An Automated Model-Based Adaptive Architecture in Modern Games

Chek Tien Tan
Department of Computer Science
DigiPen Institute of Technology, Singapore
10 Central Exchange Green, #01-01, Singapore 138649
chtan@digipen.edu

Ho-lun Cheng
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
hcheng@comp.nus.edu.sg

Abstract

This paper proposes an automatic model-based approach that enables adaptive decision making in modern virtual games. It builds upon the Integrated MDP and POMDP Learning AgeNT (IMPLANT) architecture (Tan and Cheng 2009) which has shown to provide plausible adaptive decision making in modern games. However, it suffers from highly time-consuming manual model specification problems. By incorporating an automated priority sweeping based model builder for the MDP, as well as using the Tactical Agent Personality (Tan and Cheng 2007) for the POMDP, the work in this paper aims to resolve these problems. Empirical proof of concept is shown based on an implementation in a modern game scenario, whereby the enhanced IMPLANT agent is shown to exhibit superior adaptation performance over the old IMPLANT agent whilst eliminating manual model specifications and at the same time still maintaining plausible speeds.

Introduction and Related Work

Decision making in modern game agents is a hard problem as modern game environments are well-known to contain a high level of uncertainty as well as limited observability. As this depicts a different problem domain from that of classic games, traditional game theoretic methods in artificial intelligence (AI) (Hsu 2004; Schaeffer et al. 2005; Buro 1997) cannot be directly applied to modern games. Partially Observable Markov Decision Processes (POMDPs) (Kaelbling, Littman, and Cassandra 1998) represent the state of the art in decision theory that naturally models a game agent acting in an uncertain environment with unobservable attributes. Unfortunately, POMDPs solution algorithms are notoriously computationally intractable (Burago, Rougemont, and Slissenko 1996), especially so in huge modern game worlds.

In a previous work by Tan and Cheng (2009), they have already devised an architecture called the IMPLANT which is based on a POMDP formulation. It tackles the intractability problem by exploiting the fact that the partially observable component of a virtual game world can be reduced into solely representing the player model. The argument is that a virtual game is entirely artificially crafted (fully observable) except for the human player that is unpredictable (partially observable). Hence the decision making problem is decomposed into MDP and POMDP abstract constituents respectively, with the bulk of the game environment represented in the tractable MDP. Although they have shown good performance in terms of both adaptation and speed in a limited domain, there are two limitations in their work that cripples generic applicability in modern games.

The first limitation is in their implementation in a simplified tennis game, whereby the game dynamics was completely handcrafted. Basically this means that the transition function in the MDP model was entirely manually specified. Perhaps in a highly simplified small game this is possible but the assumption was that the game environment model can be readily obtainable in general, which is impractical. In the context of a more complex game like a role-playing game (RPG) or a real-time strategy (RTS) game, this task would seem close to impossible as the number of states and probabilistic transitions are significantly larger.

The second limitation is that the player model represented by the POMDP was also handcrafted. Similarly in their simplified tennis game, the player archetypes was modeled according to common play styles extracted from a real-life book on tennis (Matsuzaki 2004). This kind of handcrafting is similar to other current work whereby player adaptation architectures are highly dependant on expert knowledge to specify the player archetypes (van der Sterren 2006; Sharma et al. 2007; Thue et al. 2007; Barber and Kudenko 2007; El-Nasr 2007). Other than the manual labor required in this process, the adaptation performance would nevertheless be also dependant on the competency of the experts' knowledge. Moreover, real-life tennis might be different from the player types in a video game.

In general, although IMPLANT has shown to be a plausible in a simple setting, there exists applicability limitations in typical modern games, due to the assumptions that the MDP and POMDP models are readily obtainable. Additionally, it is questionable to assume that the experts' knowledge in handcrafting the models were qualified enough to produce accurate adaptation. This paper aims to addresses these issues by automating both the processes via

1. using a priority sweeping reinforcement learning agent to automatically learn the MDP model, and
2. using the TAP model (Tan and Cheng 2007) to automati-

Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

cally specify the POMDP model.

