Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2011, Article ID 107160, 16 pages
doi:10.1155/2011/107160

Research Article

Tactical Agent Personality

Chek Tien Tan[1] and Ho-lun Cheng[2]

[1] *Games Studio, Center for Human Centred Technology Design, School of Software, Faculty of Engineering and Information Technology, University of Technology, Sydney, Building 10, Level 4 Broadway NSW 2007, Australia*
[2] *Department of Computer Science, National University of Singapore, 3 Science Drive 2, Singapore 117543*

Correspondence should be addressed to Ho-lun Cheng, hcheng@comp.nus.edu.sg

Received 30 August 2010; Revised 21 December 2010; Accepted 11 March 2011

Academic Editor: Soraia Raupp Musse

This paper proposes a novel agent personality representation model used to provide interagent adaptation in modern games, coined as the Tactical Agent Personality (TAP). The TAP represents the tactical footprints of a game agent using a weighted network of actions. Directly using the action probabilities to model an agent's personality, removes the time and effort required by experts to craft the model as well as eliminates the performance dependency on expert knowledge. The effectiveness, versatility, generality, scalability, and robustness claims of the TAP architecture and its variations are applied and evaluated across a variety of game scenarios, namely, First-person shooters (FPSs), real-time strategy (RTS) games, and role-playing games (RPG), where they are shown to exhibit plausible adaptive behavior.

1. Introduction

In the player modeling domain of modern game AI, current approaches are hard to generalize as they require manual expert knowledge to be incorporated in each game. Player modeling (obtaining a useful representation of the player for adaptive game play) is one aspect of game AI that needs to be emphasized because it represents a class of methods to enable the synthesis of agent behavior adaptive towards the player. And the player is one of the most important factors in creating intelligent game agents. Current methods [1–5] mainly involve classification tasks based on manually specified player archetypes. This requires much specialized expert knowledge to be incorporated and hence limits the generalization capabilities of these methods in different games.

Many studies in player modeling have been used for player action prediction. Donkers and Spronck [6] made use of preference functions to evaluate state preferences in order to predict the next action of the player. Thue and Bulitko [7] introduced a concept which they term as goal-directed player modeling, whereby 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. This limits the method to only quest-driven games. Yannakakis and Maragoudakis [8] also targeted at action prediction based on a Bayesian network with a subset of attributes of the current world state as the inputs. Van der Sterren's [1] work uses a naive Bayes classifier to classify player behavior into different archetypes and then use it to customize the NPC agent's tactical behavior.

A number of player modeling approaches [2–5] have been studied in the field of interactive storytelling whereby the game story is automatically adjusted online according to the play style. In general, they use similar techniques based on the tracking of player actions and using them to adjust a vector of player archetypes. For example, in El Nasr's work [5], the vector of archetypes include relunctant hero, violent, coward, truth seeker, and self-interested.

As mentioned, these methods share a similar problem. They mainly try to classify the player into archetypes, and it was unclear how these archetypes were formulated, and from their trait names, it seems possible that considerable expert domain knowledge is needed to generate them. This limits their generalization capabilities.

In view of these limitations, this paper hence describes the Tactical Agent Personality (TAP), an effective model used for inter-agent adaptation that eliminates the baggage of requiring expert knowledge whilst providing high-performance adaptation. The TAP model basically uses a statistical model based purely on action footprints, and this model is directly used in a learning framework to affect future action choices, hence eliminating the need for an intermediate level of expert intervention. The details can be found in Sections 3.1 and 3, which describe the basic TAP model and the corresponding TAP-based adaption framework, respectively.

In the derivation of the TAP, a collection of work has been performed to develop the TAP-based framework, and this paper basically consolidates these efforts as a collective and holistic proof of the feasibility and versatility of the TAP model in the various modern games genres. The subsequent sections describe these studies in detail, namely, the basic TAP used in a multiagent cooperative FPS game [9] (Section 3.1), Strategic Agent Personality (SAP) used in a hierarchical learning framework for an RTS game [10] (Section 4), and the TAP model based on temporal links used in an RPG game [11] (Section 5). The paper lastly provides a deeper analysis of the conclusions in view of all the studies as a whole, as well as a discussion of future work in Section 6.

