

TAP: An Effective Personality Representation for Inter-Agent Adaptation in Games

Chek Tien Tan and **Ho-lun Cheng**

Department of Computer Science
 National University of Singapore
 3 Science Drive 2, Singapore 117543
 {tancheckt, hcheng}@comp.nus.edu.sg

Abstract

Tactical Agent Personality (TAP) is a modeling concept to capture tactical patterns in game agents, based on a personality concept introduced by Tan and Cheng (2007), to allow behavior adaptation to different play styles. We introduced a weighted sequential topology to the actions set to capture tactical preferences rather than individual action preferences. This produces a personality representation of much higher expressive power that improves the adaptation performance and subsequently enables a larger variety of action genres to be adaptable. A TAP-based learning framework is then constructed and it is shown to perform better than the one based on the previous agent personality. Consequently, we also implement an RPG scenario that demonstrates its ability to generate adaptive plausible behavior in a much larger variety of action genres.

Introduction

Increasingly complex modern games genres have aroused tremendous interests in game agent intelligence. Role Playing Games (RPG), Real Time Strategy (RTS) games, First Person Shooters (FPS) and Sports games make up the major components of modern games and game agents constitute the core of the game worlds. A game agent is defined as a fictional character in the game world, either a player controlled character (PC) or a non-player character (NPC). From substantial roles like the player's team mates and opponents, to simpler roles like blacksmiths and innkeeper, NPCs fill a variety of roles. Producing intelligent and interesting behaviors in game agents has been a topic of much interest in the domain of game Artificial Intelligence (AI) (Horswill & Zubek 1999; Hussain & Vidaver 2006; Geramifard, Chubak, & Bulitko 2006; Khoo & Dunham 2002; McDonald *et al.* 2006; Sweetser 2005; White & Brogan 2006).

In this paper, we propose an agent adaptation framework that:

- enables adaptation to different tactical styles of another game agent, so as to achieve a common game goal
- utilizes sequential action preferences to capture tactical footprints

- provides a uniform framework irregardless whether PC or NPC

We find these valuable aspects lacking in current game AI research and current work inadequate. To accomplish our task, we base our learning framework on the Tactical Agent Personality (TAP) - a new perspective of inter-agent adaptivity via a generic personality representation model. It is based on a previous related work (Tan & Cheng 2007) which we found lacking important aspects.

Motivation

Although modern game AI has improved vastly over the years, adaptation to different player tactical styles is still lacking in game agents. This makes NPCs uninteresting and predictable, sometimes even irritating. In RPG games, fully automated NPCs like those in *Diablo II* often get lost and draw unwanted enemies to the player. On the other end, highly controllable NPCs like those in *World of Warcraft* require a lot of manual instructions in order for them to work the way the player expects, according to the player's strategy. This problem is escalated in RTS games, where the number of agents are huge and too much micro-management needs to be involved. In current FPS games like *No One Lives Forever 2*, ally NPC bots are able to react to a multitude of combat scenarios like automatically finding cover and performing flanking maneuvers on the enemies instinctively. However, the ally agents act the same way regardless of whoever is playing the game. In all, most AI in current game agents may react automatically to new unintended situations, but does not tailor to different styles and levels of players. More generally, the agents are not aware of the tactics of other agents around them.

To interpret other agents behaviors, capturing sequential information is essential, apart from only knowing their actions. For example in a doubles tennis game, if my teammate frequently runs in front of the net right after his serve, I would know that he is an aggressive player, and thereby act according to that. In games, a non-temporal method called personality-based adaptation has been proposed previously (Tan & Cheng 2007). It is based on capturing action preferences of other agents in a compact, simple form as shown in the left image in Figure 1. Although the framework is shown to work effectively to a certain extent, it suffers from two serious limitations by ignoring temporal information. The first

is a lack of descriptive power which might deter adaptation performance. The second problem is that agents might produce erratic behavior when actions of a different genre is added to their adaptable set. For example, a warrior in an RPG has a *rage* action which boosts its attack power temporarily, meaning the logical action after going into *rage* should be *attack*. The agents of the previous work may just *idle* or *dodge* after that because of the probabilistic model which ignores sequence. This makes the NPCs look unintelligent and might even cause frustration to the player when it hinders game goals.

