Declarative Optimization-Based Drama Management in Interactive Fiction

Mark J. Nelson, Michael Mateas, David L. Roberts, and Charles L. Isbell Jr.
Georgia Institute of Technology

The authors investigate previous research that used game-tree search to optimize drama-manager actions in interactive stories and show that search does not perform well in general.

Our work relates to 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, nonlinear plotlines with multiple endings, complex story branching and merging, and numerous subplots. Figure 1 shows an example of an interactive drama, a screen shot 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, local triggers guide stories in games. 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 (for example, "if the player is in the room and if the player is carrying a gun, then have the nonplayer character 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.

1 The interactive drama *Façade*. The player speaks to Grace, a nonplayer character. She will respond in a way that takes into account the player's responses as well as the story.

A drama manager (DM) is one solution to this problem: it's a system that watches a story as it progresses, reconfiguring the world to fulfill the author's goals. A DM might notice a player doing something that fits poorly with the current story and will attempt to dissuade him or her. This is accomplished using soft actions such as having a nonplayer character start a conversation with a player to lure him or her to something else, or by more direct actions such as locking doors.

We present work applying search-based drama management (SBDM)[2] to the interactive fiction piece *Anchorhead*, to further investigate the algorithmic and authorship issues involved.

Approaches

Declarative optimization-based drama management (DODM) guides the player by projecting possible future stories and reconfiguring the story world based on those projections. This approach models stories as a set of possible plot points, and an author-specified evaluation function rates the quality of a particular plot-point sequence.

The DM has a set of actions it can make to modify the world to guide the player toward a story that maximizes the evaluation function, taking into account any effect on evaluation the DM actions might have. DM actions might include things such as causing a nonplayer character to bring up a particular conversational topic, causing certain parts of the world to become inaccessible, or leaving items where the player is likely to find them. At each step, the DM chooses the DM action that maximizes projected story quality, subject to a model of a player's likely actions.

This all takes place in an abstract model, connected to the real game by passing messages back and forth, as Figure 2 illustrates. The game tells the DM when plot points occur, and the DM tells the game when it wishes to take a DM action. In the figure, when the proprietor opens the

Published by the IEEE Computer Society
0272-1716/06/$20.00 © 2006 IEEE

puzzle box, the game recognizes this as the open_puzzle_box plot point and tells the DM. The DM decides on the temp_deny_get_amulet action and sends it to the game, which implements it by not allowing the player to get the amulet.

DODM is a generalization of SBDM. SBDM specifically uses a modification of a game-tree search to perform the projection, while DODM is agnostic about the projection method. SBDM rests on 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 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. The "Related Work" sidebar (on the next page) discusses other approaches.

You can leave the courtyard to the east, or enter the little shop to the south.

> s

A warm and pleasantly dim light surrounds you and suffuses this cozy little shop. The proprietor watches you quietly from behind the display case.

> show proprietor the puzzle box

The proprietor takes the puzzle box and turns it over in his hands carefully. "Now, this is a tricky one," he says. "Frightfully difficult, unless you know the catch of course." His fingers flicker dexterously over the box, sliding a panel here, pressing a corner in there. Suddenly the lid pops open with a faint snick.

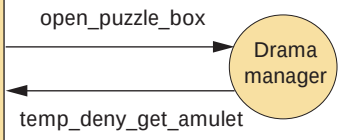
He places the box on top of the display case. "And there you have it," he says. "A present for you."

> x display case

Inside the display case are a deck of tarot cards, an amulet, and a geode.

> ask for the amulet

The proprietor reaches in through the back of the case and takes out the amulet. It spins slowly as he holds it up to the light. "Nice, isn't it?" he says, absentmindedly returning it to the case.



2 An excerpt from *Anchorhead*, showing the relationships between concrete game-world actions, abstract plot points, and drama-manager actions.

Anchorhead

Anchorhead 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, in terms of the number of plot points and 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 lacks some features we wished to test, such as the ability for players to reach 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 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, such as

- 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; and
- the real-estate agent who had overseen the inheritance is nowhere to be found.

