Evaluating the Authorial Leverage of Drama Management

Sherol Chen,[1] Mark J. Nelson,[1,2] Michael Mateas[1]

[1] Expressive Intelligence Studio
University of California, Santa Cruz
{sherol, michaelm}@soe.ucsc.edu

[2] School of Interactive Computing
Georgia Institute of Technology
mnelson@cc.gatech.edu

Abstract

A drama manager (DM) monitors an interactive experience, such as a computer game, and intervenes to shape the global experience so that it satisfies the author's expressive goals without decreasing a player's interactive agency. Most research on drama management has proposed AI architectures and provided abstract evaluations of their effectiveness; a smaller body of work has also evaluated the effect of drama management on player experience. Little attention has been paid, however, to evaluating the authorial leverage provided by a drama-management architecture: determining, for a given architecture, the additional non-linear story complexity a drama manager affords over traditional scripting methods. In this paper, we propose three criteria for evaluating the authorial leverage of a DM: 1) the script-and-trigger complexity of the DM story policy; 2) the degree of policy change given changes to story elements; and 3) the average story branching factor for DM policies versus script-and-trigger policies for stories of equivalent quality. We apply these criteria to declarative optimization-based drama management (DODM) by using decision tree learning to capture equivalent trigger logic, and show that DODM does in fact provide authorial leverage.

Introduction

Technology can expand the possibilities of narrative both for those who experience and those who tell stories, in particular by making narrative be interactive. Authoring interactive narratives, however, has proven quite challenging in practice. Narrative in games, although sharing some qualities with non-interactive storytelling, delivers a highly interactive experience, which requires new ways of approaching authoring. Traditional approaches to authoring interactive stories in games involve a scripted and heavily linear process, and extending this process to large stories with complicated interactivity is difficult. Drama managers provide an alternative approach, by allowing the author to assume a system that knows something at run-time about how to manage the story. Such approaches, however, are difficult to evaluate from the perspective of authors looking for reasons to use a drama manager rather than traditional authoring approaches.

Authorial *leverage* is the power a tool gives an author to define a quality interactive experience in line with their goals, relative to the tool's authorial *complexity*. It has been pointed out that the "burden of authoring high quality dramatic experiences should not be increased because of the use of a drama manager" [Roberts & Isbell, 2008], but determining whether that is the case depends on determining both the complexity of an authoring approach and the gains it provides.

Previous work has studied how experience quality can be improved by DODM [Weyhrauch, 1997]. This does not directly imply that DODM provides an authorial benefit, however. To do that, there needs to be some reason to believe that traditional authoring methods could not have achieved the same results, or that they would have required considerably more effort to do so.

A way to get at that comparison is to look at the set of traditional trigger-logic rules that would be equivalent to what a drama manager is doing. We propose three criteria for evaluating the authorial leverage of drama managers in this manner: equivalent script-and-trigger complexity of their policies, policy change complexity, and average branching factor of their policies. We present preliminary work applying these metrics to declarative optimization-based drama management (DODM), by examining the equivalent trigger-logic for a drama-manager policy as captured by a decision-tree learner.

Drama Management

In this work, we focus on DODM, an approach to drama management based on *plot points*, *DM actions*, and an *evaluation function* [Weyhrauch, 1997].

Plot points are important events that can occur in an experience. Different sequences of plot points define different player trajectories through games or story worlds. Examples of plot points include a player gaining story information or acquiring an important object. The plot points are annotated with ordering constraints that capture the physical limitations of the world, such as events in a locked room not being possible until the player gets the key. Plot points are also annotated with information such as the plot point's location or the subplot/quest it is part of.

The evaluation function, given a total sequence of plot points that occurred in the world, returns a "goodness" measure for that sequence. This evaluation is a specific, author-specified function that captures story or experience goodness for a specific world. While an author can create custom story features, the DODM framework provides a set of additive features that are commonly useful in defining evaluation functions [e.g. Weyhrauch, 1997; Nelson & Mateas, 2005].