The last aspect is the importance of representing and interpreting both NPC and PC behaviors equally. In growingly popular massive multi-player online games (MMOGs), e.g. *Guild Wars*, multiple PCs and NPCs are mixed in a team for game missions. As the agent architecture for both are the same, there is no need for different paradigms to represent their behavior profile. Moreover some game worlds deliberately obscure the differences. Hence a homogenous representation is advantageous for simplicity and efficiency.

Our Approach

To capture the temporal sequence of actions uncertainty in a direct manner, we extend a previous work in personality-based adaptation (Tan & Cheng 2007), inspired by the concept of Bayesian networks (Russell & Norvig 2003) (Figure 1). Instead of placing weights on each action, we include edges between each action and place weights on these edges. This captures the preferential uncertainty of the action transitions in a natural and straightforward manner. In doing so, we are trying to capture the tactical patterns in a representation, as opposed to only recording the action preferences previously. Hence we name it Tactical Agent Personality (TAP), the details of which are described in the later section on *Tactical Agent Personality*. By including temporal knowledge in the model, we also aim to resolve the limitation as described above as well as enable even more flexibility and accuracy in adaptation. In a nutshell our improvements are:

- improving adaptation performance and accuracy
- enhancing applicability to different games by enabling more action genres to be adaptable

In this paper, we will focus on ally NPCs that adapt towards the PC as well as amongst each other. We find that this aspect is lacking as most current game AI research aims to create better opponents to challenge the player. We generalize the notion of game agents so as to make our system more flexible and extensible to different types of games, especially MMOGs with mixed PCs and NPCs. In MMOGs that focus only on human players, our framework can also make use of the adaptation information to provide tactical guidance about the other players.

Similar to the previous work, the inter-agent adaptation process in this paper will be independent from the action planning process. This enables a focused approach to agent-adaptation and leaves the agent-environment adaptation process to the action planning. This decouples the computationally intensive machine learning process from the

low level action selection process. Also, this provides more modularity which enables our inter-agent adaptive system to work with various action selection methods from simple rule-based action selection systems to newer paradigms like Goal-Oriented Action Planning (GOAP) (Orkin 2004).

In the remaining sections of this paper, we will first review work related to our area of research. We then define the TAP representation and describe the solution framework built on top of it. Next, we show our experimental results as empirical proof of our concept. Lastly, we conclude the paper along with our plans for future work.

Related Work

Research in behavior planning architectures in game agents (Horswill & Zubek 1999; Hussain & Vidaver 2006; Geramifard, Chubak, & Bulitko 2006; Khoo & Dunham 2002; McDonald *et al.* 2006; Sweetser 2005; White & Brogan 2006) generally do not focus on the player's presence. Sweetser (2005) combines influence maps with cellula automata techniques for game agent decision making. Khoo and Dunham (2002) devised a stateless FSM system for game agents to efficiently react to environmental changes. Others (Horswill & Zubek 1999; Hussain & Vidaver 2006) made use of work in robotics and applied them to game agents. In view of various agent controllers, McDonald *et al.* (2006) devised a five-level architecture as an abstracted framework to enable inter-operability amongst them. A prominent advancement in this area was the introduction of dynamic scripting (Spronck *et al.* 2006) where they balanced the practicality of scripting with the learning capability of reinforcement learning.