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 magic shop 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 William's identity and some of the Verlac family's background. Parallel to this progression, the

Related Work

Bates[1] first proposed search-based drama management (SBDM) and developed by Weyhrauch[2]; Lamstein and Mateas proposed reviving the technique.[3]

Weyhrauch applied SBDM to *Tea for Three*—a simplified version of the Infocom interactive fiction *Deadline*—achieving impressive results in an abstract story space with a simulated player. The Mimesis architecture constructs story plans for real-time virtual worlds.[4] The generated plans are annotated with a rich, causal structure, and the system monitors 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 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.[5] Declarative optimization-based drama management (DODM), by contrast, tries to incorporate player action into a quality plot rather than insisting on a prewritten plot.

Mateas and Stern developed a beat-based drama manager (DM) for their interactive drama *Façade*, using the concept of a dramatic beat.[6] 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 DM selects one of the available beat-level actions. We hypothesize that this style of management makes beat-based DMs 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.

A more detailed review of the drama-management literature (as of 1997) is available elsewhere.[7]

References

1. J. Bates, "Virtual Reality, Art, and Entertainment," *Presence: The J. Teleoperators and Virtual Environments*, vol. 2, no. 1, 1992, pp. 133-138.
2. P. Weyhrauch, *Guiding Interactive Drama*, doctoral dissertation, tech. report CMU-CS-97-109., School of Computer Science, Carnegie Mellon Univ., 1997.
3. A. Lamstein and M. Mateas, "A Search-Based Drama Manager," *Proc. AAAI Workshop Challenges in Game AI*, AAAI Press, 2004.
4. R.M. Young et al., "An Architecture for Integrating Plan-Based Behavior Generation with Interactive Game Environments," *J. Game Development*, vol. 1, no. 1, 2004, pp. 51-70.
5. B. Magerko and J. Laird, "Building an Interactive Drama Architecture," *Proc. 1st Int'l Conf. Technologies for Interactive Digital Storytelling and Entertainment* (TIDSE), Fraunhofer IRB Verlag, 2003.
6. M. Mateas and A. Stern, "Integrating Plot, Character, and Natural Language Processing in the Interactive Drama Façade," *Proc. 1st Int'l Conf. Technologies for Interactive Digital Storytelling and Entertainment* (TIDSE), Fraunhofer IRB Verlag, 2003.
7. M. Mateas, "An Oz-Centric Review of Interactive Drama and Believable Agents," *AI Today: Recent Trends and Developments*, Lecture Notes in AI 1600, M. Woodbridge and M. Veloso, eds., Springer, 1999, pp. 297-328.

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 where she can find a book revealing the full family background, and forming the other possible ending.

Modeling a story with plot points

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

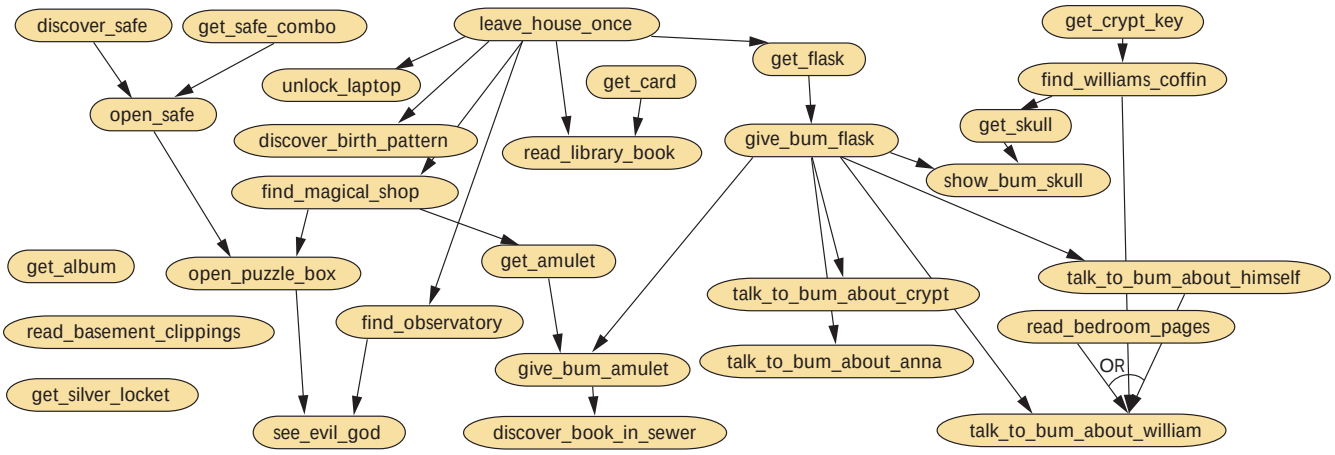
The author assigns plot point ordering constraints, so that the DM only considers possible sequences that could actually happen. For example, the plot point open_safe can only happen after both the plot points discover_safe and get_safe_combo have already happened. (These ordering constraints specify only what must happen based on the actual mechanics of the game world—undesirable but possible sequences 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 DAG's topological orderings. 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 three different plot points can satisfy this requirement, for which two ORs in the constraint are necessary. (We didn't use any NOT constraints in *Anchorhead*, but we can easily imagine their use in other stories.) Figure 3 shows the 29 plot points we used to model *Anchorhead*'s day two, arranged into an AND-OR graph showing their constraints.