In the remaining sections of this paper, essential background of the IMPLANT architecture is first given. Then the automation of both the MDP and POMDP specifications are described respectively. Next, the experimental setup is depicted along with the experimental results as empirical proof of concept. Finally a discussion is made on the results and the paper is concluded along with the plans for future work.

IMPLANT Architecture

The overall architecture of IMPLANT (Integrated MDP and POMDP Learning AgeNT) is as shown in Figure 1. The central idea is to decompose the game world into an MDP and a POMDP (Kaelbling, Littman, and Cassandra 1998) abstract with their respective policies computed. Current states and observations are individually obtained from each abstraction and processed via an Action Integrator (AI) function, which computes a resultant optimal action that the agent performs and affects the underlying game world, which then provides a reward signal that reinforces the action.

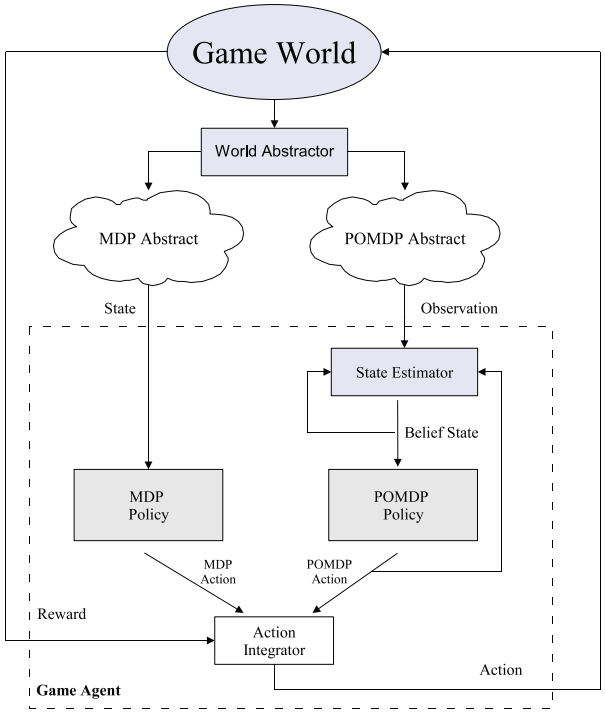

Figure 1: An overview of the IMPLANT architecture. The agent maps a separate MDP and POMDP abstract from the game world and finds optimal policies according to each abstract. For each game state and observation, the agent then obtains a single action via the Action Integrator from the two policies. The action is then reinforced by the reward afterwards.

Given any game world, it is manually modeled into an MDP abstract, M_{co}, and a POMDP abstract, M_{po}:

$$M_{co} = \langle \hat{\mathcal{S}}_{co}, \hat{\mathcal{A}}_{co}, \hat{T}_{co}, \hat{R}_{co}, \gamma \rangle, and \quad (1)$$
$$M_{po} = \langle \hat{\mathcal{S}}_{po}, \hat{\mathcal{A}}_{po}, \hat{T}_{po}, \hat{R}_{po}, \gamma, \hat{\mathcal{O}}, \hat{O}, b_0 \rangle,$$

where the subscripts $_{co}$ and $_{po}$ label the MDP and POMDP parameters respectively.

With the MDP and POMDP abstracts defined, they are solved to produce the optimal policies π_{co} and π_{po} respectively. With the policies generated, the agent decides the current resultant action a to take, computed via the Action Integrator which consists of a two-stage process. The first stage obtains a combined abstract action $\hat{a}$ whereby:

$$\hat{a} = \begin{cases} \hat{a}_{co} \bigcap \hat{a}_{po} & \text{if } \hat{a}_{co} \bigcap \hat{a}_{po} \neq \emptyset, \\ \hat{a}_{co} \bigcup \hat{a}_{po} & \text{else.} \end{cases} \quad (2)$$

This process aims to choose a set of actions optimal to both the POMDP abstract (player model) and the MDP abstract (game environment). This set will either be the intersection between $\hat{a}_{co}$ and $\hat{a}_{po}$, or the union between them when the intersection set is empty, in which the agent falls back to choose any action that is optimal to either the POMDP or MDP abstract.