A number of studies also focused on specific areas of planning like pathfinding (Geramifard, Chubak, & Bulitko 2006; Silver 2005) and tactical positioning (Remco Straatman & van der Sterren 2006; Christian J. Darken 2006). Recent pathfinding work (Geramifard, Chubak, & Bulitko 2006; Silver 2005) concentrated on cooperative, multi-agent pathfinding primarily in RTS games. Straatman *et al.* (2006) provides a simple map representation to aid the game agent in evaluating tactical positions, whilst Darken and Paull (2006) combines level annotation with sensor grid algorithm to allow the agent to dynamically find cover. In all, these studies primarily focus on the adaptivity towards the game environment in general, and primarily leave the player out of the equation. Also, they are mainly devised for agents that go against the player, not along with the player. Again, these work did not address the issue of adaptability towards the human player.

In player-based adaptation in games, there has been much recent work based on the notion of player modeling (Charles *et al.* 2005; Donkers & Spronck 2006; van der Sterren 2006; Thue & Bulitko 2006; Yannakakis & Maragoudakis 2005). Sterren's (2006) work in using a naive Bayes classifier to classifying player behavior has partly influenced the way we have devised the TAP in this paper, as TAP is also based on the notion of capturing tactical behavior characteristics. In other work, Donkers and Spronck (2006) made use of preference functions to evaluate state preferences to predict the next action of the player. Charles *et al.* (2005) presented a

good overview of various ways to gather a player’s profile, and then described ways of making use of it in different aspects of the game. In the work done by Thue and Bulitko (2006), they introduced a concept which they term as goal-directed player modeling, in which they made use of the fact that quest goals are always known in advance in RPG games, to predict the player’s future actions to accomplish them, which limits the method to only quests-driven games. Yannakakis and Maragoudakis (2005) also targeted at action prediction based on a Bayesian network with a subset of attributes of the current world state as the inputs. They were partially able to predict player actions in previously unseen circumstances. Our method also builds on this general Bayesian idea of linking up actions probabilistically, and making use of it to plan actions. Generally, these studies use techniques that aim to predict specific user actions ahead of time, and thereby act on them. This might be extremely resource intensive as player actions take place almost every second, which would likely cause disruptions to the actual gameplay. Moreover, their representations are only aimed at modeling the player actions, and not game agents in general, which limits the flexibility of application.

Tactical Agent Personality (TAP)

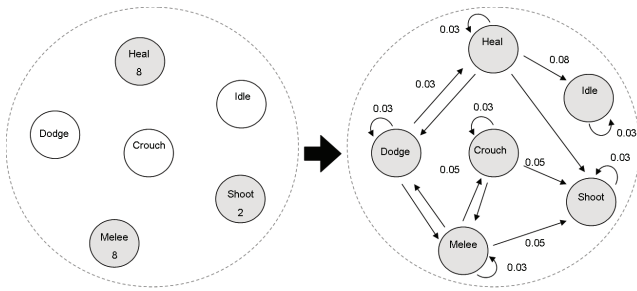


Figure 1: Agent Personality Improvement: In the new TAP model (right), a weighted topology depicting the preferential sequence of actions is added to the previous personality model (left).

The Tactical Agent Personality (TAP) is reformed from the Agent Personality model as introduced by Tan and Cheng (2007). To briefly review the previous model, the Agent Personality is a weighted set of allowable actions as shown in the left image in Figure 1. To exhibit a different personality, it simply means a different combination of weights, since that directly results in different actions if the action selection mechanism depends on the weights. The Agent Personality also serves as a statistics of action preferences for other agents to infer play styles so as to effectively adapt to each other.

However, as mentioned previously, adaptation applicability is severely limited without temporal information lacking in the model. Hence we have reformed the model by including directed edges between each action as shown in Figure 1. Then, instead of placing weights on each action, we place weights on each edge to denote a preference of that transition. Note that the Figure is only illustrative and does not

show all the links for viewing clarity. Formally, if P is the set of all TAPs, the TAP, $P_k \in P$, of an agent k is a function that assigns a value, W , to each action pair.

$$P_k : A \times A \rightarrow W, \quad (1)$$

where A is the set of all allowable actions. Now, the model captures preferences of more descriptive tactical patterns rather than only action preferences previously. As explained, this enables more action genres to be adaptable and improves adaptation performance, which will be shown in the experiments. Description of how the TAP will be used in an adaptive framework will be described in the next section.

