Evaluating the Authorial Leverage of Drama Management

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

[1] Expressive Intelligence Studio
University of California, Santa Cruz
{sherol, anne, 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 work on drama management has proposed AI architectures and provided abstract evaluations of their effectiveness. A smaller body of work has evaluated the effect of drama management on player experience, but little attention has been paid to evaluating the authorial leverage provided by a drama management architecture: determining, for a given DM architecture, the additional non-linear story complexity a DM 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 vs. script-and-trigger policies for stories of equivalent quality. We present preliminary work towards applying these metrics to declarative optimization-based drama management, using decision-tree learning to capture the equivalent trigger logic for a DM policy.

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 drama management. This does not directly imply an authorial benefit, however. To show 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 where the plot point happens or its subplot.

The evaluation function, given a total sequence of plot points that occurred in the world, returns a "goodness" evaluation 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 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.

Measuring Authorial Leverage

The evaluation of DODM thus far has established at least preliminary positive results for: the technical features of optimization (Weyhrauch, 1997; Nelson et al, 2006; Nelson & Mateas, 2008), the effect on player experience (Sullivan, Chen, & Mateas, 2008), and the correspondence of some evaluation functions to expert notions 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.

One way to understand the operations of a DM is to generate a set of script-and-trigger logic that acts equivalently to the way the DM does. 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 then correspond to scripts to execute, represented by DM actions. 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 it 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 a trigger-logic equivalent 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. Other drama managers can 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 noticeably 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 were based off of 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. Its 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 had the same high quality experience every time could, according to the first two metrics, yield significant leverage.

Decision Trees

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 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 at the leaf node, represented by a DM action to take. The tree format is simply a compact (and inducible from data) representation of the total set of 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

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

degree of pruning captures a small script-and-trigger system that accounts for as much of the DM's behavior as possible, given the small permitted tree.

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.

DM policy evaluations in 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). The figure below shows the world's 10 plot points and their required precedence relationships:

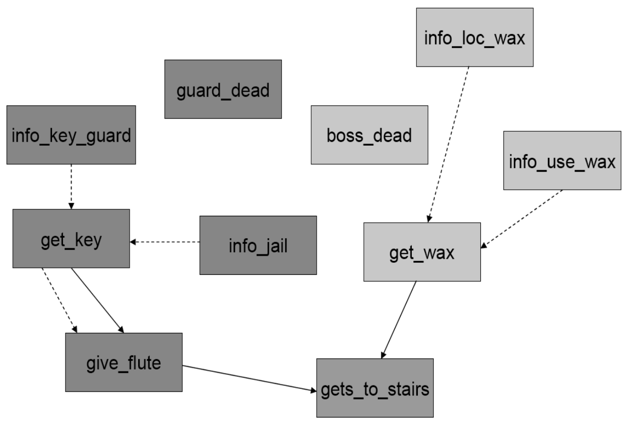

Figure 1: This is the directed-acyclic graph for the story world

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 minimal terminal node size, with a larger minimal terminal node size resulting in smaller trees as splitting does not continue as far.

The following histogram 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) and a number of trees at various levels of pruning. 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

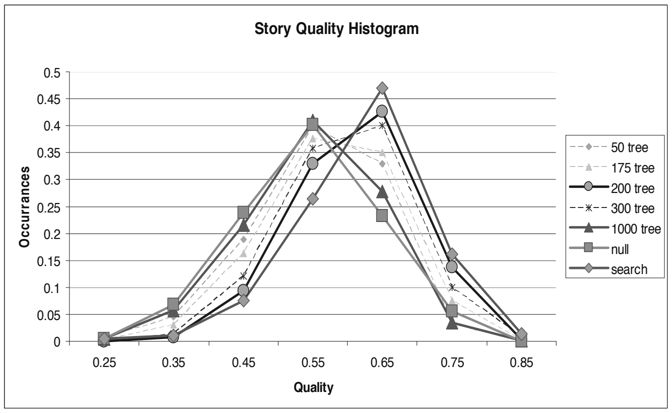

Figure 2: Story quality histogram of search, null, and decision tree policies.

the best match with the search-based policy (the actual DM policy) is found at moderately low levels of pruning, labeled 200. 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 overall performance, so aren't a good capture of an equivalent policy.

