Targeting Specific Distributions of Trajectories in MDPs*

David L. Roberts[1], **Mark J. Nelson**[1],
Charles L. Isbell, Jr.[1], **Michael Mateas**[1], **Michael L. Littman**[2]
[1] College of Computing, Georgia Institute of Technology, Atlanta, Georgia, USA
[2] Department of Computer Science, Rutgers University, Piscataway, New Jersey, USA
{robertsd, mnelson, isbell, michaelm}@cc.gatech.edu, mlittman@cs.rutgers.edu

Abstract

We define **TTD-MDP**s, a novel class of Markov decision processes where the traditional goal of an agent is changed from finding an optimal trajectory through a state space to realizing a specified *distribution of trajectories* through the space. After motivating this formulation, we show how to convert a traditional MDP into a TTD-MDP. We derive an algorithm for finding non-deterministic policies by constructing a trajectory tree that allows us to compute locally-consistent policies. We specify the necessary conditions for solving the problem exactly and present a heuristic algorithm for constructing policies when an exact answer is impossible or impractical. We present empirical results for our algorithm in two domains: a synthetic grid world and stories in an interactive drama or game.

Introduction

In many interesting real-world problems, the path that you take to the goal is as important as whether or not you get there. Consider the problem of drama management in game worlds. The subjective qualities of a story are determined not just by its ending, but by the entire trajectory of story events. The goal here is to guide a player through a space of possible story trajectories that are consistent with the author's intent while still preserving as much player autonomy as possible. The story trajectory is a function of both the player's actions and the drama manager's reconfigurations of the story world. The author prefers some story lines to others; additionally, the player will want to play the game or participate in the interactive drama more than once. Variety of experience becomes crucial for replayability, therefore, a formulation of this as an optimization problem in story space is inadequate since it will tend to prefer a specific trajectory. Instead, we propose posing this as the problem of learning a stochastic policy that matches a desired probability distribution over stories. We call this optimization problem a *targeted trajectory distribution MDP* (TTD-MDP).

Although we focus on drama management, our formulation applies to a variety of tasks, including agents in online games, virtual tour guides, and robotic entertainers, where in contrast to traditional goal-based systems, the *path* from start to goal is what defines the quality of the solution. In general, whenever an intelligent agent is expected to produce a variety of experiences in the face of a variety of users (or the same user multiple times), we believe that TTD-MDPs are a useful characterization of its task.

In the remainder of this paper, we formally derive TTD-MDPs and optimality criteria. We show how to compute exact optimal policies when possible and introduce an algorithm applicable when an exact solution is either impossible or impractical. Lastly, we verify our methods empirically by presenting results on both an artificial grid world and on an instance of a significantly larger drama-management MDP.

Motivation: Drama Management

As noted above, our derivation of TTD-MDPs is inspired by the growing body of work in drama management and related areas. Bates' (1992) and Weyhrauch's (1997) approach to this problem is based on plot points, actions provided to the drama manager (DM), and a story-evaluation function. Plot points are abstract representations of story events (such as the player finding an object or receiving some important information), annotated with information useful in evaluation and prerequisites specifying valid orders; DM actions are specific ways the drama manager can intervene in the story (*e.g.* by having a computer-controlled character walk up to the player and start a conversation); and the evaluation function, provided by the author, rates the quality of completed stories, taking into account trajectories that consist of the sequence of both plot points and DM actions.

Treated as a traditional optimization problem, the goal of a drama manager is to optimize the use of available DM actions in response to a player's actions, given the set of plot points and an evaluation function. We previously modeled this as a reinforcement learning (RL) problem (Nelson *et al.* 2006) where the goal is to learn a policy that, given a world modeled by states and a transition function, specifies an action to take in each state in a manner that maximizes the expected value of the story-evaluation function. In this case, state is specified by the sequence of plot points and DM actions that have happened so far; actions are the DM actions (including doing nothing); transitions are the likelihood of plot points occurring in any given state, and are given by the

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

combination of prerequisites for a plot point, the player, and any influence from drama-manager actions; and the evaluation function is the one provided by the author.