2. A Simple but Sufficient Personality Model

Generally, the TAP model can be described as a statistical record of agent actions and hence can be directly modeled without any human intervention. For any agent in the game (PC or NPC), its TAP consists of a set of actions (that are allowed to be adaptive), with each action tagged with a relative action value as shown in Figure 1. 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 an action value to each action

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

where A is the set of all allowable actions.

As can be expected of such an arrangement, this simply means that when an action selection mechanism is applied, the action value determines the chance of choosing that action in view of other simultaneous actions. That is to say, the values are normalized to determine a probability of choosing it. For example, if the actions set only contained the actions *Melee*, *Shoot,* and *Heal* with action values 0.6, 0.8, and 0.6, respectively, then the resultant probabilities that each of these actions will be taken are $0.6/(0.6 + 0.8 + 0.6) = 0.3$, $0.8/(0.6 + 0.8 + 0.6) = 0.4$, and $0.6/(0.6 + 0.8 + 0.6) = 0.3$, respectively. Hence, combining different action values in the set exhibits different personalities. This is a simple, practical, and flexible way to define personality in agents. A change in personality inevitably leads to a change of actions being exhibited, hence, affecting the agent's behavior. Intuitively, this is also true in humans, as our personalities directly translate into the actions we take in life, and these actions do not happen with absolute certainty every single time.

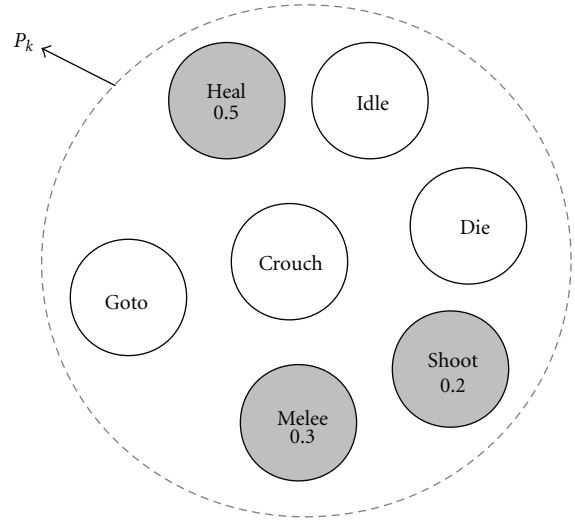

Figure 1: An example of Tactical Agent Personality (TAP). Each action is tagged with a relative action value that can be evaluated into a probability of choosing it. The shaded ones are the adaptable actions whilst the unshaded ones are nonadaptable.

The TAP is a generic personality model that can be utilized differently in the different contexts. For the PCs, the TAP is used to capture action footprints of the player, or in other words, a statistical record of the actions taken. The details of how this is done is left to the implementer as the focus of this paper is on the framework and not the implementation details. However, the most straightforward way would be to obtain a cumulative average count of actions over time. On the other hand, for the NPCs, the TAPs are automatically generated and learnt over time (as depicted in the next section), depending on the PCs' TAPs. Using this approach, adaptation can be limited within the space of personalities, and action planning can take place thereafter, independent of the adaptation process. This decouples the potentially time-consuming adaptation process from the actual action planning process, which allows more flexibility for the game programmers to tweak performance.

Note also that in Figure 1, it can be seen that there are actions that can be labeled as nonadaptable (like the *die* action). This is simply an option that provides the game designer the ability to turn off certain actions in case some of these actions appear to result in erratic behavior after user trials are performed. In the industry, this might be an important step to fine-tune some of these NPC behaviors in shipped versions of the games. That being said, however, in the research sense, all actions should be made adaptable to achieve the most generically behaving NPCs. This paper leaves this option open to allow for more flexibility.