The second stage in the AI function defines a further action selection process which determines a single most optimal action $a \in \hat{a}$ based on the in-game situation. Since the offline policy already defines an action set optimal to both the POMDP and MDP abstracts, only a minimal amount of learning needs to be performed online. A simple and efficient learning algorithm is adapted from a previous work by Tan and Cheng (2008a), whereby each action a is assigned a weight ω_a. Action selection then follows a standard ϵ-greedy algorithm (Sutton and Barto 1998) and weight updates are performed via the function $\omega_a = \omega_a + \alpha(\gamma_\tau - \overline{\gamma_\tau})$, where γ_τ is a variable reward signal generated at time τ, $\overline{r_T}$ is a reference point to determine the relative size of γ_τ, and α is a positive step-size parameter to control the magnitude of change.

The IMPLANT architecture has shown to perform better than naive implementations of reinforcement Q-learning (Sutton and Barto 1998) as well as a pure POMDP AI agent in a simplified tennis game (Tan and Cheng 2009).

Automatic MDP Specification

As mentioned, the first limitation in the IMPLANT architecture was that the model of the game environment (the transition function in particular) was presumably easy to obtain. In the previous tennis implementation (Tan and Cheng 2009), it was relatively straightforward to hand-code all the transition probabilities as the mechanics of tennis gameplay is simple and well-defined. The same cannot be said of game genres with complicated game mechanics (like an RPG) and a manual specification of the probabilities would be close to impossible. To address the issue of practicality in modern games, this problem cannot be overlooked. Hence an automated approach to transition function specification would be advantageous as a pre-processing stage.

Automatically learning a model of the environment is not a new problem and mature solutions are available in the research literature of the reinforcement learning domain. The specific method to be employed is a procedure adapted from prioritized sweeping. Prioritized sweeping is a model-based reinforcement learning method that is shown to converge fast. It ensures that the game mechanics (transition probabilities and rewards) are populated in the most efficient manner. Performing prioritized model updates in such a way targets areas where there are large changes in current estimates first. The model building process is as shown in Algorithm 1. This acts as a pre-processing step for the IMPLANT agent.

Input: set of all states $\mathcal{S}$ and set of all actions $\mathcal{A}$
Output: the game environment model defined by the transition function T and reward function R
initialize PQ, T, R, $resultantStateCount$ and $totalCount$;
while *simulation is running* **do**
  get current state $s \in \mathcal{S}$;
  get least tried action a;
  set $totalCount(s,a)$=$totalCount(s,a)+1$;
  execute a and record resultant state s' and reward r;
  set $resultantStateCount(s,a,s')$= $resultantStateCount(s,a,s')+1$;
  set $T(s,a,s')$=$\frac{resultantStateCount(s,a,s')}{totalCount(s,a)}$;
  set $R(s,a)$=r;
  insert s into top of PQ;
  while *there is computation time left* **do**
    get s from top of PQ;
    store the old value V^{old}=$V(s)$;
    update the new value $V(s)$ = $\max_{a\in\mathcal{A}}(R(s,a)+\gamma\sum_{s'\in\mathcal{S}}T(s,a,s')V(s'))$;
    compute the value change ΔV=$V^{old}-V(s)$;
    set $priority_s$ = 0;
    foreach *predecessor state* $\tilde{s}$ *and action* $\tilde{a}$ *leading into* s **do**
      set $priority_{\tilde{s}}$ = $\max(\Delta V \times T(\tilde{s},\tilde{a},s'), priority_{\tilde{s}})$;
      insert $\tilde{s}$ into PQ according to $priority_{\tilde{s}}$;
    end
  end
end

Algorithm 1: The IMPLANT model building algorithm. A procedure based on the prioritized sweeping method in reinforcement learning. As a pre-processing stage, the algorithm is encoded in a model building agent which iteratively populates the transition probabilities and rewards in a prioritized manner.

Automatic POMDP Model specification

The other limitation in the IMPLANT architecture was that it was not known how the POMDP player model can be obtained in general. In this paper we propose to use the Tactical Agent Personality (TAP) representation formulated by Tan and Cheng (2007). A description of the TAP model is given next, followed by a justification of why the TAP is well-suited for this purpose.