While this approach has proven promising, it is clear that it has its limitations. Specifically, if there is a policy that allows the drama manager to guide the player to the same highly-rated story every time, the RL process will consider that an effective solution even though this limits replayability. Any distribution over good outcomes is an artifact of the stochasticity introduced by the user rather than the result of reinforcement learning. As such, it becomes difficult to optimize for the *distribution*.

Rather than attempting to maximize any given episode, the goal should be to maximize long-term enjoyment by achieving a target distribution over highly-rated episodes. In other words, one should learn a stochastic policy that produces a distribution over actions rather than a policy that chooses a specific action.

TTD-MDPs

A typical MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $P : \{\mathcal{S} \times \mathcal{A} \times \mathcal{S}\} \rightarrow [0, 1]$ is a transition function, and $R : \mathcal{S} \rightarrow \mathbf{R}$ is a reward function. The solution to an MDP is a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$. An optimal policy ensures that the agent receives maximal long-term expected reward.

A TTD-MDP shares components with a traditional MDP. Specifically, a TTD-MDP is defined by a tuple $(\mathcal{T}, \mathcal{A}, P, P(\mathcal{T}))$, with states $\mathcal{T}$ that are partial or complete finite-length trajectories of MDP states; a set of actions $\mathcal{A}$; a transition model P; and a target distribution over complete trajectories $P(\mathcal{T})$.[1] Note that the target distribution replaces the reward function. The solution to a TTD-MDP is a policy $\pi : \mathcal{T} \rightarrow P(\mathcal{A})$ providing a distribution over actions in every state. The optimal policy results in long-term behavior as close to the target distribution as possible.

Any finite-length discrete-time MDP can be converted to a TTD-MDP. Consider an MDP with a set of states $\mathcal{S}$ and sets of actions available in each state $\mathcal{A}_s$, the probability $P_{i+1}(s')$ that the process is in state s' at time $i + 1$ is defined recursively by:

$$P_{i+1}(s') = \sum_{\forall s \in \mathcal{S}, a \in \mathcal{A}_s} (P(s'|a, s) \cdot P(a|s) \cdot P_i(s)) \qquad (1)$$

where $P(s'|a, s)$ is the transition model encoding the dynamics of the world and $P(a|s)$ is the policy under the agent's control. During an actual episode, $P_i(s) = 1$; if we assume (as is commonly done) that the policy is deterministic, we get a common form of Equation 1, rewritten as: $P_{i+1}(s') = \sum_{\forall s \in \mathcal{S}, a \in \mathcal{A}_s} P(s'|\pi(s), s)$.

Because we are interested in trajectories in TTD-MDPs, we can simply roll the history of the MDP states into the TTD-MDP trajectories, resulting in an TTD-MDP where each trajectory represents a sequence of states in the underlying MDP, optionally including a history of the actions taken. Note, that because we require a finite number of trajectories for the TTD-MDP, in this formulation the MDP cannot have cycles.

Dealing with trajectories means that the "state" space of the TTD-MDP forms a tree. The power of this insight becomes evident when we restate Equation 1 for TTD-MDPs:

$$P(t') = \sum_{\forall a \in \mathcal{A}_t} (P(t'|a, t) \cdot P(a|t)) \cdot P(t). \qquad (2)$$

In other words, for every partial or full trajectory t', the transition probability $P(t'|a, t)$ is nonzero for exactly one $t \sqsubset t'$ that is its prefix. This observation follows from the fact that each trajectory represents a unique sequence of states $s_1, \ldots, s_{\|t\|}$ and therefore has a unique prefix. Thus, the summation need only account for possible actions taken in the preceding trajectory rather than actions in multiple MDP states. Because each trajectory has a fixed length and can therefore appear at only one specific time, we can drop the i subscripts.

Finally, we need a target distribution. In general, any arbitrary target distribution can be used. There are a variety of ways one might imagine encoding a distribution. In this work we will focus on the case where a distribution has been specified and may be queried.

We can now define an algorithm to compute a policy $P(a|t)$ for every partial trajectory in $\mathcal{T}$.