Although the concept of TAP is simple, it is shown to be a powerful asset when used in an adaptation framework, as will be seen in the subsequent sections. 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 next subsection describes an adaptation framework that makes use of TAP.

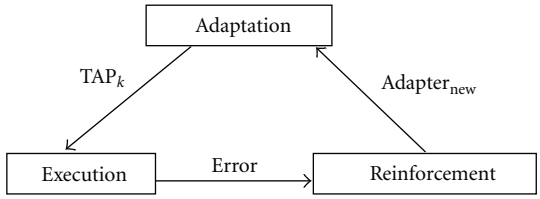

Figure 2: The adaptive game loop for the TAP adaptation framework. A cyclic process that enables online training for the adapters.

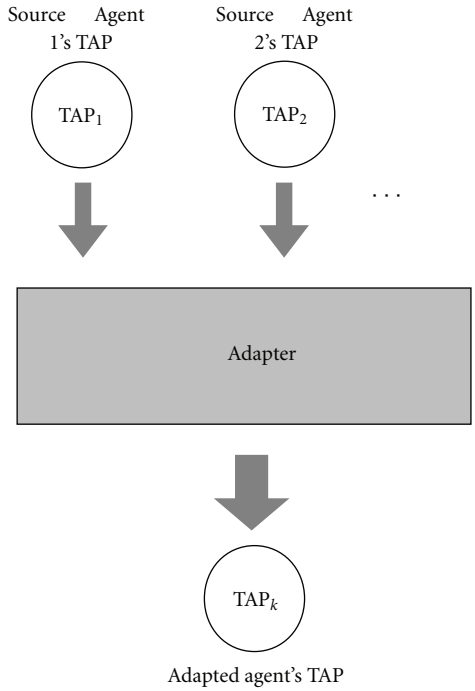

Figure 3: The TAP adaptation process. For each agent, its adapter takes in either one or more TAPs from other agents to generate its own TAP.

3. The TAP Adaptation Framework

The TAP adaptation framework [10] is as shown in Figure 2. It is basically a learning cycle that enables online training for the *adapters* (as shown in Figure 3) belonging to each agent. The details of each process are as follows.

(1) Adaptation. Each NPC agent possesses an adapter module which is responsible for interpreting the other agents' TAPs and hence generating its own TAP, as shown in Figure 3. Again, it is to be highlighted that this paper focuses on the framework and not the implementation details. As an initial implementation, a feed-forward neural network with a single hidden layer is chosen as the adapter might possibly deal with large input dimensions, although it can be any function approximator in general. A diagram of the network structure is shown in Figure 5, and more implementation details can be found in Section 3.1. Note that before using the action values derived from the output of the neural network, they would need to be converted into probabilities as described in Section 2. The back-propagation algorithm [12] is used for the network's learning processes. Basically, the adapter can be configured to either selectively take in the other NPCs' TAPs as input, or only take in the player's own TAP, which would make it a purely player-centric adaptation. After each agent obtains its own TAP, the game proceeds to the execution step.

(2) Execution. For the PC, the execution stage involves a 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 action 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 an RPG or respawn periods in an FPS. At the end of execution, an error is calculated for the next stage, which is reinforcement.

(3) Reinforcement. The error can also be thought of as the reward function of game parameters, based on reinforcement learning terminology [13]. 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 [13]). Here, the standard ϵ-greedy algorithm [13] is adopted. 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.

The ϵ-greedy algorithm, though simple, has an intrinsic advantage for modern games. Other than effectiveness in the agent behavior, diversity also plays a part in making game AI interesting to the player [14]. In an adversarial example, when the enemy NPC AI is effective, it provides a challenge for the player to overcome. When the AI is diverse, it provides a variety of challenges for the player to play against, especially when the player gets tired of trying to beat the optimal agent. A similar statement can be made for cooperative game play in ally NPC agents. The apparent randomness provided in the ϵ-greedy algorithm provides diversity in the agent behavior, as an enhancement to the effectiveness. Hence, the ϵ variable can also be thought of as a mediator between effectiveness and diversity, both of which are important criterions in keeping game play interesting.