Tactical Agent Personality

Generally, the TAP model can be described as a statistical record of agent actions. The TAP model directly takes the available agent actions as the player archetypes, which can be directly specified without any human intervention.

For any agent in the game (PC or NPC), its TAP consists of a set of actions each tagged with a relative weight value (between 0 and 1) as shown in Figure 2. Formally defined, if $\mathcal{P}$ is the set of all agent personalities, the agent personality, $P_k \in \mathcal{P}$, of an agent k is a function that assigns a value to each action.

$$P_k : A \rightarrow [0,1], \tag{3}$$

where A is the set of all allowable actions.

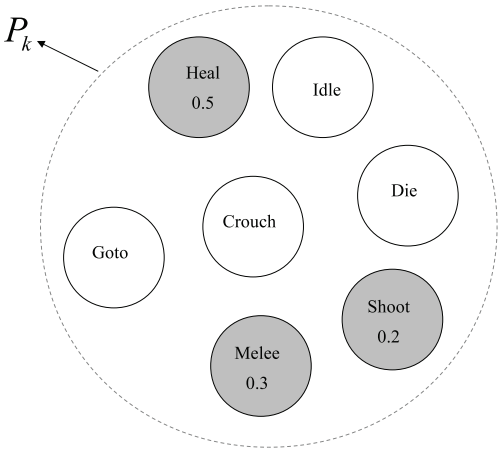

Figure 2: An example Tactical Agent Personality (TAP). Each action is tagged with a relative weight value that can be evaluated into a probability of choosing it.

Although the concept of TAP is simple, it is shown to be a powerful asset when used in an adaptation framework whereby a game agent's actions is determined automatically via a classifier that receives ally agent TAPs as input. It also models the natural phenomenon that a human's personality is directly reflected in their actions. This is shown to work plausibly in various modern game implementations (Tan and Cheng 2007; 2008b; 2008a; 2008c). Moreover, the simplicity of the TAP representation provides a great ease of application which should entice game practitioners to use it in actual commercial games.

The TAP Belief State

To see why the TAP model fits the requirement of the POMDP belief state in IMPLANT, we look at the definitions of the belief state and the TAP.

Definition A POMDP belief state represents the sufficient statistical information that an agent needs in order to act optimally on its next action.

Definition A TAP model of Agent A represents the sufficient statistical information that an Agent B needs in order to act optimally towards the Agent A.

In the IMPLANT architecture, the POMDP is purely responsible for player-specific adaptation, thus the belief state actually needs to only capture information needed to represent the player's behavior. This is exactly what the TAP does. Hence it can be seen that the TAP can actually be used to represent the belief state in the POMDP component of the IMPLANT architecture.

The POMDP component of the IMPLANT architecture is represented by $M_{po} = \langle \hat{\mathcal{S}}_{po}, \hat{\mathcal{A}}_{po}, \hat{T}_{po}, \hat{R}_{po}, \gamma, \hat{\mathcal{O}}, \hat{O}, b_0 \rangle$. The task now is to define the set of states $\hat{\mathcal{S}}_{po}$. Specifically, the action nodes a in the player's TAP P now defines the set of POMDP states $\hat{\mathcal{S}}_{po}$:

$$\hat{\mathcal{S}}_{po} = \{a | \forall a \in P\}.$$

This means that the belief state now captures sufficient information to represent the behavior of the player such that adaptive actions can be performed by the IMPLANT agent. This is verified by the extensive evaluations that have been performed on the TAP model (Tan and Cheng 2007; 2008b; 2008a; 2008c). This claim will be further verified empirically in an implementation of the enhanced IMPLANT model in the next section on evaluation.

By utilizing the TAP representation to define the states of the POMDP component, it also means that a weak bound is imposed on the number of states in the POMDP. This is crucial as tractability is the main hindrance to practical applications of POMDPs. Hence now, the POMDP states in the IMPLANT architecture is bounded by the equation

$$|\hat{\mathcal{S}}_{po}| \leq |\mathcal{A}|.$$

In other words, the number of POMDP states is at most the number of actions in the ground MDP of the game environment. Although there is no strict rule on the number of actions an agent can have in a modern game, this size will most likely not be very large as it defines the number of primitive actions an agent can perform. For example in a fairly complex game like *World of Warcraft*, the maximum number of actions available to an agent is around 20, which actually falls within the higher range of modern games with a large number of actions.