1: Build a tree of all possible trajectories.
2: Initialize each leaf node (complete trajectory) with its target probability $P(t)$.
   In reverse topological order:
3: **for** Every t **do**
4: **for** Every child t'_i of trajectory t **do**
5: Condition Equation 2 on t:

$$P(t'_i|t) = \sum_{\forall a \in \mathcal{A}_t} (P(t'_i|a, t) \cdot P(a|t))$$

6: **end for**
7: This forms a system of $|\mathcal{T}_{t'_i}|$ linear equations in $|\mathcal{A}_t|$ unknowns:

$$\vec{P}(t'_i|t) = \vec{P}(t'_i|a, t) \cdot \vec{\pi}(t)$$

    which can be solved for π using standard linear algebra.[2]
8: Substitute π into Equation 2 and determine if there is at least one non-zero product of a row and column.
9: **if** There is no non-zero product **then**
10: $P(t) = 0$
11: **else**
12: $P(t) = \sum_i P(t'_i)$
13: **end if**
14: **end for**

[1] As we shall see, the inclusion of actions in the trajectory allows us to ensure that our TTD-MDP is not underconstrained.

[2] In order to have a well specified linear system it is necessary for the number of trajectories reachable from the current trajectory to be at least as large as the number of available actions. If this is not the case, the action taken can be included in the representation of the trajectory so the same state-based trajectory will appear to be different depending on the action. This procedure ensures at least as many trajectories as actions.

In steps (9)-(13), we identify when no non-zero product of rows of $\vec{P}(t'_i|a,t)$ and the column vector $\vec{\pi}$ exists. If there is no non-zero product, then there is no policy that will satisfy any percentage of the goal probability $P(t'_i)$ and we set $P(t) = 0$.

Practical Issues

Impossible Constraints

When an exact solution is obtainable the algorithm will return an optimal policy; however, this is not always possible. There are two types of errors we can encounter. First, there may be no vector $\vec{\pi}(t)$ that satisfies the linear system exactly. Second, even when there is an exact solution, the elements of $\vec{\pi}(t)$ may not be probabilities (though they will still sum to 1.0).

Lemma 1. *For a given trajectory t with subsequent trajectories $t'_1, \ldots, t'_n$ and actions $a_1, \ldots, a_m$, the following condition is sufficient for there to be no exact solution to the system of equations where the entries of $\vec{\pi}(t)$ are probabilities:*

$$\exists i \text{ such that either}$$

$$\forall a : P(t'_i|a,t) < P(t'_i|t) \tag{3}$$

$$\text{or } \forall a : P(t'_i|a,t) > P(t'_i|t) \tag{4}$$

Proof. The proof follows from the observation that unless the probabilities of two actions bracket the desired distribution (or at least one matches exactly) there is no convex combination of actions that can result in the desired distribution. □

Corollary 2. *If the system has an exact solution, then for every t'_i, there exist a_j, a_k such that $P(t'_i|a_j,t) \leq P(t'_i|t)$ and $P(t'_i|a_k,t) \geq P(t'_i|t)$.*

For example, consider a trajectory t', three subsequent trajectories t_1, t_2, t_3 and three actions a_1, a_2, a_3 whose linear system is defined by:

$$\begin{bmatrix} 0.0 \\ 0.3333 \\ 0.6667 \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 & 0.0 \\ 0.0 & 0.5 & 0.5 \\ 0.5 & 0.0 & 0.5 \end{bmatrix} \cdot \vec{\pi}(t) \tag{5}$$

Then, the solution vector

$$\vec{\pi}(t) = \begin{bmatrix} 0.3333 \\ -0.3333 \\ 1.0000 \end{bmatrix}$$

does not represent a probability distribution.