DM actions are actions the DM can take to intervene in the unfolding experience. Actions can *cause* specific plot points to happen, provide *hints* that make it more likely a plot point will happen, *deny* a plot point so it cannot happen, or *un-deny* a previously denied plot point.

When DODM is connected to a concrete game world, the world informs the DM when the player has caused a plot point to happen. The DM then decides whether to take any actions, and tells the world to carry out that action.

Given this model, the DM's job is to choose actions (or no action at all) after the occurrence of every plot point so as to maximize the future goodness of the complete story. This optimization is performed using game-tree search in the space of plot points and DM actions, using expectimax to backup story evaluations from complete sequences.

Related Work

Most related work on drama management is in proposing AI architectures with abstract evaluations of their effectiveness. A few projects have also been evaluated through user tests and simulations: U-Director [Mott & Lester, 2006], PaSSAGE [Thue, et al. 2007], Anchorhead [Nelson & Mateas. 2005], and EMPath [Sullivan, Chen, & Mateas, 2008]. U-Director was evaluated for run-time performance as well quality of the experience with simulated users. The PaSSAGE project was evaluated by 90 players to evaluate the impact of its drama management model on player experience. Anchorhead, the first storyworld using DODM created since Weyhrauch's dissertation, evaluated DODM using simulated players. EMPath, the first fully-playable real-time game built using DODM was evaluated using both simulated and real players, to verify that DODM can indeed improve the quality of the player experience over the un-drama-managed case. Note that all these evaluation approaches focus on player experience, not on the authorability of the drama management approaches. In this paper we propose criteria for evaluating authorial leverage, and apply this criteria to DODM.

Measuring Authorial Leverage

The evaluation of DODM thus far has shown that it can improve the quality of the experience using simulated players [Weyhrauch, 1997; Nelson et al, 2006; Nelson & Mateas, 2008] and real players [Sullivan, Chen, & Mateas, 2008], and has established that the evaluation function can correspond with expert evaluations of experience quality [Weyhrauch, 1997]. None of this establishes the usefulness of DODM for authors, however, if similarly impressive results could have been achieved just as easily using traditional trigger-logic authoring techniques.

Script-and-trigger authoring and DM equivalents

Traditionally, interactive story experiences are authored with sets of scripts and triggers: the author specifies particular events or world states that trigger scripts, which then perform some sequence of actions in response. This typically involves keeping track of various state flags such as which items a player possesses, which NPCs they have talked to, etc., and conditionally triggering actions based on these state flags.

One way to understand the operations of a DM is to generate script-and-trigger logic that encodes a policy equivalent to the DM's. We do that by generating a large set of traces of the DM operating on a number of different stories, and then using a decision-tree learner to summarize the DM's operation. The internal nodes in the learned decision tree, which split on state values, correspond to the tests that exist in triggers; the leaves, which are DM moves, then correspond to scripts to execute. A particular path from the root node to a leaf defines a script to execute given the conjunction of the set of triggers along the path.

Evaluating DM via equivalents

We propose looking at the equivalent trigger-logic formulations of DODM policies to establish the authorial leverage of DODM from three perspectives.

Complexity of script-and-trigger equivalents. First, if the script-and-trigger equivalent of a DM policy is unreasonably complex, then scripts-and-triggers is an infeasible way of authoring that policy. We can determine the smallest decision tree that achieves performance reasonably close to the drama manager, and qualitatively consider whether it would be reasonable to hand-author it. Alternately, we can start with a reasonable hand-authored policy for a small story world, and see how the complexity of required new additions scales as we add additional events and locations in the story world.

Ease of policy change. Second, if experiences can be tuned and altered easily by changing some DM parameters (e.g. the author decides the experience should be faster paced), and the equivalent changes in trigger-logic would require many complicated edits throughout the system, DM adds authorial leverage. DODM in particular uses a number of numerical values/weights/probabilities to define experience goals, which can be changed to re-weight criteria in decisions throughout the story. Drama managers can also allow for changes such as adding or removing story goals in a planning formalism. If simple changes at those levels of authorship result in a significantly different script-and-trigger-equivalent policy, DM effectively allows an author to re-script the original from a compact representation, or to easily create a set of variations on a given experience.

