Search-Based Drama Management in the Interactive Fiction *Anchorhead*[*]

Mark J. Nelson and **Michael Mateas**
College of Computing
Georgia Institute of Technology
Atlanta, Georgia, USA
{mnelson, michaelm}@cc.gatech.edu

Abstract

Drama managers guide a user through a story experience by modifying the experience in reaction to the user's actions. Search-based drama management (SBDM) casts the drama-management problem as a search problem: Given a set of plot points, a set of actions the drama manager can take, and an evaluation of story quality, search can be used to optimize the user's experience. SBDM was first investigated by Peter Weyhrauch in 1997, but little explored since. We return to SBDM to investigate algorithmic and authorship issues, including the extension of SBDM to different kinds of stories, especially stories with subplots and multiple endings, and issues of scalability. In this paper we report on experiments applying SBDM to an abstract story search space based on the text-based interactive fiction *Anchorhead*. We describe the features employed by the story evaluation function, investigate design issues in the selection of a set of drama management actions, and report results for drama managed versus unmanaged stories for a simulated random user.

Introduction

A drama manager guides a player's experience—ideally in an unintrusive manner—in order to create a narratively-pleasing story arc while still allowing the player's actions to be relatively unconstrained. The alternatives are to either write a game in which *all* possible story arcs are reasonably good, or to give up the goal of a narratively-pleasing story arc entirely. We're particularly interested in adding plot to large, open-world games, with a number of subplots and multiple endings, all without enforcing prewritten linear or trivially-branching storylines.

Search-based drama management (Weyhrauch 1997), or SBDM, 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. Search over possible choices of DM actions and the possible resultant sequences of plot points (subject to a model of what the user is likely to do) is run at each step to choose the DM action that maximizes expected plot quality. This all takes place in an abstract model, connected to the real game by passing messages back and forth, as illustrated in figure 1: 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.

SBDM rests on two fundamental assumptions: That an evaluation function can encode an author's aesthetic, and that search can be used to effectively guide a game's plot progression. Peter Weyhrauch (1997) 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, while SBDM remains promising, 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. Further work, especially on algorithmic issues, is necessary before it can be effectively used in real-world stories.

Related Work

Search-based drama management was first proposed by Bates (1992) and developed by Weyhrauch (1997); reviving the technique was proposed by Lamstein & Mateas (2004). 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 user.

The Mimesis architecture (Young *et al.* 2004) constructs story plans for real-time virtual worlds. The generated plans are annotated with a rich causal structure, and the system

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

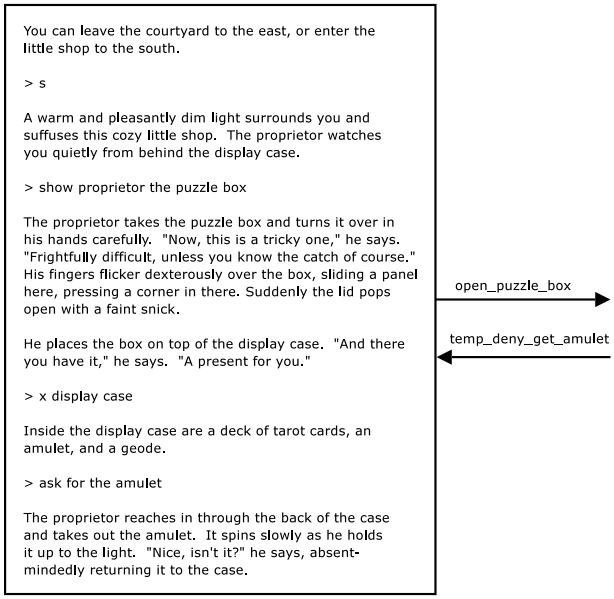

Figure 1: An excerpt from *Anchorhead*, showing the relationships between concrete game-world actions and the abstract plot points and DM actions. When the proprietor opens the puzzle box, the game recognizes this as the plot point `open_puzzle_box` and tells the drama manager. The drama manager decides on the DM action `temp_deny_get_amulet` and sends it to the game, which implements it by not allowing the user to get the amulet.