Level of detail

A difficult question is how abstract or specific to make plot points: The more high level and general, the more complex the bits of code that recognize them and signal the DM that they've occurred. We envision those bits of code as simple and easy-to-code triggers. Some small amount of complexity might 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 should 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 dialogue. As might be expected, tradeoffs exist between fine and coarse mod-



3 Plot points modeling *Anchorhead*'s day two, 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.

eling. The DM cannot make decisions about plot components not represented as plot points, so the more plot points, the more decisions the DM can make. However, each added plot point increases the optimization problem's complexity. 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 among the rest (barring a perfect evaluation function). This leads to the heuristic that any plot point you might conceivably want to change (that is, 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.

Player modeling

Finding the action that maximizes expected story quality requires some model of the likelihood 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 (that is, those whose ordering constraints are satisfied) are equally likely. We also test a variant of this in which we assume that the player sometimes listens to DM actions that hint at particular plot points, making those plot points more likely than the rest (these are the same player models that Weyhrauch uses.)

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.

Choosing a set of DM actions

The next authorial issue is choosing a set of DM actions—that is, the tools the DM will have to work with. Various types of conceivable actions exist: preventing and causing events, giving 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 the player never finds it, but if the player has already seen it, we need to be more careful. Designing unintrusive DM actions depends a lot on the story world. One generalization is that it's much easier to accomplish with plot points involving characters, since they can often be plausibly made to start conversations, perform actions, and so on.

Types of DM actions

We're currently investigating five types of DM actions:

- permanent deniers,
- temporary deniers,
- causers,
- hints, and
- game endings.

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 house's basement, 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 the DM triggers by using the same criteria it uses for its other decisions.

In *Anchorhead*, we have four permanent deniers, five temporary deniers and the five corresponding reenablers, four causers, 10 hints, and two game endings, for a total of 30 DM actions.

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 added two possible constraints on DM actions: must-follow and must-follow-location. A must-follow constraint allows the DM to only choose a DM action 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 the DM to only choose a DM action immediately after a plot point that happens in a particular location. For example, we can constrain any DM actions that cause the bum to take an action that is 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 percent of cases, but the player's playing causes the story to unfold into one of the other 10 percent, then there is little choice but to push the story toward 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 difficult to make believably undoable.

Specifying an evaluation function

An evaluation function encodes the author's story aesthetic declaratively, which is one of DODM's main attractions. The author specifies the criteria used to evaluate a given story and annotates plot points with any necessary information (such as their location or the subplot they advance), and the DM tries to guide the story toward one that scores well according to that function. In the process of doing so, the DM 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.

To ease authoring, DODM can be used with a feature toolbox representing common authorial goals. To make weighting various goals straightforward, we give all features values within the range of 0.0 to 1.0, so an author can specify an overall evaluation function as a weighted feature combination. At present, these generalized features view the plot point space as flat, with each plot point given equal importance. Therefore, portions of the story with more plot points will figure more heavily into the evaluation function. We might conceivably address this in the future by allowing either a hierarchical space of plot points, or importance values attached to each plot point.

To ease authoring, declarative optimization-based drama management can be used with a feature toolbox representing common authorial goals.

We use seven features in our evaluation function for *Anchorhead*, all of which we can apply to any story where the goal the feature encodes would be desirable.