Variability of experiences. The first two leverage metrics are based on the relationship between the amount of work and the quality of the work's outcome. This third measure of leverage is necessary to ensure that there is a variety of diverse experiences in addition to stories of great quality. It is necessary to consider frequency of variability because high quality stories are easily hand authored, although difficult to author in large numbers. An AI system that guides the player on the same high quality experience every time could, according to the first two metrics, yield significant leverage, but would not offer significant leverage according to this third metric. In the subsequent sections we will demonstrate how using DODM simultaneously leads to higher quality and significant variation.

Implementing Decision Trees with DODM

We induced decision trees from example drama-managed story traces using the J48 algorithm implemented in Weka, a machine-learning software package.[1] Each drama-manager decision is made in the context of a partially completed story, so the training data is a set of (*partial-story*, *dm-action*) pairs, generated by running the expectimax search-based drama manager to generate thousands of examples of its actions. Partial stories (the independent variable) are represented by a set of boolean flags indicating whether each plot point and DM action has happened thus far in this story, and, for each pair of plot points *a* and *b*, whether *a* preceded *b* if both happened.

The tree that results can be interpreted as a script-and-trigger system. Each interior node, which splits on one of the boolean attributes, is a test of a flag. The path from a root node to a leaf passes through a number of such flag tests, and their conjunction is the trigger that activates the script, the leaf node that indicates which DM action to take. The tree format consolidates common tests to produce a compact (and inducible from data) representation of the total set of trigger conditions and the scripts they trigger. A given story produces a number of partial story instances, since each step of the story is a decision point for the drama manager.

The tree induced from the drama-management traces can be used as a drama-management policy, specifying the DM action (script) to take in the story states where action is needed (triggers). Decision trees of various sizes can be induced by varying the pruning parameters: a low degree of pruning will effectively memorize the training examples, while a high degree of pruning results in a small tree exhibiting more generalization (and thus more error) across the training examples.

Any of the policies—the actual DM policy or any of the decision trees—can be run with a simulated player to generate a histogram of how frequently experiences of various qualities occur. More successful drama management will increase the proportion of highly rated experiences and decrease that of lower-rated experiences.

Varying the degree of pruning allows us to see how much performance is sacrificed by limiting to a simple script-and-trigger system; or alternately to see what level of script-and-trigger complexity is needed to achieve performance similar to the drama manager. The following subsections will evaluate the authorial leverage of a drama manager by the three performance measures discussed in this paper.

Script and Trigger Equivalents for EMPath

We performed our preliminary evaluations on EMPath, a Zelda-like adventure game [Sullivan, Chen, & Mateas, 2008] that was developed to test DODM in a traditional game genre. It is set in a 25-room dungeon and has, at most, 10 plot points that can possibly occur. In addition to the game, there are 32 DM actions that DODM may choose to employ at various points in the story (33 DM actions when counting the choice to do nothing).

We ran DODM in this world to generate 2500 drama-managed story traces, producing 22,000 instances of training data from which to induce a decision tree. To vary pruning, we varied the maximum terminal node size (number of examples captured by a terminal node), with a larger terminal node size resulting in smaller trees (less splitting on data).

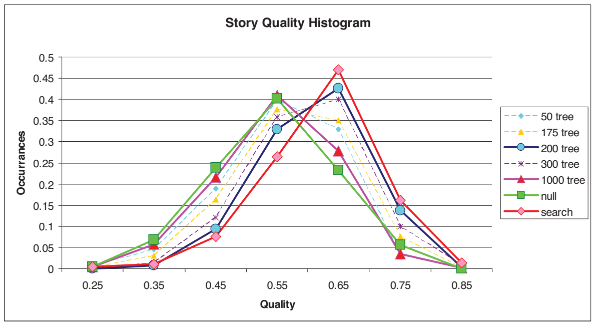

Figure 1. Story quality histogram of search, null, and decision tree policies.