Evaluation

The improved IMPLANT architecture is implemented in a typical RPG boss battle scenario with a screenshot as shown in Figure 3. The results show that the TAP model is essential in providing better adaptation performance, and that the reinforcement learning model builder works in reasonable time. The details are described in the subsections that follow.

Experimental Methodology

The experiments in this section aims to achieve two goals, namely to test whether

Figure 3: A screenshot of the TAP-enhanced IMPLANT RPG agent (the robot at the bottom right) in the RPG experimental setups. The agent at the bottom left is the player, the agent at the top left is the boss agent and the agent at the top right is its enemy helper agent.

1. the inclusion of the TAP model into the IMPLANT architecture truly improves the effectiveness within efficient in-game speeds, and
2. whether the priority sweeping model builder builds the MDP in plausible offline speeds.

An auxiliary goal is to prove the feasibility of the IMPLANT architecture's generality in a different game genre - a more complicated RPG game as opposed to the previous simplified sports game (Tan and Cheng 2009). In general, we are testing for effectiveness and efficiency of the improved IMPLANT architecture. The sections that follow describe the game setups that aim to fulfil these test goals.

Game Mechanics

The game is set as a boss battle in an RPG game, commonly encountered in climactic moments or locations in the game. This battle basically involves the player (and the ally agent) facing against a superior enemy agent called the *boss agent*. The boss agent is generally far stronger than any enemy agents the player has encountered in that section of the game world. The boss agent also has a helper enemy agent. Hence it is a battle between the team consisting of player agent and IMPLANT agent versus the enemy team consisting of the boss agent and the enemy helper agent.

Basically the states indicate the targets, health points (HP) and status of each agent. The TAP model is represented by a single player model attribute which is instantiated by the actions available to the player as defined by the TAP framework. The game agents possess a wide variety of actions like *target*, *slash*, *cyclone*, *fireball*, *snowblast*, *heal*, *rage* and *evade*. The actions are differentiated in terms of range, state modifications, damage modifier probabilities and re-useable time. For example the *snowblast* action inflicts a 5hp damage to all agents within 3 tiles of range and performs double the damage with a probability of 0.3. The full descriptions are omitted in this paper due to space limitations. The tran-

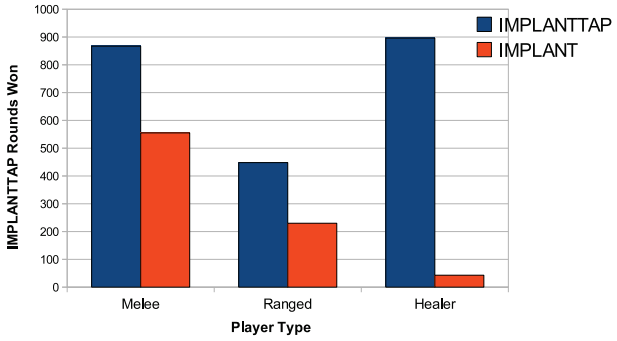

Figure 4: Effectiveness test results for the TAP-enhanced IMPLANT RPG Agent: histogram of number of rounds won by ally team. A plot of the number of rounds won by the ally team (the player agent and the AI agent). Higher means better performance. Each bar set represents a comparison between the TAP-enhanced IMPLANT RPG Agent (IMPLANTTAP) and the stripped-down IMPLANT agent (with the TAP component removed). Each set represents an experimental setup with a particular player model.