General features

Three features specify general properties we'd like our stories to have: location flow, thought flow, and motivation.

Location flow is a measure of spatial locality of action: the more plot point pairs 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.

We calculate thought flow in a similar way as location flow, but it measures the player's (assumed) thought continuity, as specified by an optional thought annotation on plot points. This feature prefers 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 occur out of nowhere or happen 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 unmotivated.

Features for stories with multiple endings

With multiple subplots leading to multiple potential endings, we need two additional features—plot mixing and plot homing—to evaluate the plot interaction.

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.

Metafeatures

The final two features—choices and manipulativity—rate the drama management's impact 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 give the player as many action choices at any given time as possible, rather than achieving a highly rated story by forcing the player into one. Without this feature, a DM 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 is a way of trading off just how much guidance the DM should give the player.)

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

Evaluation methodology

Drama management's goal is to improve experience quality 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 nondrama-managed cases. Successful drama management should shift the distribution of story scores to the right 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 to construct the managed distribution. In the experiments reported here, we constructed the nondrama-managed distributions 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 2,000 simulated runs each. All histograms use a bin width of 0.02 for evaluation function values.

Search results

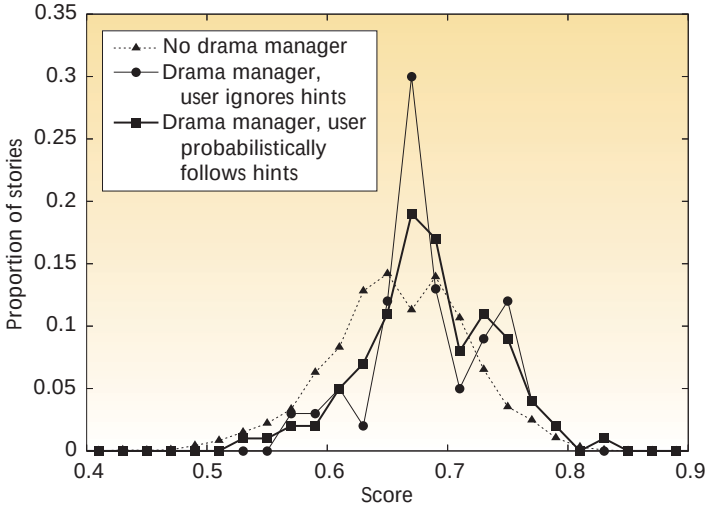
SBDM uses a variant of game-tree search after each plot point to project possible future stories and decide which DM action (if any) to take. The search projects possible future stories that alternate between the DM choosing its best action and the player choosing a random action. This is essentially the same as standard mini-max game-tree search—in which values are backed up a search tree by choosing the minimum value at your opponent's choice points and the maximum value at your own choice points—except that the minimizing nodes are replaced with averaging nodes because we model the player as acting randomly rather than adversarially.

Drama management's goal is to improve experience quality over a whole range of possible player behavior.

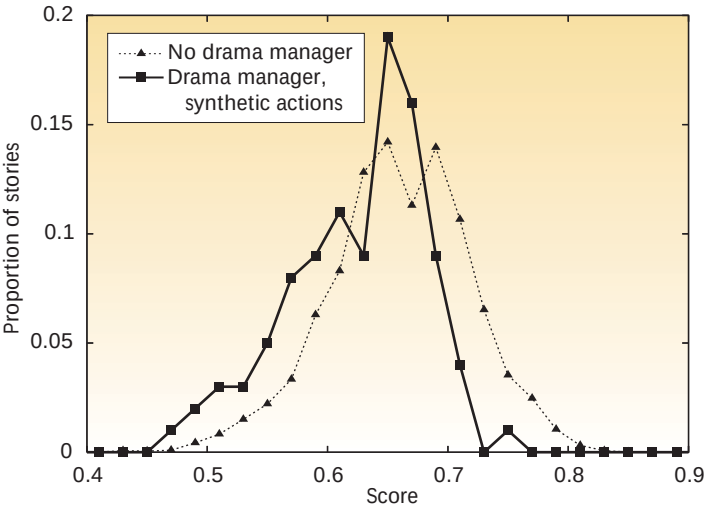
The problem that immediately arises is that actually performing a complete search over all possible future combinations of DM actions and plot points is computationally infeasible because the search space's size grows exponentially with the size of the story.