In this workflow, it can be seen 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 [15].

Figure 4: A screenshot of the fundamental TAP experimental game environment. The game represents a typical FPS setting. The PC is in the center, and each of the other NPCs in white are busy with their own tasks chosen. The opponents are the zombies scattered over the map in packs.

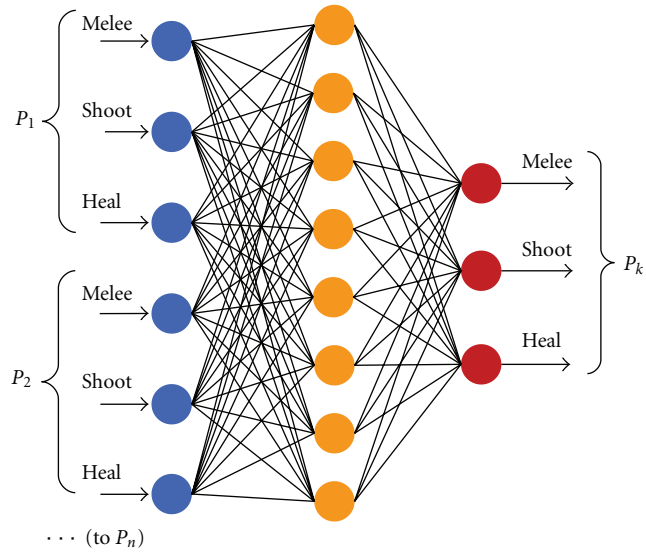

Figure 5: Experimental setup of the fundamental TAP model tests: the neural network implementation of the adapter. This diagram shows the neural network structure that is implemented for the experiments. The number of input neurons (blue-colored neurons) is equal to $n_t \times n_a$, where n_t is the number of input TAPs and n_a the number of adaptable actions in each TAP. The number of output neurons (red-colored neurons) is hence n_a. The number of hidden neurons (orange-colored neurons) is 8.

3.1. Evaluation. An initial evaluation of the basic adaptation framework based on the fundamental TAP formulation is performed using an actual game setting. A typical FPS game scenario is built and a screenshot is as shown in Figure 4.

3.1.1. Experimental Methodology. The experiments in this section aim to establish tests that can determine whether the TAP model can provide *competent* adaptation towards the player. Competency can be further separated into the dimensions

(1) effectiveness,

(2) versatility,

(3) robustness,

of the TAP-based adaptation framework. Effectiveness measures the capability of the framework to provide the intended results, which is for the agents to adapt and win the game. Versatility measures how well the framework applies to different player types, which aims to show that the TAP-based framework can serve its primarily goal of providing player-centric adaptation. Robustness measures the consistency of the results in different situations, so as to ascertain the validity of the results. The sections that follow describe the game setups that aim to fulfil these test goals.

3.1.2. Experimental Setup. The goal of the game is to kill all the zombies in the game world. The PC is a marine equipped with a laser gun with limited range. The player is accompanied with 5 ally NPCs carrying the same type of gun. The PC and his NPCs can also run up close to inflict melee damage on the opponent. The zombies are packed in groups and when any of the agents are in range, they will approach the respective agent and melee attack by biting them. The game ends when either all zombies are killed or the PC is killed. An agent's melee attack does double the damage compared to shooting because the agent exposes itself to the attacks of the zombies and attracts more zombies towards it. All attacks have a certain miss chance which results in no damage to the target when it happens. Also, all attacks have a chance of hitting critically for twice the damage.