TAP-based Adaptation Framework

A formalization of the game agent planning problem can be referred in the previous work (Tan & Cheng 2007). As the focal point of this study is on the enhancements to the personality model, we will be using a similar framework which consists of the following stages as shown in Figure 2. It is basically a learning cycle that enables online training for the adapters belonging to each agent.

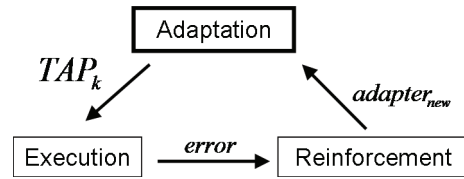


Figure 2: The adaptive game loop: a cyclic process that enables online training for the adapters.

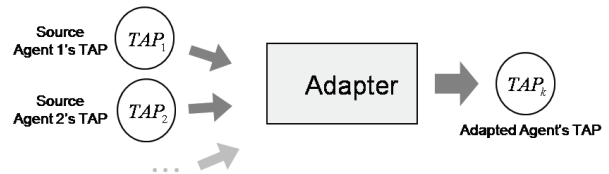


Figure 3: The adaptation process: for each agent, its adapter takes in either one or more TAPs from other agents to generate its own TAP.

Adaptation

Each NPC agent possesses an adapter module to take charge of interpreting other agents’ TAPs and hence generating its own TAP, as shown in Figure 3. As we are dealing with even larger input dimensions (as the TAP is now defined by n^2 edges, where n is the number of allowable actions), the implementation of the adaptor will still be a feed-forward neural network with a single hidden layer, although it can generally be any function approximator. The back-propagation algorithm (Orr, Schraudolph, & Cummins 1999) is used for the network’s learning processes. In general it can be configured to either selectively take in other NPCs’ TAPs as input, or only take in the player’s TAP, which would make it

a purely player-centric adaptation. We leave this flexibility to be exploited by the designer. After each agent obtains its own TAP, the game proceeds to the execution step.

Execution

For the PC, the execution stage involves collection of the player statistics to make up the player’s TAP. Each weight in the TAP is basically an average of the number of times the respective transition has occurred. For the NPCs, this stage is an execution of actions based on their TAP. It is worth pointing out that the end of execution does not always mean a “game over” or “you win” scenario, although it can be defined this way. Execution should end at a suitable predefined time in the game which does not disrupt game-play, for example, map transitions in a RPG, or re-spawn periods in a FPS. At the end of execution, an error is calculated for the reinforcement stage next.

Reinforcement

The error can also be thought of as the reward function of game parameters, based on reinforcement learning terminology (Sutton & Barto 1998). This error is passed on to each adapter to enable it to update the weights of the neural network accordingly. At the end of reinforcement, an updated adapter is being generated for each agent.

Also, at each loop, the adapters need to choose whether to exploit the learnt knowledge and generate a well adapted TAP, or to generate a random TAP to possibly create new learning instances for the adapters (the classical exploration versus exploitation dilemma (Sutton & Barto 1998)). Here, we will adopt the standard ϵ -greedy algorithm (Sutton & Barto 1998) with $\epsilon = 0.2$. This means that at the adaptation step, each adapter will generate a random TAP with probability ϵ , and exploit the knowledge to generate the best adapted TAP otherwise.

Re-iterating the advantages of this workflow, we can see that the resource-intensive adaptation and reinforcement processes only happen in between executions, hence eliminating disruptions to the main game-play. Also, the occasional exploration mechanism adds variety to NPC behaviors which enhances player entertainment value (Spronck 2005).

Empirical Evaluation and Discussion

To evaluate our TAP-based framework, we have created a typical battle scenario in a mechanical version of an RPG game. We chose an RPG game as each game character possesses a good variety of different action genres. A screenshot of the planar view is as shown in Figure 4.