The histogram above shows the performance of the drama manager in the EMPath story world, compared to the performance of a null policy (which always takes no DM actions – this is the un-drama-managed experience) and a number of trees at various levels of pruning. The tree sizes in the legend refer to the maximum terminal node sizes of the different trees. It is apparent that the performance of the smallest trees (greatest pruning), such as the one labeled 1000, performs only slightly better than the null policy, whereas the best match with the search-based policy (the actual DM policy) is found at moderately low levels of pruning (the "200" tree). In addition, the least-pruned trees (e.g. 50) overfit to the particular runs in the training set, as we'd expect, resulting in worse generalization on the test set, and thus do not capture the DM policy well either.

[1] http://www.cs.waikato.ac.nz/ml/weka/

Figures 2 and 3 show the highly pruned (1000) policy with 17 nodes, and the best performing (200) policy with 70 nodes.

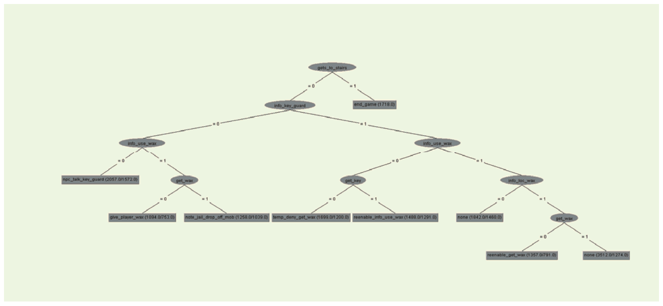

Figure 2. The poorly evaluating decision tree (17 nodes).

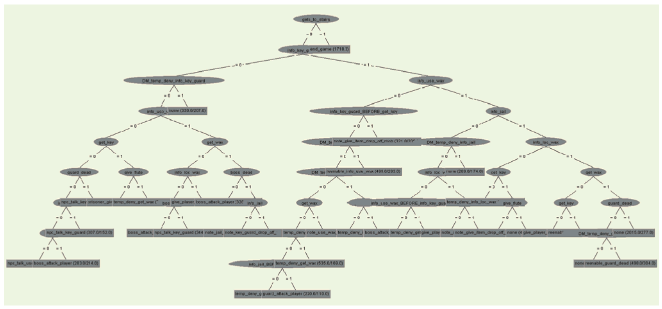

Figure 3. The highest evaluating decision tree (70 nodes).

Although this zoomed-out view gives only a general idea of the policies, the policy in Figure 3 is already clearly quite complex for such a small story world, while the more reasonable policy in Figure 2 doesn't perform well.

Figure 4 gives a zoomed-in view of part of the best-performing tree, showing some of the equivalent script-and-trigger logic that it captures.

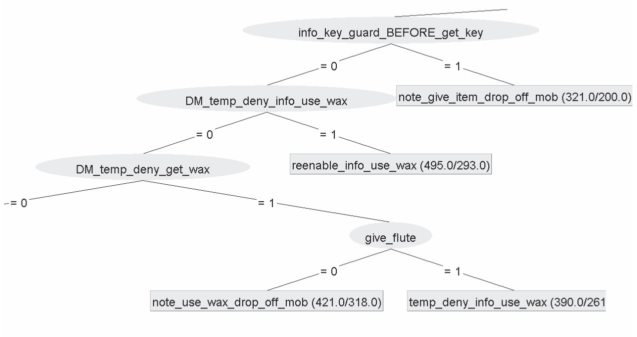
info_key_guard_BEFORE_get_key
= 0
= 1
DM_temp_deny_info_use_wax
note_give_item_drop_off_mob (321.0/200.0)
= 0
= 1
DM_temp_deny_get_wax
reenable_info_use_wax (495.0/293.0)
= 0
= 1
give_flute
= 0
= 1
note_use_wax_drop_off_mob (421.0/318.0)
temp_deny_info_use_wax (390.0/261

Figure 4. Zoomed in view of the 200 pruned tree.