In these experiments, only the actions *melee*, *shoot*, and *heal* are allowed to be adaptable, so that the results can be more obvious. An example personality of that used in our experimental setting is as shown in Figure 1. The action planning system utilized in the experiments is basically a rule-based system. The NPCs generally follows the PC wherever he goes. At each zombie encounter, the NPCs will choose to either run up close and melee the zombie, stay far and shoot it, or if an agent's health is less than full, decide whether to heal him. The choice of these three actions are dependant on the action values defined in the agents' TAPs.

All the result graphs to be shown are based on analyzing an error value E, which is calculated based on the following formula:

$$E = \sum_{i \in I} \left| \frac{\varsigma_i}{\sum_{j \in I} |\varsigma_j|} \times V(i) \right|, \tag{2}$$

where I is the set of game attributes useful in determining the success rate, $V(i)$ is the value of attribute i, and ς_i represents the corresponding coefficients to balance the weight of each attribute $i \in I$. In this evaluation, there are only two elements in I. The value $V(1)$ of the first element $i = 1$ represents the total number of ally NPC agents dead, and the value $V(2)$ of the second element $i = 2$ is the total number of zombies alive. The corresponding coefficients ς_1 and ς_2 are 0.3 and 0.7, respectively. These two attributes are the most straightforward metrics to determine the success

rate of the game. More weight was given to the former attribute because the goal of the game is to eliminate the zombies, and that the latter attribute is more of a factor to determine teamwork effectiveness, though it does have an effect on the game's goal. Thus, it can be seen that E inversely represents the degree of successful game play, which can be used to determine the effectiveness of the TAP-based framework.

As shown in Figure 5, The adapter is implemented as a feed-forward neural network (with a single hidden layer) using back propagation [12]. In each iteration, the agent will choose to utilize the most updated TAP with probability ϵ and generate a new random TAP otherwise (with probability $1 - \epsilon$). As the network weights will be continually updated in each iteration, the initial network weights are simply randomized.

The ϵ is initially set at 0.5 to allow for more exploration and then progressively decreased to allow for more exploitation thereafter. In the rounds that a random TAP is generated, a threshold value *Thres* is set (at $THRES = 0.2$) to determine whether the explored TAP performs good enough to warrant a reinforcement. If $E < THRES$, the reinforcement is performed, which consists of the following weight updates on each network link:

$$w'_{ij} = w_{ij} + \kappa \Delta w + \eta \delta_j o_i, \tag{3}$$

where w'_{ij} and w_{ij} are the new and previous network weights from neuron i to neuron j, κ is the momentum (set at 0.9), Δw is the last change in weight, η is the learning rate (set at 0.7), δ_j is the error at neuron j, and o_i is the output at neuron i. The activation function uses the normal sigmoid function. The error δ_j at each output neuron accounts for the difference between the network generated outputs (o_j) and the outputs that generated good performance (t_k)

$$\delta_j = o_j\left(1 - o_j\right)\left(t_k - o_j\right). \tag{4}$$

The error δ_j at each hidden neuron is a weighted sum of all the outgoing errors (δ_j) of the neurons it is connected to,

$$\delta_j = o_k(1 - o_k) \sum w_{jk}\delta_k. \tag{5}$$

The number of input neurons (blue-colored neurons) is equal to $|P| \times |A|$ where $|P|$ is the number of input TAPs and $|A|$ the number of adaptable actions in each TAP. The number of output neurons (red-colored neurons) is hence $|A|$. The number of hidden neurons (orange-colored neurons) is 8. The network topology is fully connected (only in the forward direction) in between each layer. In general, the parameters used in the framework for the experiments aim to achieve a high rate of learning. This is because AI has to be visible quickly in modern games.

3.1.3. Experiments and Results. The experiments devised aim to test the effectiveness and versatility criterions as described above. Robustness is an accompanying criterion that is maintained throughout the tests.