Game Mechanics

The game is modeled to mimic the complexity and uncertainty of a modern game. Different action genres that are vastly different in nature are created in the game as shown in Table 1. Each agent can be one of 3 classes of characters typically found in RPG games, and each can possess a possibly different AI controller. Every agent possesses the basic actions *idle* and *dodge*, but each character class possesses

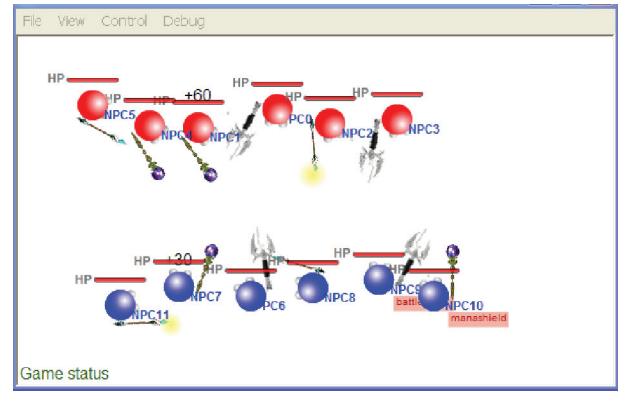


Figure 4: A screenshot of our experimental game scenario. Two teams of agents consisting of warriors, mages and priests pit against each other. They can be differentiated by the weapons they carry.

different personal attributes and actions as briefly shown in Table 1.

We have also modeled various aspects of uncertainty of a typical RPG game so as to show practical applicability of our algorithm. All class-specific actions have a certain chance of whether it will succeed, miss or critical and amount inflicted calculated based on the class attributes, weapon attributes, and character attributes. In addition, all attacks have a range and cool-down time to re-use it as well.

Experiments and Results

To verify the claim that our TAP-based framework performs better and that we can enable more action genres with plausible behavior, we have the following experimental setups.

Multi-agent TAP-based Adaptation Here we demonstrate a basic proof of concept that our adaptation architecture works in complex game environments with a variety of different action genres. We also setup a game such that it makes use of all the different action genres as depicted in Table 1. The game is setup as a battle between two teams with the same setup. Each team consists of 1 PC and 5 NPCs, and each team consists of 2 warriors, 2 mages, and 2 priests. The first team will be using our TAP-based framework (we call this team-TAP), and it will be going against a team with synthetically scripted behavior (team-scripted) in a series of runs.

The results are as shown in Figure 5. The S value on the y-axis is basically a score value that is the normalized difference between the hitpoints of team-TAP minus team-scripted. It is a value between -0.5 and 0.5 , the former meaning team-TAP has been wiped out with all team-scripted agents having full hitpoints, and vice versa. It can be seen that the score is improving as the game proceeds, meaning the adaptation does proceed to make the agents perform better.

Improved Performance To compare against the performance of our new TAP-based framework against the previ-

Table 1: Action Descriptions

Class	Action	Description
All	idle	increase small amount HP, every small time period
	dodge	decrease chance to take damage, for a time period
Warrior	slash	medium damage to target, close range
	battleshout	increase max hitpoints, for a time period
	rage	increase strength, (damage based on strength)
Mage	fireball	huge damage to target, long range
	manashield	decrease damage taken, for a time period
	frostbolt	small damage target, medium range, target speed reduces
Priest	heal	heal target, medium range
	mindblast	small damage to target, medium range
	resurrect	bring back dead target, (use once only)

ous (Tan & Cheng 2007), we have duplicated the game scenario in their game setup. As with that experiment, we used 6 agents and only created the actions *melee*, *shoot* and *heal* and only made them adaptable. We then pit our team using the TAP-based framework (team-TAP) against a team using the old framework (team-AP), both with untrained adapters at first. The results are as shown in Figure 6.