One trace through this segment specifies the following rule. If info_key_guard_BEFORE_get_key is false (i.e. either info_key_guard or get_key plot points haven't happened, or the info_key_guard plot point happened second); and the DM action temp_deny_info_use_wax has not been used; and the DM action temp_deny_wax has not been used; and the plot point give_flute has happened; all conjoined with any tests further up the tree; then take the DM action temp_deny_info_use_wax. This is specifying a series of exclusion tests (in which case other plot points would be appropriate), followed by a choice of what to do if all of them pass, depending on whether the flute has been given yet. Hundreds of these sorts of rules get automatically generated; while they could all be authored by hand in principle, the fact that even in such a small story world it requires a tree of this size to reasonably approximate the DM's performance gives some indication of the infeasibility of doing so.

Ease of Expanding the EMPath world

As a way of testing the ease of policy change (the second authorial leverage criterion), we created three versions of EMPath with increasing world complexity. Table 1 summarizes the three game variants.

	# plot points	# DM actions	# quests	map size
empath-small	10	33	3	25
empath-med	14	47	5	64
empath-large	18	62	6	64

Table 1. Game policy variations.

Each story variant is used to create its own decision-tree training data, by producing 1000 stories each. The training data is built from the partial stories from each 1000-story set. Story worlds that were bigger had larger data sets as a result (8780, 12594, and 16437 respectively).

Recall that the second authorial leverage criterion is ease of policy change. Using DODM, to incorporate the logic for the new subquests into the game, all the author has to do is provide the DM with the new plot points and DM actions, and include the larger world map in the player model (see [Sullivan, Chen and Mateas 2009] for details on the player model). To change the policy for the script-and-trigger-equivalent trees, the author would have to manually add and delete trigger conditions to account for the new content. Given our EMPath variants and the induced script-and-trigger equivalent logic, we need a way of comparing the differences between trees in order to measure the ease (or difficulty) of changing one tree into another. As a simple of measure of this, we find a decision tree that best fits the search-based DODM performance for each EMPath variant (using the same techniques as described above), and compare the sizes of the trees. If the sizes of the trees vary significantly between EMPath variants, then there would be significant authorial difficulty in manually creating new script-and-trigger logic for each variant. Note that, even if the trees are the same size, there could be significant differences between trees, differences that would best be captured with some version of edit distance. But tree size gives us a first approximation of this difference.

Figure 5 graphs the node size of the best-fitting decision tree for each of the variants. There is a significant increase in the node size of the decision tree from empath-small to empath-med and from med to large. Tree sizes grow significantly from empath-small to empath-large, meaning that, to expand the game from empath-small to empath-large, the author would have to make hundreds of edits to the script-and-trigger-equivalent logic.

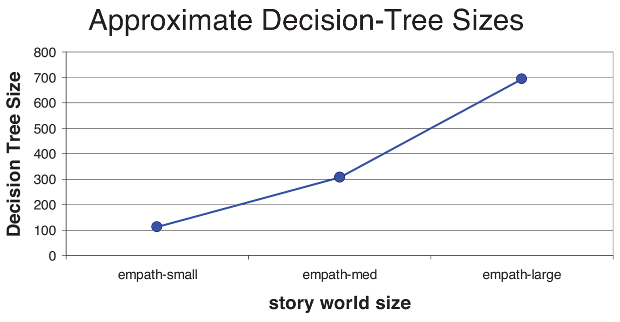

Figure 5. Approximated complexity for the most optimal decision tree policy.

To determine whether, using the second criterion, DODM provides authorial leverage, we need to compare these hundreds of edits with the authoring work required using DODM. To include 8 additional plot points and 29 drama manager actions, the author must describe each plot point and action to the DM. Plot points and DM actions are defined by a list of attribute/value pairs.

Consider the `get_sword` plot point as an example of one of the 8 new plot points added to expand from empath-small to empath-large.

