Implementation Cost and Efficiency for AI Experience Managers

David Thue[1,2] and **Vadim Bulitko**[1] and **Howard J. Hamilton**[2]

[1]Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8, Canada
dthue | bulitko @ ualberta.ca

[2]Department of Computer Science
University of Regina
Regina, SK, S4S 0A2, Canada
hamilton@cs.uregina.ca

Abstract

The study of Artificial Intelligence (AI) experience managers seeks to create software agents that can support compelling, interactive user experiences without needing any online guidance from human experts. Evaluating the utility of such AI managers is important in both academia and industry, both for measuring our progress in the field and for estimating a given manager's practical viability. While several methods have been studied that evaluate a manager's effectiveness, relatively few have explored the question of how costly a manager might be to implement in practice. We explore the latter question in this paper, presenting a formal way to estimate the cost of implementing an AI experience manager at scale.

1 Introduction

In an *interactive story*, one or more players are given the opportunity to act as characters in an unfolding narrative experience. Each experience occurs in the context of an *interactive environment*, where players alternately perceive states and perform actions as the experience proceeds. Through their actions, each player exercises *agency* – the ability to influence the course of their experience in an interactive environment by performing actions therein. Agency is generally considered to be beneficial in daily life (Larson 1989), but its presence in interactive stories makes it difficult for the designer of the story's environment to guarantee that any *particular* experience will be had. Instead, the designer provides content for a variety of possible experiences, each of which could occur as the result of different player actions.

To help ensure that each player will be affected as the designer intended given *any* valid sequence of their actions, an *experience manager* can alter the course of an interactive experience whenever multiple viable continuations exist (Riedl et al. 2008). When treated as an Artificial Intelligence (AI) agent, an experience manager works to maximize an objective function provided by the designer, toward ensuring that each player is affected by their experience in the way that the designer intended (Weyhrauch 1997). For example, the AI experience manager in *Façade* dynamically builds a sequence of dramatic interactions (called "beats") among two virtual characters and the player, and each beat is associated with the designer's estimate of how much tension it will add to the story (Mateas 2002; Mateas and Stern 2003). The designer provides the manager with a desired curve of tension versus time, and the manager works to maximize an objective function defined as one over the absolute value of the difference between two values of tension: the amount currently in the story (which it estimates online) and the amount that the designer specified for that time. The assumption underlying this design is that the more closely the estimated tension curve in each player's story matches the desired curve, the more likely the player will be affected by their experience as the designer intended (e.g., feeling catharsis). In general, after each player action, the state of the environment changes to a new state that offers one or more actions for the player to perform. Whenever more than one state can occur following a given player action (e.g., the first states of different beats in *Façade*), an experience manager can select the one that maximizes its objective function, and thereby try to ensure that each player's experience will affect them in the intended way.

To accelerate progress toward creating AI experience managers, it is important to build and critically evaluate complete, working prototypes (Mateas and Stern 2003). Unfortunately, evaluating such managers is a challenging problem (Rowe et al. 2009). Comparing the effectiveness of two managers through controlled play-testing can be time-consuming, and due to their complex nature, it can be difficult to attribute a manager's effectiveness to only one of its components (Billings and Kan 2006). Furthermore, even when a manager appears to be effective during testing, it can be unclear whether or not the cost of implementing that manager would be acceptable for a larger interactive environment. These concerns suggest two complementary families of evaluations: those which assess a manager's *effectiveness* (i.e., how successful it was in producing the result that was intended by its designer) and those which assess its *efficiency* (i.e., its effectiveness relative to the amounts of designer effort and computation that it requires). We refer readers to Rowe et al.'s (2009) work on the *StoryEval* framework for an overview of prior work on evaluating manager effectiveness, and note that recent research has explored this area as well (Thue et al. 2011; Ontañón and Zhu 2011; Ramirez and Bulitko 2012; Fendt et al. 2012).

Compared to the recent interest in evaluating manager effectiveness, evaluating manager efficiency has received less attention. Chen et al. (2009) proposed measuring "authorial leverage" as a way to evaluate different methods of implementing a given manager. By their definition, one method gives more authorial leverage than another when, for a given amount of designer effort, it allows the designer to (i) create a more complex policy for the manager, (ii) change an existing manager's policy in a more complex way, or (iii) provide players with a wider variety of experiences in the interactive environment. Although three types of authorial leverage were defined, recent research efforts have tended to focus on the third type (variability of experience) (Sullivan 2012; Li et al. 2013) without addressing the other two. As the variability of players' experiences increases, however, having ways to estimate the cost of creating or changing a manager's policy will become increasingly important, because the effort required to support a manager's operation is inherently tied to the so-called "authoring bottleneck" problem (Csinger, Booth, and Poole 1994).