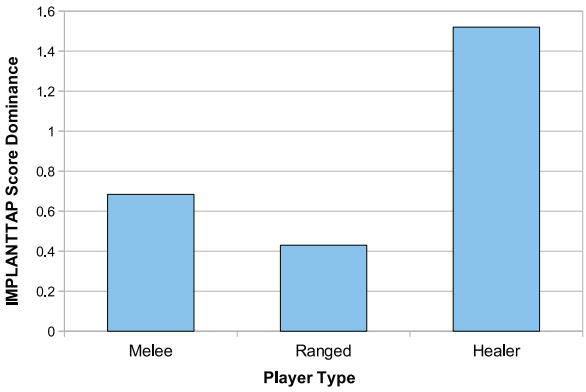

Figure 5: Effectiveness test results for the TAP-enhanced IMPLANT RPG Agent: Histogram of the score values. Higher means better performance. Each bar set represents a comparison between the TAP-enhanced IMPLANT RPG Agent (IMPLANTTAP) and the stripped-down IMPLANT agent (with the TAP component removed). Each set represents an experimental setup with a particular player model.

sitions and rewards are automatically specified by the prioritized sweeping model builder as described.

Experiments and Results

The experiments devised aim to test the effectiveness and efficiency criterions as described above.

Effectiveness Test In a single experimental set, two variations of the agent architecture are implemented. The first variation (IMPLANTTAP) is the improved IMPLANT architecture with the TAP model. The second variation (IMPLANT) is the IMPLANT architecture with the player model component removed. The aim is to show whether the TAP model is effective in improving the adaptation performance of the agent.

To achieve robustness, the game is ran for 1000 rounds for each player model and scores are plotted in Figures 4 and 5. To test the versatility of the improved architecture across different player types, each experimental set contains a different setup of the player, namely the *melee*, *ranged*, and *healer* models, based on the abstract actions in the TAP-defined POMDP model.

As seen in Figures 4 and 5, the IMPLANTTAP architecture generally outperforms the stripped-down IMPLANT architecture constantly throughout all the player models, both in the number of wins as well as the score. This shows that the inclusion of the player model component (TAP) is essential to improving the performance of the game agent. Consequently, it also means that adapting to the player is as important as adapting to the game environment.

To determine the significance of the results, unpaired two-sample one-tailed heteroscedastic Student's t-tests are performed on each pair of scores. The null hypothesis is that there is no different between the performance of the IMPLANTTAP and IMPLANT teams. The alternative hypothesis we want to achieve is that the IMPLANTTAP architecture performs better than the IMPLANT architecture. Hence a one-tailed test is used on each corresponding pair of scores. As no assumption can be made about the variances of each distribution, the heteroscedastic (unequal variance) t-test is used. The results of the t-test are as shown in Table 1. It can be seen that the p-values are all lower than 0.01 which implies that we can say with 99% confidence that our results are significant.

	Melee	Shoot	Heal
T-test p-value	< 0.010	< 0.010	< 0.010

Table 1: Effectiveness test results for the TAP-enhanced IMPLANT RPG Agent: Table of t-test p-values. These values verify the significance of the results that the IMPLANTTAP architecture is better than the IMPLANT architecture.

Efficiency Test The speed results are tabulated for both the IMPLANTTAP and IMPLANT architectures, as shown in Table 2, whereby it can be seen that the time required to query a decision in-game is still within a matter of milliseconds, which is unnoticeable by the human eye. This shows that the IMPLANT architecture is still very much viable in a much more complex setup like this.

The pre-computation time is shown for 5000 epoches in which 200 states are updated in each epoch. It appears that pre-computation time is rather lengthy due to the addition of the model building stage which takes quite a while to sweep through the huge game world. The pre-computation times are the same for both IMPLANTTAP and IMPLANT because it is dominated by the priority sweeping model building process. Nevertheless, this is still much better than taking weeks to design and hand-craft the model which is fur-

thermore prone to human errors. Currently, this seems like a reasonable tradeoff to make.

	Average Pre-computation Time	Average Query Time
IMPLANT	1h23m	0.002s
IMPLANTTAP	1h23m	0.011s

Table 2: Efficiency test results for the TAP-enhanced IMPLANT RPG Agent: Table of average adaptation speeds. The policy pre-computation and query times are tabulated for the TAP-enhanced IMPLANT (IMPLANTTAP) and stripped-down IMPLANT (with no TAP) setups. The pre-computation times are averaged over 20 runs. The query times are averaged over 100 runs. The pre-computation times are the same as the two setups use the same model builder. The query times are also both within a matter of milliseconds.