- `get_sword`
- `QUEST = sword`
- `MOTIVATED_BY = {info_sword, info_loc_sword}`
- `COORD = 6 0`

The `quest` attribute describes which quest the plot point is part of, the `motivated_by` attribute describes the list of plot points that should motivate, for the player, this plot point happening, while the `coord` attribute stores the initial map location at which this plot point will occur (initial location of the sword, which can potentially be moved around by drama management actions). When evaluating the quality of potential future sequences of plot points, the evaluation function will use the attribute values to determine the quality of a particular sequence; for example, the evaluation function would decrease the rating of a sequence in which `info_sword` and `info_loc_sword` *don't* happen before `get_sword`, because the player acquiring the sword is not motivated in that sequence.

Now consider `give_player_sword`, one of the 29 drama management actions added to expand empath_small to empath_large.

- `give_player_sword`
- `CAUSES = get_sword`
- `MANIPULATION = 0.9`

This DM action can force the plot point `get_sword` to happen by making an NPC walk up and give the sword to the player (with appropriate dialog from the NPC). The `manipulation` attribute indicates how manipulative the player is likely to find this action (how rail-roaded the action might make them feel). The value of 0.9 (1.0 is maximum) indicates that this is a strongly manipulative action.

In addition to defining plot points and drama manager actions, the author also defines an evaluation function, expressed as a linear weighted sum of evaluation features. An example of an evaluation feature is one that scores how motivated the events in a plot point sequence are, that is, how often, for each plot point in the sequence, its `motivated_by` plot points happen earlier in the sequence. The author can tune the relative importance of the different features by adjusting the weights associated with each feature. Adjusting the weights of the evaluation features determines characteristics for the overall quality metric used to evaluate the story. So, even without adding any additional plot points or DM moves, the author can adjust the experience purely by changing evaluation features or adjusting weights. Thus, another way to measure ease of policy change would be to learn decision trees for several different weightings and evaluation feature combinations, and measure how different these trees are from each other. In this paper, we only address policy change associated with adding new plot points and DM moves.

Variability of Stories for EMPath

The final measure for authorial leverage is in the variety of quality experiences. The simplest way to measure variety is to sum up the total of unique stories. Figure 6 shows the histogram for number of unique stories (out of 50,000 simulated player runs) in the empath-small story world for trees of decreasing size, where the leftmost tree is the best fitting tree. The first thing to note is that the tree that best matches the DODM policy, the 137 tree, still produces over 6000 unique stories (unique sequences of plot points). Thus, DODM is not forcing a small number of stories to always occur. Second, note that as we move towards smaller trees (increased generalization), the number of unique stories grows (more than 14000 in the smallest tree). But we know from Figure 1 that smaller trees result in worse story-quality histograms. Thus, the higher script-and-trigger complexity of the larger tree (the DM-equivalent tree) is producing an increase in story quality while still supporting a wide-variety of experiences.

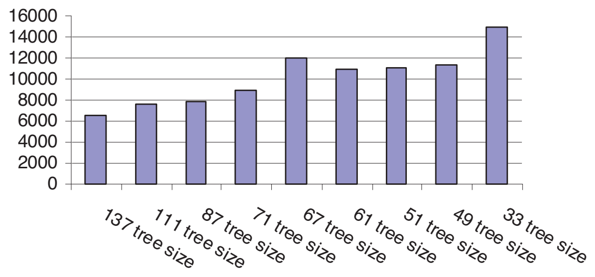

Figure 6. Histogram for unique stories according to tree size.

Decision-tree policy issues

Although decision trees are a nice way of automatically capturing the DM policy in a way that can be interpreted as a script-and-trigger system, there are a few difficulties with the policies they produce. The generalization that takes place in decision-tree induction can produce choices of actions that would not be permitted in a particular state. Since the decision tree learner doesn't have access to internal constraints used by DODM, it may make unsafe generalizations. Two instances where the decision trees produced invalid choices of DM action are: 1) taking *causer* DM actions that cause plot points which have already happened; and 2) not knowing that *denier* actions for critical plot points must be *reenabled* eventually.