In this paper, we present a method for estimating the cost of implementing or modifying the policy of an AI manager. We describe a framework for experience management as the foundation for our approach, and consider a related approach by Chen et al. We then present our cost estimation method, and summarize how we used it to compare two ways of implementing an existing manager's policy. We conclude by discussing the benefits and limitations of our approach, and offer ideas for future work.

2 Problem Formulation

Following Thue & Bulitko (2012), an interactive environment can be thought of as a graph where each node ($s \in S$) is a state of the environment and each edge ($a \in A$) is an action that the player can perform. The dynamics of the environment (i.e., how each action causes the environment to transition from one state to another) are governed by a *transition function* ($\tau : S \times A \to S$). We refer to an interactive environment generally as a "game" henceforth.

2.1 Experience Management

We adopt Thue & Bulitko's (2012) definition of *experience management* as a process that changes a game's transition function during gameplay; doing so allows us to treat experience managers that change the game's state (e.g., some drama managers (Weyhrauch 1997; Mateas and Stern 2003; Nelson et al. 2006)) and those that change the game's rules (e.g., dynamic difficulty adjustment (Hunicke and Chapman 2004)) in a unified way. However, instead of using Thue & Bulitko's definition of an experience manager's objective (which was to maximize the long-term sum of the player's rewards derived from playing), we borrow from Weyhrauch's (1997) and Nelson et al.'s (2006) work on optimization-based drama management to define it more generally: an experience manager's *objective* is to maximize a designer-provided objective function.

We assume that the game's designer wishes for each player to be affected by their experience in some particular way. To achieve this goal using an experience manager, the designer specifies three key components. First, they specify a set of state/action pairs ($D \subseteq S \times A$), where each pair indicates a point in the experience at which the manager should consider changing the transition function. We refer to each such pair as a *decision point*. Second, they specify a set of transition functions ($\mathcal{T}$) whose elements are all designer-approved alternatives to the game's current transition function. Whenever a decision point is reached, the manager will change the game's transition function to one that it chooses from $\mathcal{T}$. Third, they specify an *objective function* (ϕ), which maps from a decision point $\langle s, a \rangle$ and a transition function τ to a real number ($\phi : D \times \mathcal{T} \to \mathbb{R}$). Generally speaking, this function estimates the likelihood of the current player being affected by their experience in the designer's intended way, if the manager were to change the game's transition function to τ after point $\langle s, a \rangle$ occurred during gameplay.

Given these components, a manager's *policy* is a function χ which, given a decision point $\langle s, a \rangle$ and a set of transition functions $\mathcal{T}$, selects a transition function from the given set in a way that maximizes its objective function (Equation 1).

$$\chi(\langle s, a \rangle, \mathcal{T}) = \arg\max_{\tau \in \mathcal{T}} \phi(\langle s, a \rangle, \tau) \qquad (1)$$

Although not required for a general experience manager, prior specifications of objective functions for experience managers have tended to depend on a particular extra component. Specifically, a vector of n *features* ($\boldsymbol{f}$) is used map any alternating sequence of states and actions that can occur in an experience (i.e., a player *history*, $h \in H$) to a vector of n real numbers ($\boldsymbol{f} : H \to \mathbb{R}^n$). By changing the total number of features or the range of each feature, the designer can set the amount of abstraction with which each feature vector represents its associated state/action history.

2.2 Implementation Cost

At a high level, our goal in this paper is to evaluate the efficiency of a given manager in terms of the cost of implementing its policy. Since specifying the sets of decision points and transition functions (D and $\mathcal{T}$) in Equation 1 is part of creating the manager's *domain* (i.e., the inputs to the manager's policy), we focus on the cost of specifying the objective function (ϕ) henceforth. Furthermore, whenever two objective functions are such that the manager's policy would be the same regardless of which function it used, we say that both functions result in the same manager policy.

In general, the cost of specifying an objective function might depend on the parameters of its components (e.g., the number of features in the feature vector, n). For example, a given method of specifying an objective function might require 10 units of designer effort per feature for each state in the environment, resulting in a cost of $10n$ units. Since some parameters might be more important than others (to the designer) in terms of how changing them affects implementation cost, we seek a solution that allows the designer to focus their analysis on a particular subset of its parameters.

The problem that we aim to solve is as follows. Given (a) an experience manager's policy, (b) two or more methods for specifying objective functions that result in that policy, and (c) a subset of the functions' parameters to analyze, characterize the relative efficiency of the given methods in terms of the indicated parameters of interest. For example, suppose that we have two methods (M1 and M2) for specifying objective functions that both result in a given manager policy. Suppose further that both methods use a feature vector as one of their components, and that we wish to analyze their efficiency with respect to increasing the number of features that are used (n). Our goal would then be to determine whether M1 is more efficient than M2 (in terms of designer effort) over the different possible values of n.