The tree trees below represent the highly pruned (1000) policy, and the best performing (200) policy, respectively:

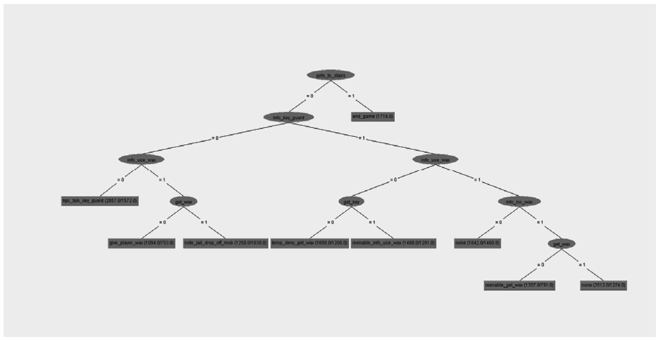

Figure 3: The poorly evaluating decision 1000-tree (17 nodes).

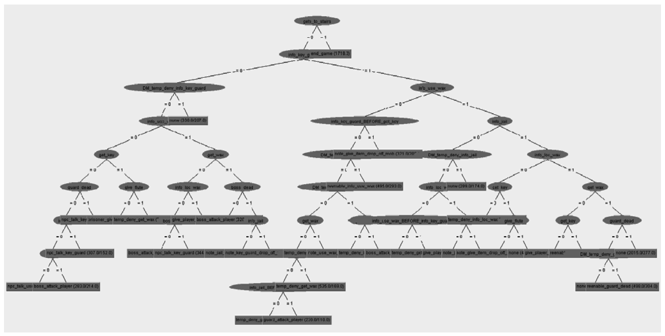

Figure 4: The highest evaluating decision 200-tree (70 nodes).

Although this zoomed-out view gives only a general idea of the policies, the second policy is already clearly quite complex for such a small story world, and the more reasonable first policy empirically doesn't perform as well.

The zoomed-in view of part of the best-performing (200) tree below shows some of the equivalent script-and-trigger logic that it captures:

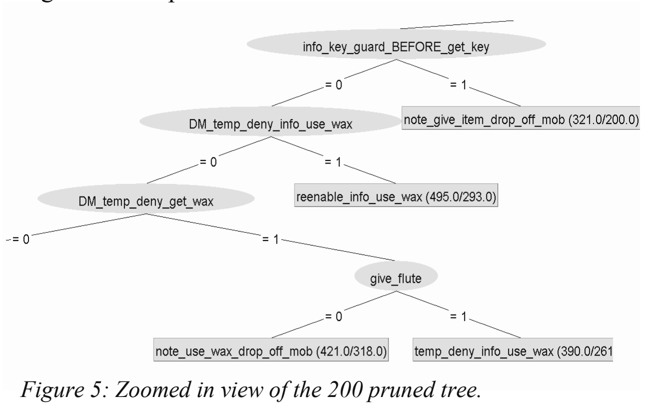
info_key_guard_BEFORE_get_key
= 0
= 1
DM_temp_deny_info_use_wax
note_give_item_drop_off_mob (321.0/200.0)
= 0
= 1
DM_temp_deny_get_wax
reenable_info_use_wax (495.0/293.0)
= 0
= 1
give_flute
= 0
= 1
note_use_wax_drop_off_mob (421.0/318.0)
temp_deny_info_use_wax (390.0/261

Figure 5: 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 (for which other DM actions would be appropriate), followed by a choice of what to do if all of them pass; that choice depends 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.

Decision-tree policy issues

Although decision trees are a nice way of automatically capturing a form of a DM policy 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: The decision-tree-learning algorithm has no notion of the internal structure of DODM so may make unsafe generalizations. Two instances where the decision trees produced invalid choices of DM action were: 1) taking *causer* DM actions that cause plot points which 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 that authoring. Since authoring drama-manager-like interaction in stories is commonly done via scripts and triggers, we proposed that one way to evaluate 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) consider the complexity and complexity scaling with story size of the script-and-trigger versions; 2) consider the ease with which stories can be rebalanced or changed by changing DM parameters versus editing a set of scripts and triggers; and 3) examine the variability of stories produced by a script-and-trigger system and a DM policy, e.g. by their branching factors.

We presented preliminary results in inducing a script-and-trigger equivalent form of a DODM policy in a Zelda-like world, EMPath, and evaluating it by our first proposed criterion, showing that the resulting policies are quite complex to hand-author even in this small domain. Future work should evaluate DODM according to all three criteria, in several story worlds; ideally, it would also compare DODM to other drama-management approaches using a similar evaluation of authorial leverage.

There are additional ways to evaluate DM authoring that could be developed. For example, the authoring metaphors used by various drama-management systems can be compared (Magerko, 2007), as can the ease of authoring within each system's metaphor, e.g. the ease of specifying DODM's evaluation function, and similar parameters in other systems.

References

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

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. 2008. Integrating drama management into an adventure game. *Proceedings of AIIDE 2008*.

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