(a) Effectiveness Test. Three sets of experiments are crafted to determine whether the TAP-based framework is effective. In the first set of experiments, the game scenario is run 500 times with scripted NPCs. Secondly, 500 iterations are performed again with each NPC having its adapter adapting to only the PC. In a third set of 500 iterations, the adapters will adapt to the PC as well as all the other NPCs in the team. Each round takes anywhere from 10 seconds to 5 minutes depending on how fast the player is killed or how fast the zombies are all killed.

To establish robustness, each iteration contains randomly generated setups. In a single iteration, the scripted behavior of the NPC is derived from fixed values of each attribute found in the TAP representation. The notion of "script" here is actually not the conventional way of handcrafting sequences of actions. The scripted agents here actually use the same action selection mechanism as the adaptive NPCs. The only difference being that the scripted NPC TAPs do not change over time. For example, one setup might be to fix a value of 0.8 for healing, 0.1 for shooting, and 0.1 for melee. This means the scripted NPC will perform healing 80% of the time and both shooting and melee 10% of the time. The reason for doing so is to maintain the same behavioral space for both teams, with the only difference being the adaptivity based on the TAP. These fixed values are randomly generated at the start of each iteration. Moreover, the positions of the zombies are reshuffled randomly to test for robustness (but with all sets of experiments having the same random seed to ensure fairness). Hence, these experiments aim to test whether the personality, adapted NPCs are better than scripted ones and also to test the online adaptability of our framework.

The effectiveness results are consolidated into a single graph as shown in Figure 6. In general, it can be seen from Figure 6 that the agents using the TAP framework (only adapts to the PC) performs better as compared to scripted agents, with the adaptive agents producing a total average of $E = 0.576$ versus a total average of $E = 0.628$ for scripted ones. Consequently, when using agents that adapt to all other agents, it performs vastly better than both the scripted and the agents that only adapts to the PC, with a total average of $E = 0.393$. This might be due to the fact that the team of agents that adapt to each other has implicitly learnt cooperative behavior with the entire team, whilst the team of agents that only adapt to the PC only learns how to complement the PC.

To determine the significance of the results, unpaired one-tailed two-sample heteroscedastic Student's t-tests [16] are performed on each pair of results. The null hypotheses are that there are no differences between the performance of the teams. The alternative hypotheses to be achieved are that the fully adaptive framework performs better than the player-only adaptive team, which in turn performs better than the scripted framework. Hence, a one-tailed test is used on each two-sample pair of results. All the experiment sets are independent of each other, so an unpaired t-test is used. As no assumption can be made about the variances of each distribution, the heteroscedastic (unequal variance) t-test is chosen. The results of the t-test are as shown in Table 1.

Table 1: Effectiveness test results for the fundamental TAP experiments: Table of t-test P-values.

	Nonadaptive versus adaptive player	Adaptive player versus adaptive	Nonadaptive versus adaptive
T-test P-value	<.010	<.010	<.010

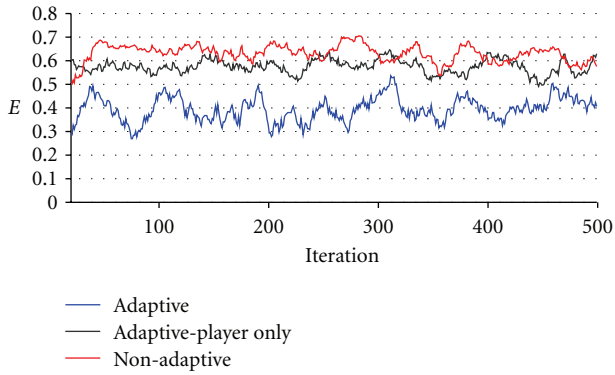

Figure 6: Effectiveness test results for the fundamental TAP experiments: plots of E against the number of iterations. Lower E means better performance. A single point on the graph represents a full game episode. The red-colored line at the top depicts the results with nonadaptive agents, the black line below shows the results with agents adaptive only towards the PC, and the blue line at the bottom shows the results with agents adaptive to all other agents.