While this solution is not a vector of probabilities, it does satisfy the linear system. Intuitively, achieving the desired distribution requires that action a_2 be "undone" some percentage of the time. Since it is impossible, in practice we zero out any negative values and renormalize. We have derived a lower bound on error when Lemma 1 holds for only one trajectory t_j: $\|\widehat{Y} - Y\|_1 = \|\vec{P}(t_i|a,t') \cdot \widehat{\pi}(t) - Y\|_1 \geq 2 \times [\min_a |P(t_j|a,t') - P(t_j)|]$ where $\widehat{Y}$ is the distribution of trajectories for a particular choice of policy $\widehat{\pi}(t)$ (*e.g* after the normalization to $\widehat{P}(a|t')$) and Y is the desired distribution. While we have not yet proven that our normalization procedure hits this lower bound, empirical analysis indicates that it always does.[3]

Intractable Problems

Our algorithm builds a trajectory tree. In practice, the trajectory tree may be infeasible to compute and store, or completely specifying a distribution over all trajectories may be difficult or impossible. Here, we introduce a method for solving TTD-MDPs in this case. The basic strategy is to sample some number of trajectories from $P(\mathcal{T})$ and build a smaller tree based on those.

1: Pick some $\mathcal{T}_s \subset \mathcal{T}$ to sample and construct a distribution. We chose the empirical distribution:

$$\widetilde{P}(T_s) = \begin{cases} 0.0 & \text{if } P(t) < \phi \\ \frac{P(t)}{C} & \text{otherwise} \end{cases}$$

by normalizing the desired probabilities of all sampled trajectories above some given threshold ϕ.

2: Build a trajectory tree using T_s and $\widetilde{P}(T_s)$

3: Solve the resulting problem using the algorithm from above, picking some recovery mechanism for action selection once a trajectory not sampled is encountered.

We assume that it is most important to reduce error on the high-probability trajectories. This method focuses on reducing overall error in the approximation by ensuring that the areas of the trajectory space that could potentially contribute the largest local error are solved as accurately as possible. Even if the evaluation sampling "falls out of" the set of sampled trajectories $\mathcal{T}_s$ (*e.g.* if non-determinism causes an undesirable action outcome), we still maintain several nice characteristics. If our deviation from the sampled trajectory tree is near the root, it is likely that most or all of the subsequent trajectories have low probability in $P(\mathcal{T})$ (and were therefore not included in $\widetilde{P}(\mathcal{T}_s)$). On the other hand, if an evaluation sample falls out of the set of sampled trajectories far down the tree, it is likely that it will result in trajectory with a high desired probability because that part of the trajectory space is more heavily represented in $\mathcal{T}_s$. In other words, the effects of sampling error are minimized when it is possible to do well and maximized in the cases where it would have been impossible to do well.

Results

To test the feasibility and accuracy of our algorithm, we ran experiments on a synthetic grid world and the drama-management MDP we previously studied.[4] The synthetic grid world is a square-grid MDP where there are at most two actions in every state; a start state $s_0 = (0,0)$ and goal state $s_g = (n,n)$; and a known transition model $P(s|a,s')$. The

[3] When Lemma 1 does not hold, we have found examples where the L_1 error is less than this bound.

[4] For more details on the drama management MDP, see Nelson *et al.* (2006) or Nelson & Mateas (2005).

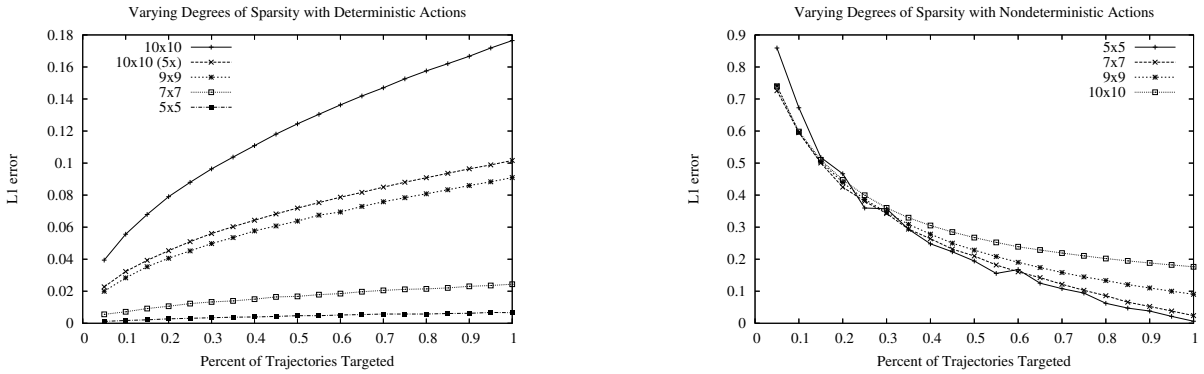

Figure 1: Percent Error evaluating with 1,000,000 samples as a function of target distribution sparsity with deterministic and non-deterministic actions.

two actions are "move right" and "move up" with the obvious meanings. Having only two actions prevents infinite-length trajectories.