3 Related Work

Chen et al. (2009) measured the efficiency of Declarative Optimization-based Drama Management (DODM) using an empirical approach. Specifically, they used DODM to manage the experiences of a group of players in an interactive story, and then induced a decision tree from this data to mimic the DODM manager's policy. Because the tree contained a large number of nodes (70), they deemed it to be too complex to implement by hand, and thus concluded that DODM provides authorial leverage. This claim implies that the DODM objective function in Chen et al.'s testbed can be specified with less designer effort than a decision tree with 70 nodes, but no direct comparisons were made. Furthermore, there seems to be no way to focus Chen et al.'s approach on particular parameters of a manager's objective function (as we wish to do; recall Section 2.2). The approach that we present in this paper overcomes these limitations by relying more on theory than on empirical analysis.

4 Proposed Approach

To characterize the efficiency of a method for specifying a manager's objective function in a way that promotes comparisons and focused analyses, we developed a formal approach based on the theory of run-time analysis. Given an algorithm to perform a certain task, run-time analysis characterizes its efficiency by estimating its computational cost. Similarly, given a way to specify a manager's objective function, we characterize its efficiency by estimating its implementation cost. Furthermore, just as computational cost is estimated as a function of the given task's parameters (e.g., the size of its input), we estimate implementation cost as a function of the parameters of a given specification method (e.g., the total number of features, n).

Our approach proceeds through five steps. First, for each given method of specifying an objective function, we explicitly describe the required steps of the method in terms of the sets and functions that we presented in Section 2.1. Second, we treat the steps as an algorithm, and use standard run-time analysis techniques to derive a cost function for each method; while doing so, we introduce any constants that are needed to represent particular amounts of designer effort. Continuing our earlier example (Section 2.2), method M1 might require e_1 units of designer effort per feature for each state in the environment (i.e., $\textit{cost}_{\text{M1}}(n, S) = e_1 n|S|$), while M2 might require e_2 units of effort for every *pair* of features, for each state (i.e., $\textit{cost}_{\text{M2}}(n, S) = e_2 n^2|S|$). Third, given a set of parameters of interest (e.g., the number of features, n), we classify each function according to its asymptotic growth rate over each parameter in the set. In the example, the cost of M1 increases linearly with n, and the cost of M2 increases quadratically with n. Fourth, for each parameter of interest, we characterize the relative efficiency of each specification method as a piecewise function of that parameter (e.g., for small values of n, M2 is more efficient than M1, but for larger values of n, M1 is more efficient). As an optional fifth step, when the values of the designer effort constants (i.e., e_1, e_2, etc.) are known, we calculate the intersection points between the cost functions of the given methods and use them to specify the ranges of our piecewise characterization more accurately. In our example, knowing that $e_1 = 10$ and $e_2 = 3$ is enough to learn that M2 is more efficient when $n \leq 3$ and M1 is more efficient when $n > 3$, because the cost functions intersect when $n \approx 3.2$.

5 Results, Discussion & Future Work

We used our approach to analyze two different methods of implementing the policy of PaSSAGE, an AI experience manager by Thue et al. (2007). Our goal with this analysis was to characterize the relative efficiencies of the methods (in terms of designer effort) with respect to the number of features that they use. Although lack of space precludes the full presentation of our results, we summarize them briefly here. We found that the cost of using Thue et al.'s original implementation method for PaSSAGE (which uses features to represent both a player model and annotations on story content quality) increases linearly with the number of features used. Meanwhile, if PaSSAGE's policy were implemented using a decision tree at each decision point (in a similar spirit as Chen et al.'s "decision tree equivalents" (2009)), the implementation cost would increase exponentially with the number of features used. Given known values from PaSSAGE's testbed domain (e.g., the total number of decision points, $|D|$) and estimates for the relative sizes of the relevant designer effort constants, we found that Thue et al.'s method appears to be more efficient than the decision tree approach for all positive values of n.

Our approach characterizes the relative efficiencies of different methods for implementing a manager's policy, and supports the comparison of competing methods with respect to particular parameters of interest. Although it can be difficult to estimate the constants in a given cost function when analyzing managers in retrospect, we suspect that fairly accurate estimates could be obtained for new managers if designer designer effort was tracked during development.

In the future, testing our approach on managers that use different components in their objective functions (e.g., a prediction function as in Weyhrauch's work (1997)) will be an important next step. Applying our approach to other methods of specifying objective functions (e.g., Roberts et al.'s TTD-MDPs (2006) or Nelson et al.'s Reinforcement Learning approach (2006b)) might also be fruitful.