It can be seen that the P values are all lower than .01 which implies that it can be said that the results are statistically significant at the 1% level.

(b) Versatility Test. To establish versatility, further experiments are also performed to show the adaptability across different player personalities. In these experiments, one TAP attribute value is varied from 0 to 1 while the other two attributes are fixed. To establish robustness, the game is also run for 500 iterations with each iteration differing in terms of the randomized elements described in the effectiveness test setups.

In the versatility tests, Figure 7 shows the results where the melee and healing attributes are fixed at 0.5 each respectively, whereas the shooting probability is varied from 0 to 1. Similarly, the melee and shooting attributes are fixed at 0.5 whilst varying the healing attribute from 0 to 1. Again, the experiment is repeated for the melee attribute.

The same conclusions as those deduced from Figure 6 (that the team-based adaptive agents perform better than player-only adaptive agents which in turn perform better than scripted agents in various configurations of the PC personalities) can be drawn here also and can thus be now generalized to more instances of the PC personalities. It can also be observed that in the melee setup, there is an abnormality that the nonadaptive team performs just slightly better than the player-only adaptive team. However, this result is insignificant as the P value for this pair is .210 as shown in Table 2.

Compared to the healing and shooting setups, it can also be seen that the improvements gained in healing is diminished. As shown in Figure 7, as the player's healing attribute get higher, the performance of both types of adapted agents gets poorer and closer to the scripted agents, likely due to the fact that he gets killed more often (and when the PC is killed, it marks the end of an execution). This is because in the game mechanics, the healer is especially vulnerable to death as the zombies are scripted to attack healers first (because healing poses a greater threat to the zombies).

To determine the significance of the results, unpaired one-tailed two-sample heteroscedastic t-tests are performed. The results of the t-test are as shown in Table 2. It can be seen that most of the P values are lower than .01 which implies that it is highly unlikely that the results have occurred by chance. However, in the shooting setup, the P-value is .112, which shows that it is likely that the result might be due to chance. This might be due to the fact that changes in the shooting attribute do not produce enough significant change for the TAP-based agents to adapt and produce enough improvement. Nevertheless, there is still well over 85% a chance that the results are significant. Similar observations can be made for the P-value of .017 in the melee setup whereby the significance is at 95%.

4. Strategic Agent Personality

To enhance the TAP model's generalization capabilities, the TAP model was adapted [11] to enable strategic decision making (on top of tactical decision making) as both tactical and strategic planning aspects are important in modern games. As described in the earlier subsections, the TAP consists of tactical actions and hence is responsible for tactical behavior like the immediate slash and heal actions. Much of modern game agent intelligence involves strategic decision making as well, an example of which occurs while deciding when to do a flanking manoeuvre on the enemy. In short, tactical decisions are short term and normally individual based whereas strategic decisions are long term and normally team based. As an example in an RTS game, a commander needs to both microcontrol its subordinates and at the same time decide on its own reactions towards immediate enemy encounters.

With the aim of creating a combined tactical and strategic decision-making mechanism into an agent, first the Strategic Agent Personality (SAP) (as shown in Figure 8) was introduced as a strategic version of TAP. In this SAP (the diagram on the right in Figure 8), the primitive actions in the TAP are replaced with high-level strategies. The commander has several strategies in mind, each of which is assigned

Table 2: Versatility test results for the fundamental TAP experiments: Table of t-test P-values.