In *Tea for Three*, Weyhrauch implemented a memoized full-depth search by creating a lookup table, indexed by story states, that contained the drama management moves to take in all possible story states. This was feasible because of symmetries in the search space that allowed him to collapse the entire search tree into a table of approximately 2.7 million nodes. However, the memoized search is relatively difficult to author because 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. (This process is less appealing than specifying a declarative evaluation function for an unchanging search process to use.) Weyhrauch noted that even the memoized search doesn't scale well: in our model of *Anchorhead*'s day two, the table would have a minimum of hundreds of millions of entries, requiring gigabytes of memory.

More promisingly, Weyhrauch reported surprisingly good results with sampling adversarial search (SAS), which performs a sampling version of mini-max 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 complete plot samples 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 stories in which no ending is reachable). In *Tea for Three*, the mean quality of stories produced through SAS+ with a depth limit of one was at the 97th percentile of the unmanaged distribution, nearly equaling the 99th percentile obtained by the memoized full-depth search.



4 Plot quality distribution with and without drama management. We ran the drama-managed runs with SAS+, limited to 2 seconds per decision.



5 Distribution of plot qualities with and without drama management, using synthetic DM actions for the drama-managed runs. We ran the drama-managed runs with SAS+, limited to 2 seconds per decision.

The performance of SAS+ on our model of *Anchorhead*, on the other hand, is much less impressive. Figure 4 shows the plot score distribution in an unmanaged story, a SAS+ managed story with a simulated player ignoring hints, and a SAS+ managed story with a simulated player probabilistically following hints as the DM 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.

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 a 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 DM as complete control as 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 DM 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.

To untangle the multiple ending effects, 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 DM is much more successful at improving the quality of the stories with the discover_book_in_sewer ending (see Figure 6a) than those with the see_evil_god ending (see Figure 6b). 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. Similar to a chess program that searches only three moves into the future and can't plan five-move combinations, SAS-style search can't effectively plan using spaced-out pairs of temporary deniers and reenablers to delay plot points.

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 on complex stories than time-limited shallow search does. As an added bonus, DM responses during gameplay are nearly instant.

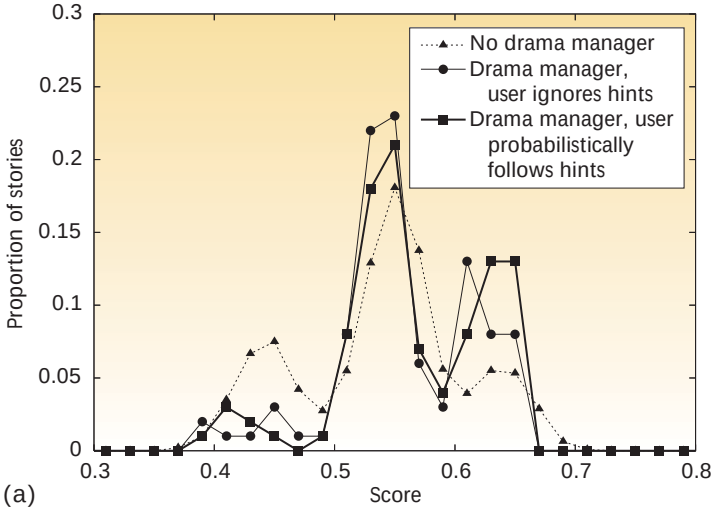
We train the policy using temporal-difference (TD) learning,[3] specifically taking as a starting point the approach used by Tesauro[4] to learn a policy for playing backgammon. Essentially, this consists of running many thousands of simulated games to estimate the values of story states—that is, the 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 article. However, the general idea is that as a simulation run progresses,

TD estimates the value of each state encountered based on its successor's current estimate. This is because 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' 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.