As shown in the graph, team-TAP consistently performs better than team-AP. It is observed that there are multiple periods where the graph tries to go downwards but we acknowledge that team-AP has an AI of its own that adapts as the game proceeds. In general the line stays above 0 with an average of 0.125 which means team-TAP wins a vast majority of the time. This shows that our new framework is indeed performing better.

Enabling More Action Genres This setup illustrates the limitation that the previous framework (Tan & Cheng 2007) produces erratic behavior when a different action genre is added to the set of adaptable actions. For the warrior class there is a *rage* action (as shown in Table 1) that increase the damage it can inflict within a certain period of time. Hence it is much more logical that an attack (*slash*) should be executed when the rage is in effect. During a series of runs, we tabulate the number of times *slash* is being executed when

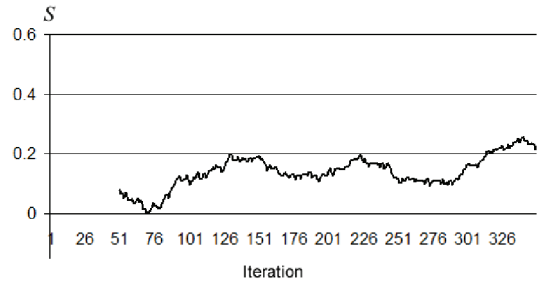


Figure 5: Experimental Results: Plot of S against number of iterations. S depicts the performance difference in team-TAP vs team-scripted.

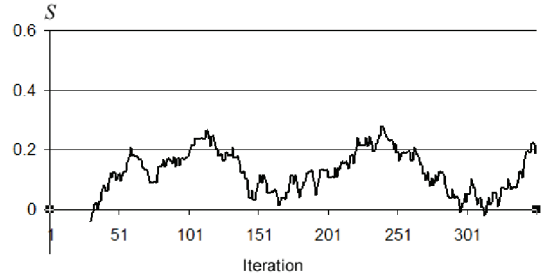


Figure 6: Experimental Results: Plot of S against number of iterations. S depicts the performance difference in team-TAP vs team-AP.

rage is in effect, for both the warrior using a trained TAP-based framework (warrior-TAP), as well as for one using the previous framework (warrior-AP) (Tan & Cheng 2007). As shown in Table 2, the number of times warrior-AP uses slash is much lesser than warrior-TAP during rage. Although it can be seen our system is not perfect, but we have successfully reduced the behavioral oddity by a great amount.

Table 2: Slash Count on 500 Rages

	Warrior-TAP	Warrior-AP
slash count	431	99

Conclusion and Future Work

We have presented our efforts at devising a generic tactical representation model called TAP, which serves as a basis for inter-agent adaptation that treats PC and NPC personalities in a uniform manner. The advantage of this is particularly obvious in current MMOGs where PCs and NPCs are built upon the same agent framework and co-exist in the same environment. Not only does it allow NPCs to automatically conform to different play styles of its allies, it allows players to receive tactical guidance about what other agents are doing, be it player or not.

Our TAP representation is an addition of a weighted sequential topology based on the personality representation

found in a previous work (Tan & Cheng 2007). We built upon its straightforward and expressive representation and improve upon its performance and shortcomings of applicability. We also verify our improved algorithm in a much more complex RPG experimental setup than before and plausible results are observed.

As future work, we hope to alleviate the curse of dimensionality as we have introduced much more complexity into the algorithm by adapting using the edges. This essentially makes the number of inputs into each adapter $m * n^2$ where n is the number of actions and m is the number of ally agents to be adapted to. We are currently looking at either pruning the infrequent edges dynamically or using Principal Components Analysis (PCA) (Michael E. Tipping 1999) for dimensionality reduction whilst maintaining expressiveness of the inputs. In other areas, we also aim to evaluate the improvements on entertainment value by using a user survey method. This aspect is important as it is vital for practical applications to modern commercial games.