These are in effect uncaptured additional complexities in a correct DM policy that a script-and-trigger system would need to deal with. An improvement to the decision-tree induction that might capture them would be to produce a number of negative examples of such disallowed choices of DM actions, and use a decision-tree induction algorithm that allows negative class examples.

Conclusions and future work

We proposed that a major open issue in the evaluation of drama managers is their *authorial leverage*: the degree of authorial control they provide over an interactive experience as compared to the complexity of the authoring involved. Since authoring drama-manager-like interaction in stories is commonly done via scripts and triggers, we proposed that one way to evaluate the authorial leverage a drama manager gives is to use decision trees to induce and examine a script-and-trigger equivalent form of a drama manager's policy. We proposed three criteria with which to do the comparison: 1) examine the complexity of the induced script-and-trigger representation; 2) consider the ease with which stories can be rebalanced or changed by changing DM parameters versus editing scripts and triggers (in this paper, the changes studied involve scaling storyworlds); and 3) examine the variability of stories produced by a script-and-trigger system and a DM policy, e.g. the implied branching factor of the experience.

We presented results in inducing a script-and-trigger equivalent form of a DODM policy in a Zelda-like world, EMPath, and evaluated it by our first proposed criterion, showing that the resulting policies are quite complex to hand-author even in this small domain. Secondly, we showed three versions of EMPath that vary in size, and measured how the decision tree equivalents scaled with these changes. This showed that adding a few plot points to the story world had drastic increases in decision tree complexities. Finally, we showed that using DODM leads to simultaneously higher quality and lots of variation, by examining the variety and frequency of unique stories in conjunction with their story-quality evaluations.

Three primary directions that future work should take are: evaluating other systems, developing further ways of investigating the performance measures, and making use of the learned script-and-trigger systems. The evaluation measures will need to be applied to other story systems in several story worlds, and ideally, would also compare DODM to other drama-management approaches using a similar evaluation of authorial leverage. The three approaches we took to evaluate DODM can be further refined; for instance, performing a more rigorous statistical analysis or using the average branching factor to measure story variation. In addition to evaluating the authorial leverage of drama management, the script-and-trigger systems demonstrated that decision tree policies were drastically faster at run time, although building the trees may take days to preprocess. Future work should examine how these learned script-and-trigger policies can be used at runtime as a "compiled" version of the optimization-based drama manager.

References

Magerko, B. 2007. A comparative analysis of story representations for interactive narrative systems. *Proceedings of AIIDE 2007*.

Mott, B. W. and Lester, J.C. 2006. U-director: a decision-theoretic narrative planning architecture for storytelling environments. *Proceedings of AAMAS 2006.*

Nelson, M.J. and Mateas, M. 2005. Search-based drama management in the interactive fiction Anchorhead. *Proceedings of AIIDE 2005.*

Nelson, M.J. and Mateas, M. 2008 Another look at search-based drama management. *Proceedings of AAAI 2008*.

Nelson, M.J. and Roberts, D.L. and Isbell Jr, C.L. and Mateas, M. 2006. Reinforcement learning for declarative optimization-based drama management. *Proceedings of AAMAS 2006.*

Roberts, D.L. and Isbell, C.L. 2008. A survey and qualitative analysis of recent advances in drama management. *International Transactions on Systems Science and Applications* 4(2).

Sullivan, A., Chen, S., and Mateas, M. From Abstraction to Reality: Integrating Drama Management into a Playable Game Experience. In *Proceeding of the AAAI 2009 Spring Symposium on Interactive Narrative Technologies II*, AAAI Press, 2009.

Sullivan, A., Chen, S., and Mateas, M. 2008. Integrating drama management into an adventure game. *Proceedings of AIIDE 2008*.

Thue, D., Bulitko, B., Spetch, M., and Wasylishen, E. 2007. Learning Player Preferences to Inform Delayed Authoring. In *Proceedings of the AAAI 2007 Fall Symposium on Intelligent Narrative Technologies*.

Weyhrauch, P. 1997. *Guiding Interactive Drama*. PhD dissertation, Carnegie Mellon University.