the larger stories on which search's performance degrades markedly (although of course this should be tested empirically).

X. CONCLUSIONS

Declarative optimization-based drama management (DODM) is a conceptually appealing way of casting the drama-management problem, as it allows an author to specify what should happen, and offloads the details of making complex tradeoffs in specific cases to an optimization framework. The initial proposal of this style of drama management as search-based drama management, however, reported results that were too optimistic in the general case. Exponential explosion in the size of the search space makes

brute-force full-depth search infeasible. Unfortunately, more tractable shallow sampling searches don't always perform well, and in some cases perform particularly badly. SAS+ does positively impact the quality of *Anchorhead*, but not as effectively as in Weyhrauch's *Tea for Three*, indicating that his positive SAS+ results don't generalize to arbitrary stories. In many ways this is to be expected; the whole point of declaratively casting the drama-management problem is to force the drama manager rather than the author to perform complex tradeoffs among story evaluation features. In general, deep search will be required to perform such tradeoffs.

Reinforcement learning provides a promising alternative to search, however, especially on larger and more complex stories. By doing the optimization offline instead of during the actual gameplay, it is able to perform much more extensive optimization than is possible with online search. However, more experiments are needed to conclusively show that reinforcement learning is able to perform well on a range of stories, including those on which search performs particularly badly.

XI. FUTURE WORK

The most immediate avenue for future work is to further develop the reinforcement-learning-based optimization approach, and demonstrate that it performs well on a variety of stories of significantly size and complexity.

Ultimately, real-world validation is needed in order to verify that this style of drama management actually impacts a player's experience in a real game. The current experiments aim at maximizing the plot-score curves, on the assumption that the author has successfully specified his or her aesthetic in the evaluation function. Checking that assumption itself could be done by evaluating whether actual drama-managed gameplay is judged by players to have improved. Real-world experimentation may also allow a better understanding of how to develop good evaluation functions in the first place.

In addition, development of a more realistic player model would allow for more accurate optimization. The current model of an essentially uniform player is a reasonable first approximation. However, in many story worlds, features of the story world, such as spatial locality or particularly strong motivating goals, will induce non-random patterns on plot-point sequences. One way of improving the model is to recognize that player behavior depends on both the player themselves and on the structure of a particular story. We already do this to some extent by having plot points be annotated with ordering constraints, so our player model doesn't do things that the game simply doesn't allow. Better player models might be built by having the author annotate plot points with more detailed information. For example, if the author provided a rough map of the story world with the locations of each plot point, we could assume (all else being equal) that plot points spatially closer to each other are more likely to happen in sequence. We could even have the author annotate plot points with an estimate of their *a priori* probability; for example, an optional hidden side-quest could be less likely than a conversation with an easily-findable NPC. Another option would be to sidestep

this process entirely and simply build an empirical model by having players play through the game while logging what they do. This is not without its pitfalls either, though, as it would require quite a bit of data to build an accurate player model, especially on a larger story, and data would have to be re-collected if the story were changed in any but the most minor ways.

ACKNOWLEDGMENT

This research is supported by a grant from the Intel Foundation and by the National Science Foundation's Graduate Research Fellowship program.

REFERENCES

- [1] M. Mateas and A. Stern, "Integrating plot, character, and natural language processing in the interactive drama Façade," in *Proceedings of the First International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE-03)*, Darmstadt, Germany, 2003.
- [2] P. Weyhrauch, "Guiding interactive drama," Ph.D. dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1997, Technical Report CMU-CS-97-109.
- [3] J. Bates, "Virtual reality, art, and entertainment," *Presence: The Journal of Teleoperators and Virtual Environments*, vol. 2, no. 1, pp. 133–138, 1992.
- [4] A. Lamstein and M. Mateas, "A search-based drama manager," in *Proceedings of the AAAI-04 Workshop on Challenges in Game AI*, 2004.
- [5] R. M. Young, M. O. Riedl, M. Branly, R. J. Martin, and C. Saretto, "An architecture for integrating plan-based behavior generation with interactive game environments," *Journal of Game Development*, vol. 1, no. 1, pp. 51–70, 2004.
- [6] B. Magerko and J. Laird, "Building an interactive drama architecture," in *Proceedings of the First International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE-03)*, Darmstadt, Germany, 2003.
- [7] M. Mateas, "An Oz-centric review of interactive drama and believable agents," in *AI Today: Recent Trends and Developments. Lecture Notes in AI 1600*, M. Woodbridge and M. Veloso, Eds. Berlin, NY: Springer, 1999, first appeared in 1997 as Technical Report CMU-CS-97-156, Computer Science Department, Carnegie Mellon University.
- [8] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.
- [9] G. Tesauro, "Practical issues in temporal difference learning," *Machine Learning*, vol. 8, pp. 257–277, 1992.
- [10] P. Dayan and T. J. Sejnowski, "TD(λ) converges with probability 1," *Machine Learning*, vol. 14, pp. 295–301, 1994.