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 (Magerko & Laird 2003) takes a prewritten plot and tries to keep the user on the plot by taking corrective action according to a state-machine model of likely user behavior. SBDM, by contrast, tries to incorporate user action into a quality plot rather than insisting on a prewritten plot.

Mateas and Stern (2003) 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. This style of management makes them particularly suited to tight story structures, where ideally all the activity in the story world contributes to the story. SBDM, on the other hand, lends itself to more open-ended story structures.

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 physical size of the world itself, such as the numbers of locations and objects available to the player.

When the story starts, the player has just arrived in the town of Anchorhead, where her husband, Michael Verlac, recently inherited the Verlac mansion from a branch of the family he hadn't been in contact with. The player begins to find out strange things about the town and the Verlac family: 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 full *Anchorhead* story is quite large, consisting of well over a hundred significant plot points, making it somewhat unwieldy for initial experiments. Fortunately, it's broken into seven relatively separate days of action, and we've chosen to focus on the second day. We modified the original second day to make it stand on its own by removing some subplots that only make sense in light of the subsequent days and moving some events from later days forwards. The end result was a story with two main subplots, each potentially leading to an ending.

In one subplot, the player discovers a safe in which a puzzle box she's unable to open is hidden. 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.

Modeling a Story with Plot Points

The first authorial task when applying SBDM 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 through which search 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

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

`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 using an AND-OR graph instead of a DAG, allowing for disjunctive constraints in addition to the conjunctive ones a DAG allows; for example, the plot point `talk_to_bum_about_william` in *Anchorhead* 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. (An obvious further extension is to allow full boolean constraints, but that hasn't proven necessary for this particular story.)

Figure 2 shows the AND-OR graph we used to model *Anchorhead*'s Day 2.

Level of detail

A major issue in choosing a set of plot points is the level of detail at which the story should be abstracted. 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 search time, and so in a time-limited environment, decreases search accuracy. In addition, 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.

An additional consideration for the present is that, following Weyhrauch's model, all plot points are considered equally important and equally likely. Thus maintaining a fairly uniform level of detail in modeling will tend to lead to better results: If, for example, one conversation is modeled as a single plot point and another conversation is modeled as ten, each individual plot point in the second conversation will be seen by the evaluation function as being as important as the entire first conversation, and similarly will be modeled as equally likely to take place (of course, this might sometimes be the desired behavior). Future extensions to address this problem could include either weighting plot points with importance values, or modeling stories with a hierarchical space of plot points.

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 user's eyes. If the user 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.

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 user 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.