We ran trials with and without deterministic actions. We also tested with various target distributions. For each trial, we selected a complete trajectory with some probability. Each selected trajectory was given uniform target probability and the remainder were given zero target probability. We varied the selection probability from 0.05 to 1.0. We then built the tree and computed a policy that was used to generate 1,000,000 samples. The relative frequency of each trajectory was compared against its target and error was reported as $\sum_{t \in \mathcal{T}} |P(t) - Emp(t)|$ (where $Emp(t)$ is the frequency of t during sampling evaluation). Lastly, we varied the size of the grid from 5×5 to 10×10. For each set of parameters, we ran 10 trials and averaged the results.

When all actions are deterministic, an exact policy can be easily computed. We plot the results of these trials in Figure 1 to illustrate the effect of sampling error. Not surprisingly, the L_1 error increases as either the size of the grid world increases or as the percentage of complete trajectories with non-zero target probability increases. Using 5,000,000 samples in the 10×10 grid world reduces this error by half.

A similar experimental setup was used to test the result of non-deterministic actions. Here, as the percentage of complete trajectories with non-zero target probability is increased, the error is reduced; however, as one would expect, absolute error is significantly higher (note the scale of the graphs in Figure 1). Intuitively, when only a few of the trajectories are desirable, any non-determinism in the outcome of actions may result in a trajectory with zero desired probability.

To test on a real-world application—one that is computationally intractable to solve exactly—we used a version of the *Anchorhead* story on which we have previously explored using RL (Nelson *et al.* 2006). In our model, *Anchorhead* has 29 plot points and 90 drama-manager actions. The evaluation function is a combination of features, such as the spatial locality of action, having plot points occur in an order that motivates later events, and so on; more details on the general framework and evaluation functions are given by Nelson & Mateas (2005).

To evaluate the effect of a drama manager, we run simulated stories and plot a distribution of story qualities. The drama manager's goal is to increase the frequency of highly-rated stories and decrease the frequency of low-rated stories. In particular, we would prefer that very low-rated stories never happen, while achieving a good distribution over highly-rated stories. We use the evaluation function and these preferences to define a TTD-MDP: Any story (trajectory of plot points) that evaluates below a threshold has a target frequency of 0, while those above the threshold should all appear, with the highly-rated ones more likely.

We build a tree from a set of sampled stories, which we can solve exactly. Unfortunately, during evaluation or actual gameplay, it is often the case that non-determinism in our player model causes us to encounter story trajectories not in the tree. We show results from two recovery strategies. In the first, the system always takes a domain-specific fallback action. In the second, online sampling search is used; that is, the system locally simulates possible stories from the current trajectory and uses the results to choose an action. In general, we find that what is best to do is domain specific. Although not shown, we also tried acting randomly to recover but that, unsurprisingly, proved to always be worse than doing nothing. It is a characteristic of TTD-MDPs in general that the path to the goal is what is important and therefore acting in an unprincipled manner can almost never be beneficial.

Figure 2 shows several story-quality distributions. In these experiments, our target distribution avoids any stories with a quality of less than 0.55, while skewing slightly towards highly-rated stories. The baseline shows the distribution of story qualities when no drama manager is used; SAS+ is Weyhrauch's sampling search; TTD:Null is the TTD-computed policy with all null actions taken when the system encounters an unknown trajectory; TTD:SAS+ is the hybrid sampling approach; and RL is an RL-trained policy.