Conclusion

In this paper, we presented a method for characterizing the efficiency of a manager's policy with respect to the amount of designer effort that it requires to implement. By drawing on the theory of run-time analysis, our method supports direct comparisons between competing methods and also allows designers to analyze how particular parameters of a policy's objective function can influence its implementation.

References

Billings, D., and Kan, M. 2006. A tool for the direct assessment of poker decisions. *ICGA Journal* 29(3):119–142.

Chen, S.; Nelson, M. J.; and Mateas, M. 2009. Evaluating the authorial leverage of drama management. In *The Fifth Artificial Intelligence and Interactive Digital Entertainment Conf. (AIIDE)*, 136–141. Menlo Park, CA: AAAI Press.

Csinger, A.; Booth, K. S.; and Poole, D. 1994. AI meets authoring: User models for intelligent multimedia. *Artificial Intelligence Review* 8(5):447–468.

Fendt, M. W.; Harrison, B. E.; Ware, S. G.; Cardona-Rivera, R.; and Roberts, D. L. 2012. Achieving the illusion of agency. In *The Fifth International Conf. on Interactive Digital Storytelling (ICIDS)*, 114–125. Springer Verlag.

Hunicke, R., and Chapman, V. 2004. AI for dynamic difficult adjustment in games. In *Challenges in Game AI Workshop, Nineteenth National Conf. on Artificial Intelligence*.

Larson, R. 1989. Is feeling "in control" related to happiness in daily life? *Psychological Reports* 64(3 Pt 1):75–84.

Li, B.; Lee-Urban, S.; Johnston, G.; and Riedl, M. O. 2013. Story generation with crowdsourced plot graphs. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, 598–604. Bellevue, WA: AAAI Press.

Mateas, M., and Stern, A. 2003. Façade: An experiment in building a fully-realized interactive drama. *Game Developers Conference (GDC 03).* San Jose, California.

Mateas, M. 2002. *Interactive Drama, Art, and Artificial Intelligence*. Ph.D. Dissertation, School of Computer Science, Computer Science Department, Carnegie Mellon University.

Nelson, M. J.; Mateas, M.; Roberts, D. L.; and Isbell, C. L. 2006. Declarative optimization-based drama management in interactive fiction. *IEEE Computer Graphics and Applications* 26(3):33–41.

Nelson, M. J.; Roberts, D. L.; Isbell, Jr., C. L.; and Mateas, M. 2006b. Reinforcement learning for declarative optimization-based drama management. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, 775–782. New York, NY, USA: ACM.

Ontañón, S., and Zhu, J. 2011. On the role of domain knowledge in analogy-based story generation. In *Proceedings of the Twenty-Second International Joint Conferences on Articial Intelligence (IJCAI)*, 1717–1722.

Ramirez, A. J., and Bulitko, V. 2012. Telling interactive player-specific stories and planning for it: ASD + PaSSAGE = PAST. In *The Eighth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, 173–178. Menlo Park, CA: AAAI Press.

Riedl, M. O.; Stern, A.; Dini, D.; and Alderman, J. 2008. Dynamic experience management in virtual worlds for entertainment, education, and training. In Tianfield, H., ed., *International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning*, volume 3, 23–42. Glasgow: SWIN Press.

Roberts, D. L.; Nelson, M. J.; Isbell, C. L.; Mateas, M.; and Littman, M. L. 2006. Targeting specific distributions of trajectories in MDPs. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI)*, 1213–1218. Menlo Park, CA: AAAI Press.

Rowe, J. P.; McQuiggan, S. W.; Robison, J. L.; Marcey, D. R.; and Lester, J. C. 2009. StoryEval: An empirical evaluation framework for narrative generation. In *AAAI Spring Symposium on Intelligent Narrative Technologies II*, 103–110. Menlo Park, CA: AAAI Press.

Sullivan, A. M. 2012. *The Grail Framework: Making Stories Playable on Three Levels in CRPGs*. Ph.D. Dissertation, UC Santa Cruz: Computer Science.

Thue, D., and Bulitko, V. 2012. Procedural game adaptation: Framing experience management as changing an mdp. In *Proceedings of the 5th Workshop in Intelligent Narrative Technologies*. Menlo Park, CA: AAAI Press.

Thue, D.; Bulitko, V.; Spetch, M.; and Wasylishen, E. 2007. Interactive storytelling: A player modelling approach. In *3rd Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, 43–48. Menlo Park, CA: AAAI Press.

Thue, D.; Bulitko, V.; Spetch, M.; and Romanuik, T. 2011. A computational model of perceived agency in video games. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE). Stanford, CA, USA.*, 91–96. Menlo Park, CA: AAAI Press.

Weyhrauch, P. 1997. *Guiding Interactive Drama*. Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.