Discussion and Conclusions

This paper has described the enhancements of two important aspects of the IMPLANT architecture so as to improve its applicability across modern game genres. The first is the infusion of the TAP model into the architecture to actualize how the states in the POMDP component is defined. The second is the utilization of a automatic model building procedure to obtain the game mechanics needed for the rest of the MDP model in the IMPLANT architecture.

The implementation in a modern RPG game scenario further improves the credibility of the IMPLANT architecture in the applicability towards modern game genres. The experiments performed on the implementation has also shown that the TAP model is indeed crucial in improving adaptation performance, or more specifically adaptability towards the player. This chapter hence furnishes the essential details necessary to improve the generalizability of the IMPLANT architecture.

It can be seen however that an offline computation time of over an hour is still needed to build the MDP model, which somewhat slows down the development process. Nevertheless we believe that this is a reasonable tradeoff to make rather than tediously manually specifying the model. Future work can hence be targeted at improving this offline computation time. Alternative methods of reinforcement learning might also be evaluated to determined whether quicker times can be obtained.

References

Barber, H., and Kudenko, D. 2007. Dynamic generation of dilemma-based interactive narratives. In *Proceedings of The Artificial Intelligence and Interactive Digital Entertainment Conference*, 2–7.

Burago, D.; Rougemont, M. D.; and Slissenko, A. 1996. On the complexity of partially observed markov decision processes. *Theoretical Computer Science* 157:161–183.

Buro, M. 1997. Takeshi Murakami vs. Logistello. *International Computer-Chess Association Journal* 20(3):189–193.

El-Nasr, M. S. 2007. Interaction, narrative, and drama creating an adaptive interactive narrative using performance arts theories. *Interaction Studies* 8(2).

Hsu, F.-H. 2004. *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton, NJ, USA: Princeton University Press.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101:99–134.

Matsuzaki, C. 2004. *Tennis Fundamentals*. United States: Human Kinetics Publishers, first edition.

Schaeffer, J.; Björnsson, Y.; Burch, N.; Kishimoto, A.; 0003, M. M.; Lake, R.; Lu, P.; and Sutphen, S. 2005. Solving checkers. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 292–297.

Sharma, M.; Mehta, M.; Ontan, S.; and Ram, A. 2007. Player modeling evaluation for interactive fiction. In *Proceedings of The Artificial Intelligence and Interactive Digital Entertainment Conference Workshop on Optimizing Player Satisfaction*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: The MIT Press.

Tan, C. T., and Cheng, H. 2007. Personality-based Adaptation for Teamwork in Game Agents. In *Proceedings of the Third Conference on Artificial Intelligence and Interactive Digital Entertainment*, 37–42.

Tan, C. T., and Cheng, H. 2008a. A Combined Tactical and Strategic Hierarchical Learning Framework in Multi-agent Games. In *Proceedings of the ACM SIGGRAPH Sandbox Symposium on Videogames*.

Tan, C. T., and Cheng, H. 2008b. TAP: An Effective Personality Representation for Inter-Agent Adaptation in Games. In *Proceedings of The Artificial Intelligence and Interactive Digital Entertainment Conference*.

Tan, C. T., and Cheng, H. 2008c. TAPIR: TAP with Input Reduction for Inter-Agent Adaptation in Modern Games. In *Proceedings of The International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia, Educational & Serious Games (CGames 2008)*.

Tan, C. T., and Cheng, H. 2009. IMPLANT: An Integrated MDP and POMDP Learning AgeNT for Adaptive Games. In *Proceedings of The Artificial Intelligence and Interactive Digital Entertainment Conference*.

Thue, D.; Bulitko, V.; Spetch, M.; and Wasylishen, E. 2007. Interactive storytelling: A player modelling approach. In *Proceedings of The Artificial Intelligence and Interactive Digital Entertainment Conference*, 43–48.

van der Sterren, W. 2006. *AI Game Programming Wisdom 3*. Hingham, Massachusetts: Charles River Media, first edition. chapter Being a Better Buddy: Interpreting the Player's Behavior, 479–494.