Compared to no drama management, SAS+ increases the number of highly-rated stories, but also results in more very poorly-rated stories. TTD:Null results in nearly the opposite of what we intended, increasing the number of low-rated sto-

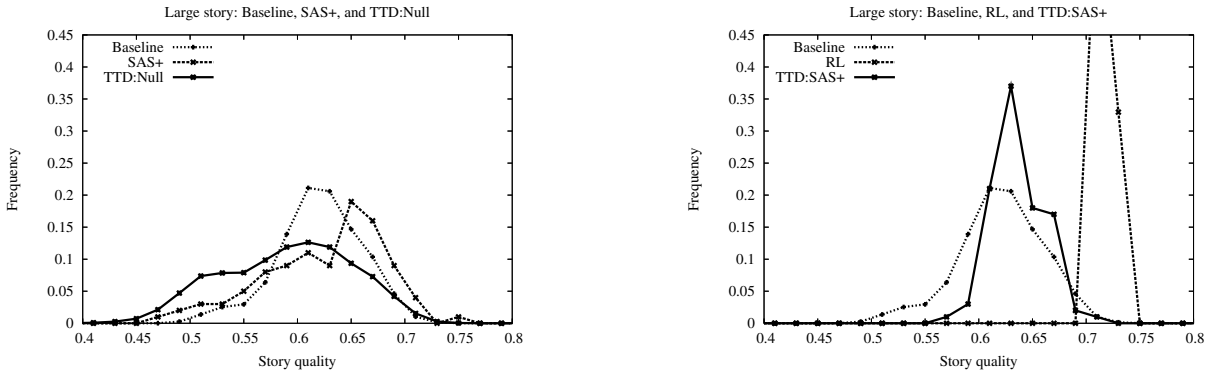

Figure 2: Graph of drama management results with and without TTD-MDPs as well as results based on threshold and/or with and without SAS+ or online policy adaption.

ries. The distribution resulting from RL is an impulse at the top of the story-quality range, illustrating that RL is solving quite a different problem, as it results in a handful of the same stories happening repeatedly. TTD using SAS+ as a fallback, however, yields results consistent with our target distribution: TTD:SAS+ has virtually no stories below the target cutoff of 0.55, and a nice distribution of stories above. It appears that the sampling used to populate the trajectory tree finds prefixes common to the stories above the threshold. Solving the TTD-MDP results in a policy that keeps the agents on these trajectories as long as possible; once the agent falls out of this space, it is not only already in a good part of the space, but the number of remaining trajectories has been reduced enough to make a sampling search feasible and effective.

To better understand the effects of approximating a solution to a TTD-MDP by sampling trajectories, we explored a smaller story with only 9 plot points and 28 DM actions. Figure 3 shows a similar improvement between using the null fallback action and sampling search; however, we also found that varying the number of samples used to build the trajectory tree has minimal effect on the resulting distribution. In this case, increasing from 10,000 to 100,000 samples did have an effect, but further increasing to 2,000,000 had no effect on performance. This suggests that this method may perform well on even larger problems.

Related Work

Although our formulation is novel, there has been some work that addresses some of the issues raised here. For example, one attempt to achieve a distribution over actions for reasons other than trading off exploration and exploitation came out of work with Cobot, an agent that interacted with humans in a socially-complex virtual environment (Isbell *et al.* 2001). There, state-action Q-values were used to induce a distribution over actions. This choice resulted in a distribution biased towards positive outcomes (as was Cobot's goal), but did not target any specific distribution, nor directly take into account the way in which locally stochastic action selection affected the global distribution of outcomes.

Kearns, Mansour, & Ng (2000) describe a trajectory-tree method in which they use a generative model of observations and transitions to construct trajectory trees in POMDPs. Their method uses a sampling approach where a number of sampled trajectory trees are used in aggregate to estimate the value of a particular policy. In our case, we construct one trajectory tree without a generative model and use it to to solve for a policy, rather than to estimate its value.

Littman (1994) uses Markov games as a framework for exploring reinforcement learning in the presence of multiple adaptive agents. Our formulation is similar in that it acknowledges a source of randomness outside the environment itself, finding a solution that requires a stochastic policy. On the other hand, this work was not directly concerned with distributions over outcomes, and assumed a zero-sum game. In the formulation of the drama-management problem, the game is not zero-sum: the player and the drama manager are, in effect, playing completely different games.