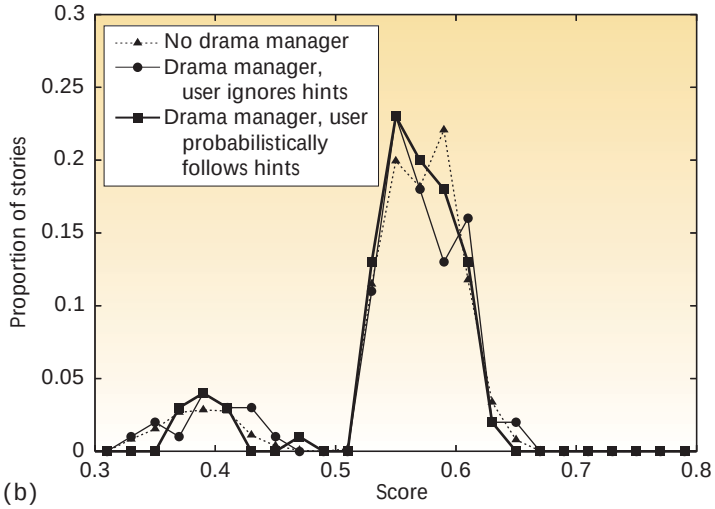
One major difference between backgammon and drama management is that there is an adversary in backgammon. Some people have hypothesized that this allows the reinforcement-learning agent to explore the edges of and weaknesses in its strategy because the adversary constantly exploits these (although precisely how this works is controversial). With drama management, the player isn't modeled as actively working against the DM, but, 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 robust.

To address this problem, we train as if the drama-management problem were adversarial: we perform the training runs with a simulated player who is actively trying to work against the DM. Even when the resultant policy is evaluated against a random player, results are good, and significantly better than the policies trained directly against the random player, as Figure 7 shows. 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 pseudoadversarial one because the adversarial player finds flaws in the DM's policy, which the DM 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—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.

Because 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 (on the next page) shows story-quality distributions for stories that are unguided, guided by search (as in Figure 6a), and guided by a reinforcement-learned policy (using our modified pseudoadversarial training procedure). The reinforcement-learned policy performs more consistently well than search, avoiding the large gap present in the search results, although its upper peak is not as tall. In terms of percentiles, Table 1 shows that the reinforcement-learned policy has a better mean score and a significantly better median score. More importantly, it does this without online (gametime) 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).

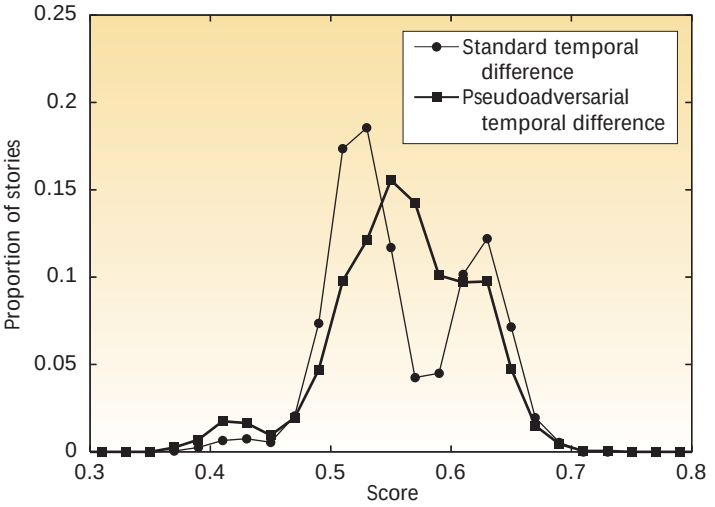


(a)

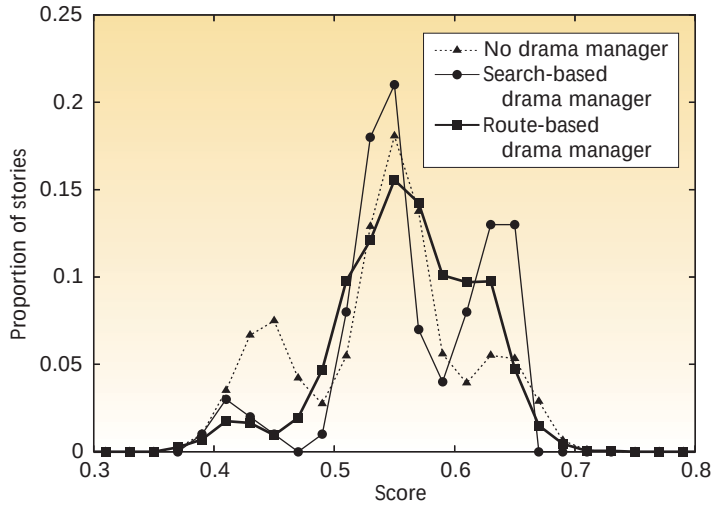


(b)

6 Distribution of plot qualities with and without drama management on the subplots considered separately: (a) discover_book_in_sewer ending and (b) see_evil_god ending. We ran the drama-managed runs with SAS+, limited to 2 seconds per decision.