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

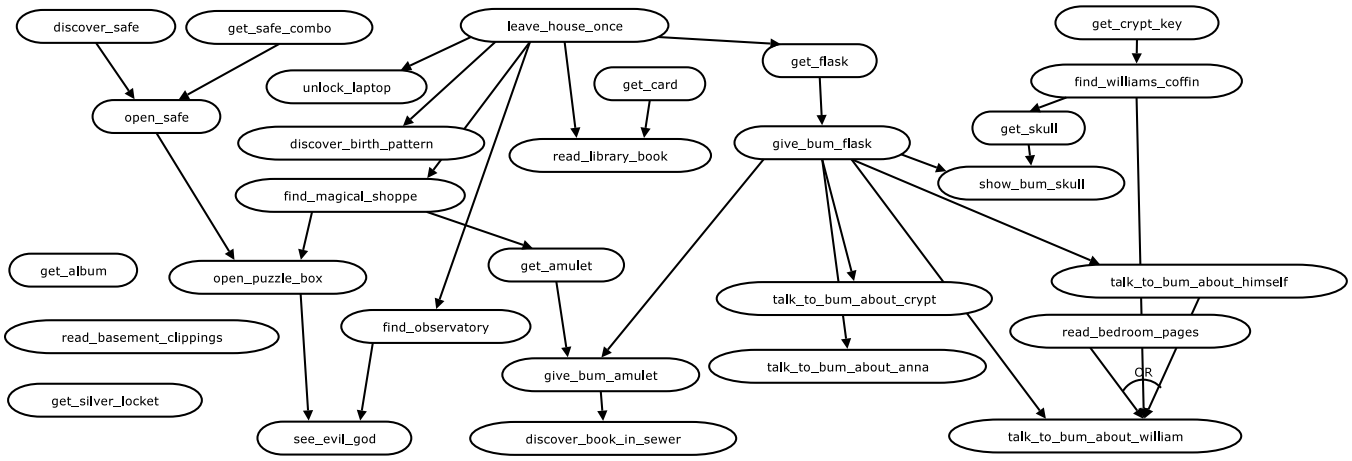

Figure 2: Plot points modeling *Anchorhead*'s Day 2, with ordering constraints. A directed edge from a to b indicates that a must happen before b, unless multiple edges are joined with an OR arc.

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 user'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 undoable (short of an undesirable *deus ex machina* style of drama management).

Specifying an Evaluation Function

An evaluation function encodes the author's story aesthetic declaratively, which is one of the main attractions of search-based drama management. The author simply specifies the criteria used to evaluate a given story, 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), while taking into account the player's actions and incorporating them into the developing story.

Toolbox of Features

In order to ease authoring, an author can choose from a toolbox of features representing common authorial goals. To make weighting 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.

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.

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 user's thoughts, as specified by an optional *thought* annotation on plot points. This feature can be seen as preferring coherent "sub-subplots"; for example, `get_safe_combo` and `discover_safe` both have a thought of `safe`, so the thought flow feature would prefer plots in which the user 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 plots points that motivated them in the player's mind. This is a subjective determination of the author; for example, finding the observatory (`find_observatory`) and noticing that the telescope is missing a lens would make opening the puzzle box to find a lens inside (`open_puzzle_box`) motivated, while finding an unexpected and unexplained lens wouldn't be a motivated plot point.

Features for Stories with Multiple Endings With multiple subplots leading to multiple potential endings, two additional features can evaluate interactions between the plots.

Plot mixing measures to what extent the initial part of the story includes plot points from multiple subplots. We'd like the user 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 plot. This is a counterpart to the plot mixing feature: While we don't want the user to go straight to one plot and finish the game right away, we do want them to do so eventually, rather than continually oscillating between plots and then stumbling upon one of the endings.

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

Choices is a measure of how much freedom the user has to affect what the next plot point will be. The goal is to allow the user as many choices of action at any given time as possible, rather than e.g. using a lot of deniers to prevent the user from doing anything undesirable. (Without this feature, a drama manager with access to a lot of 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.)

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 user as something suspicious going on (subtle hints might be judged to be less manipulative than outright causers, for example).

Searching and Results

Since the goal of drama management is to improve the quality of experiences over a whole range of possible player behavior, not to improve any one particular run, success would mean that the distribution of story scores under drama-managed play is shifted upwards as compared to the distribution with no drama management. We test this by generating and scoring random plots to construct the unmanaged distribution, and by running drama management with simulated users to construct the managed distribution. In the experiments reported here, non-drama-managed distributions were constructed from 10,000 samples each and drama-managed distributions from 100 simulated runs each; all histograms use a bin width of 0.02.

Once plot points, DM actions, and an evaluation function are specified, search through this abstract plot-point space can yield the optimal DM action for any given situation. 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 number of table entries would be in at least the hundreds of millions, requiring a table gigabytes in size.

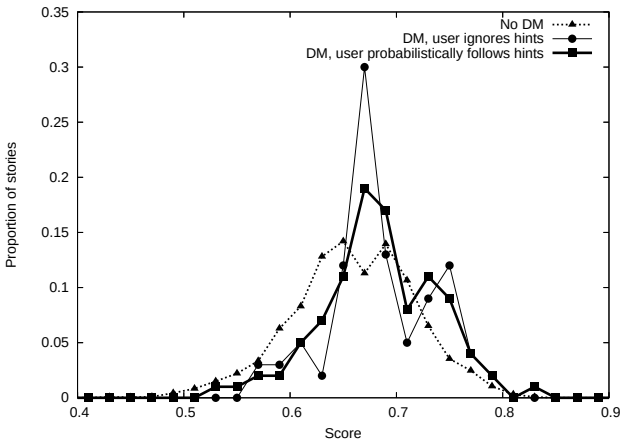

Figure 3: Distribution of plot qualities with and without drama management. The drama-managed runs were done with SAS+ limited to 2 seconds per decision.