If we think of TTD-MDPs as looking at the dynamics of an MDP, but holding non-traditional elements fixed, then we find common ground with recent work in *inverse reinforcement learning* (Ng & Russell 2000). In this formulation, the goal is to observe the policy of an agent and then infer the reward function that led to its behavior. In the drama management case, non-determinism arises from the user but also from our actions. If we think of the desired distribution then the non-determinism that is not explained by the user model is exactly the distribution over our own actions.

Conclusions and Future Work

We have presented a new class of Markov decision problems where matching a target distribution replaces the goal of maximizing expected reward. Additionally, we have specified two approaches for solving TTD-MDPs.

We believe that the problem TTD-MDPs address—namely matching desired distributions over outcomes and trajectories—is a fertile area for both theoretical and practical treatment. In the short term we are interested in deriving tighter bounds on the error that can be achieved in TTD-MDPs, particularly under error measures other than L_1.

We also intend to explore the case where target distributions are not readily available. For example, authors may

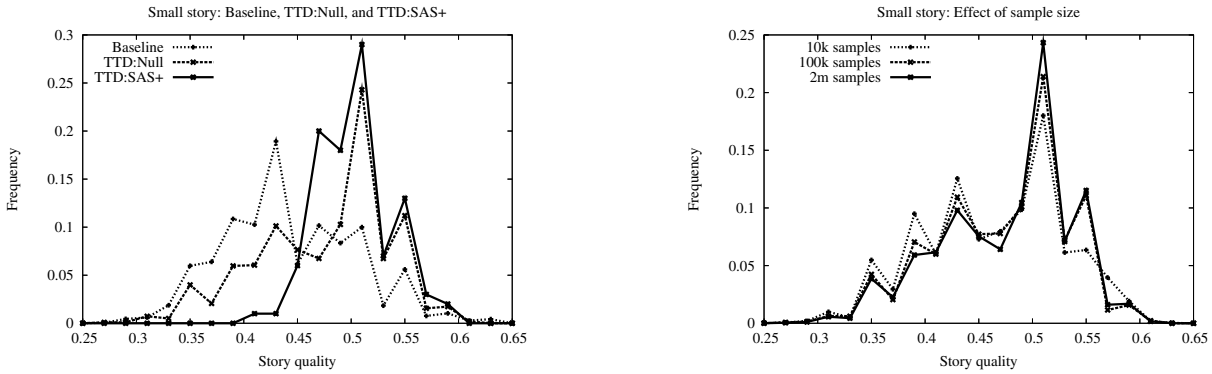

Figure 3: Graph of drama-management results on a small story, confirming the effects of SAS+ combined with TTD, and illustrating the effects of sample size on the quality of TTD policies.

find it difficult to write down a distribution, but are comfortable with specifying a story-evaluation function. Unfortunately, it is not clear that the evaluation function will lead to the author's desired distribution. The problem becomes one of also modifying the evaluation function so that it leads to the target distribution.

Additionally, we are interested in scaling TTD-MDPs to apply to even larger problems. One approach is to deal with sample error in the trajectory tree by dynamically adapting the tree through pruning existing subtrees and adding new subtrees as better estimates of the desired distribution become available. Another approach would use an online model-free method where the trajectory tree is never explicitly represented.

References

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

Isbell, Jr., C. L.; Shelton, C. R.; Kearns, M.; Singh, S.; and Stone, P. 2001. A social reinforcement learning agent. In *Proceedings of the Fifth International Conference on Autonomous Agents (Agents-01)*, 377–384.

Kearns, M.; Mansour, Y.; and Ng, A. Y. 2000. Approximate planning in large POMDPs via reusable trajectories. *Advances in Neural Information Processing Systems* 12.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML-94)*, 157–163.

Nelson, M. J., and Mateas, M. 2005. Search-based drama management in the interactive fiction Anchorhead. In *Proceedings of the First Annual Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-05)*.

Nelson, M. J.; Roberts, D. L.; Isbell, Jr., C. L.; and Mateas, M. 2006. Reinforcement learning for declarative optimization-based drama management. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-06)*.

Ng, A. Y., and Russell, S. 2000. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00)*, 663–670.

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