References

- Charles, D.; Kerr, A.; McNeill, M.; McAlister, M.; Black, M.; Kcklich, J.; Moore, A.; and Stringer, K. 2005. Player-centred game design: Player modeling and adaptive digital games. *In Proceedings of the Digital Games Research Conference* 285,298.
- Christian J. Darken, G. H. P. 2006. Finding Cover in Dynamic Environments. *In AI Game Programming Wisdom 3*. Hingham, Massachusetts: Charles River Media, first edition.
- Donkers, J., and Spronck, P. 2006. Preference-Based Player Modeling. *In AI Game Programming Wisdom 3*. Hingham, Massachusetts: Charles River Media, first edition.
- Geramifard, A.; Chubak, P.; and Bulitko, V. 2006. Biased cost pathfinding. *In Proceedings of the Artificial Intelligence and Interactive Digital Entertainment conference* 112,114.
- Horswill, I., and Zubek, R. 1999. Robot architectures for believable game agents. *In Proceedings of the 1999 AAAI Spring Symposium on Artificial Intelligence and Computer Games, AAAI Technical Report SS-99-02*.
- Hussain, T. S., and Vidaver, G. 2006. Flexible and purposeful npc behaviors using real-time genetic control. *In Proceedings of The IEEE Congress on Evolutionary Computation* 785,792.
- Khoo, A., and Dunham, G. 2002. Efficient, realistic npc control systems using behavior-based techniques. *In AAAI Technical Report* 02-01.
- McDonald, D.; Leung, A.; Ferguson, W.; and Hussain, T. 2006. An abstraction framework for cooperation among agents and people in a virtual world. *In Proceedings of the Second Conference on Artificial Intelligence and Interactive Digital Entertainment*.
- Michael E. Tipping, C. M. B. 1999. Probabilistic principal component analysis. *Technical Report NCRG/97/010, Neural Computing Research Grou.*
- Orkin, J. 2004. Applying Goal-Oriented Action Planning to Games. *In AI Game Programming Wisdom 2*. Hingham, Massachusetts: Charles River Media, first edition.
- Orr, G.; Schraudolph, N.; and Cummins, F. 1999. Cs-449: Neural networks lecture notes. Accessed December 20, 2005. Available via <http://www.willamette.edu/gorr/classes/cs449/intro.html>.
- Remco Straatman, A. B., and van der Sterren, W. 2006. Dynamic Tactical Position Evaluation. *In AI Game Programming Wisdom 3*. Hingham, Massachusetts: Charles River Media, first edition.
- Russell, S., and Norvig, P. 2003. *Probabilistic Reasoning*. Hingham, Massachusetts: Pearson Education International, second edition.
- Silver, D. 2005. Cooperative pathfinding. *In Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment conference* 117,122.
- Spronck, P.; Ponsen, M.; Sprinkhuizen-Kuyper, I.; and Postma, E. 2006. *Adaptive Game AI with Dynamic Scripting*. Netherlands: Springer Netherlands.
- Spronck, P. 2005. A model for reliable adaptive game intelligence. *In IJCAI-05 Workshop on Reasoning, Representation, and Learning in Computer Games* 95,100.
- Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: The MIT Press.
- Sweetser, P. 2005. An emergent approach to game design. *Ph.D Thesis*. Available via <http://www.itee.uq.edu.au/penny/publications>.
- 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.
- Thue, D., and Bulitko, V. 2006. Modeling goal-directed players in digital games. *In Proceedings of the Artificial Intelligence and Interactive Digital Entertainment conference* 285,298.
- van der Sterren, W. 2006. Being a Better Buddy: Interpreting the Player's Behavior. *In AI Game Programming Wisdom 3*. Hingham, Massachusetts: Charles River Media, first edition.
- White, C., and Brogan, D. 2006. The self organization of context for multi agent games. *In Proceedings of 2nd Annual Conference on Artificial Intelligence in Interactive Digital Entertainment*.
- Yannakakis, G. N., and Maragoudakis, M. 2005. Player modeling impact on players entertainment in computer games. *In Springer-Verlag: Lecture Notes in Computer Science* 3538:74.