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*, the mean quality of stories produced through SAS+ with a depth limit of 1 was at the 97th percentile of the unmanaged distribution, nearly equalling the 99th percentile obtained by the memoized full-depth search.[2]

The performance of SAS+ on our model of *Anchorhead*, on the other hand, is much less impressive. Figure 3 shows the distribution of plot scores in: an unmanaged story; a story managed by SAS+ with a simulated user ignoring hints; and a story managed by SAS+ with a simulated user probabilistically following hints as the drama manager expects. With the user ignoring hints, the mean score is at the 64th percentile and the median at the 59th; when the user 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 upwards), but less impressively so than in *Tea for Three*, indicating that the SAS+ results from that story aren't generalizable.

Indeed, we wouldn't expect SAS+ to achieve results anywhere near the 97th percentile reported by Weyhrauch in

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

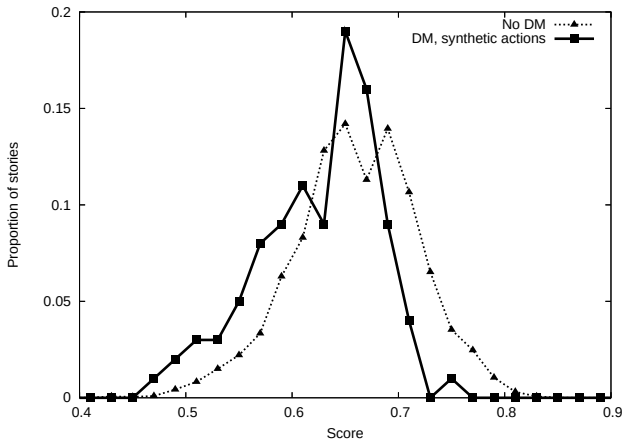

Figure 4: Distribution of plot qualities with and without drama management, using synthetic DM actions for the drama-managed runs (see text). The drama-managed runs were done with SAS+ limited to 2 seconds per decision.

general: 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 SBDM 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 possible over the story. However, as figure 4 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.

Conclusions

Search-based drama management is a conceptually appealing way of casting the drama-management problem. However, the previously-reported results for shallow sampling search were too optimistic. 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 our story, but not as effectively as in Weyhrauch's story, indicating that his positive SAS+ results don't generalize to arbitrary stories. In many ways this is to be expected: The whole point of SBDM is 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. Framing drama management as search is still appealing, as it makes the large body of search techniques and optimizations developed over decades of research available for drama management.

Future Work

The most obviously needed future work is the development of efficient algorithms for approximating the full search in a manner more robust and global than shallow sampling search. Using reinforcement learning techniques to learn a drama manager policy is one promising approach, since it would allow us to offload the bulk of the computation to precomputing the policy, rather than trying to keep up with online searches during actual gameplay.

User modeling would be helpful as well, as the forward-projection search is more accurate when it has a better idea of what is likely to happen in the game, and could also be made more efficient by not considering highly unlikely possibilities.

Ultimately, real-world validation is needed in order to verify that this style of drama management actually impacts a user'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 users to have improved.

References

Bates, J. 1992. Virtual reality, art, and entertainment. *Presence: The Journal of Teleoperators and Virtual Environments* 2(1):133–138.

Lamstein, A., and Mateas, M. 2004. A search-based drama manager. In *Proceedings of the AAAI-04 Workshop on Challenges in Game AI*.

Magerko, B., and Laird, J. 2003. Building an interactive drama architecture. In *Proceedings of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE-03)*.

Mateas, M., and Stern, A. 2003. Integrating plot, character, and natural language processing in the interactive drama Façade. In *Proceedings of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE-03)*.

Weyhrauch, P. 1997. *Guiding Interactive Drama*. Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. Technical Report CMU-CS-97-109.

Young, R. M.; Riedl, M. O.; Branly, M.; Martin, R. J.; and Saretto, C. 2004. An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development* 1(1):51–70.