7 Plot quality distribution using a policy trained with standard temporal-difference learning versus one trained with our pseudoadversarial TD training procedure, on the discover_book_in_sewer ending.



8 Distribution of plot qualities with search-based, reinforcement learning-based, and no drama management, on the discover_book_in_sewer ending.

Table 1. Summary of results for search and reinforcement learning, in percentiles.

Method	Mean (%)	Median (%)
Search	66.40	57.62
Reinforcement learning	72.21	73.80

Conclusions

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 we expected this; the whole point of declaratively casting the drama-management problem is to force the DM 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 performs more extensive optimization than possible with online search. However, we need to do more experiments to conclusively show that reinforcement learning can perform well on a range of stories, including those on which search performs particularly badly.

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 significant size and complexity.

Ultimately, we need real-world validation 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. We could check the assumption itself by evaluating whether players judge as improved actual drama-managed gameplay. Real-word experimentation might 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 nonrandom patterns on plot-point sequences.

One way of improving the model is to recognize that player behavior depends on both the players themselves and on the structure of a particular story. We already do this to some extent by having plot points annotated with ordering constraints, so our player model doesn't do things that the game simply doesn't allow.

We might build better player models 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 nonplayer character.

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 we would have to recollect data if the story were changed in any but the most minor ways. ■

Acknowledgments

This research was supported by a grant from the Intel Foundation and by graduate research fellowships from the National Science Foundation and the US Department of Homeland Security.

References

1. M. Mateas and A. Stern, "Integrating Plot, Character, and Natural Language Processing in the Interactive Drama Façade," *Proc. 1st Int'l Conf. Technologies for Interactive Dig-*

ital Storytelling and Entertainment (TIDSE), Fraunhofer IRB Verlag, 2003.
2. P. Weyhrauch, *Guiding Interactive Drama*, doctoral dissertation, tech. report CMU-CS-97-109., School of Computer Science, Carnegie Mellon Univ., 1997.
3. R.S. Sutton, "Learning to Predict by the Methods of Temporal Differences," *Machine Learning*, vol. 3, no. 1, 1988, pp. 9-44.
4. G. Tesauro, "Practical Issues in Temporal Difference Learning," *Machine Learning*, vol. 8, nos. 3-4, 1992, pp. 257-277.
5. P. Dayan and T.J. Sejnowski, "TD(λ) Converges with Probability 1," *Machine Learning*, vol. 14, no. 3, 1994, pp. 295-301.

Mark J. Nelson *is a National Science Foundation graduate research fellow and is pursuing a PhD at the Georgia Institute of Technology's College of Computing. His research interests include intelligent interactive systems, artificial creativity, and applications of machine learning. Nelson has a BS in computer science from Harvey Mudd College. Contact him at mnelson@cc.gatech.edu.*

Michael Mateas *is an assistant professor at the Georgia Institute of Technology with a joint appointment in the College of Computing and the School of Literature, Communication, and Culture. His research interests include AI-based art and entertainment. Mateas has a PhD. in computer science from Carnegie Mellon. Contact him at michaelm@cc.gatech.edu.*

David L. Roberts *is pursuing a PhD at Georgia Institute of Technology's College of Computing. His research interests include technical issues related to building intelligent artifacts tasked with human interaction. Roberts has a BA in computer science and math from Colgate University. Contact him at robertsd@cc.gatech.edu.*

Charles L. Isbell Jr. *is an assistant professor at the Georgia Institute of Technology's College of Computing. His research interests include developing machine-learning algorithms and tools to enable life-long learning in autonomous systems. Isbell has a BS in computer science from Georgia Tech and a PhD from the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. Contact him at isbell@cc.gatech.edu.*