	Nonadaptive versus adaptive player	Adaptive player versus adaptive	Nonadaptive versus adaptive
Shooting	<0.010	0.112	<0.010
Healing	<0.010	<0.010	<0.010
Melee	0.210	<0.010	0.017

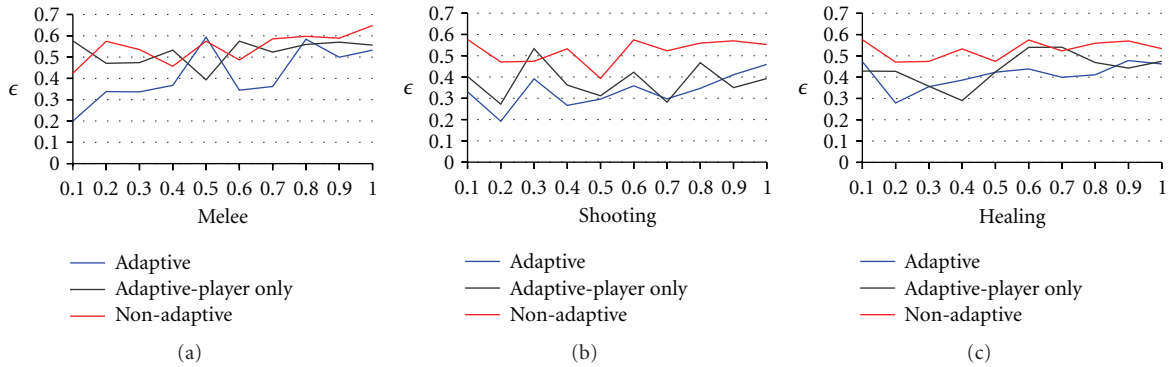

Figure 7: Versatility test results for the fundamental TAP experiments: plot of E against the melee, shooting, and healing probability, respectively. The color code is the same as in Figure 7.

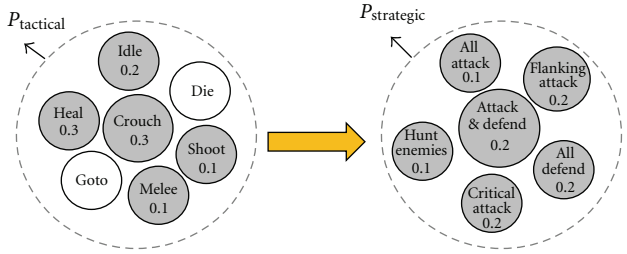

Figure 8: An example of Strategic Agent Personality (SAP). This diagram shows the differences between the fundamental Tactical Agent Personality (left) and Strategic Agent Personality (right). For the TAP, each action is tagged with a relative weight that can be evaluated into a probability of choosing it. The shaded ones are the adaptable actions whilst the unshaded ones are nonadaptable. The SAP is similarly defined.

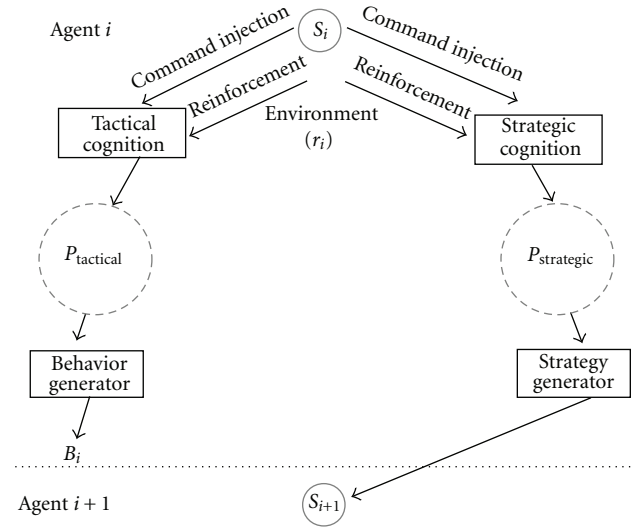

Figure 9: The combined tactical and strategic hierarchical learning framework. The tactical behavior and its strategy for the agents in the current hierarchical level is governed by the strategy being generated from the commander agents in the previous hierarchical level. Central to the framework are the adaptive tactical and strategic personalities which control the selection process for the behaviors and strategies for each agent.

a weight before the start of the decision making process. Similarly, if $\mathcal{P}_{\text{strategic}}$ is the set of all SAPs, the SAP of an agent, $P_{\text{strategic}} \in \mathcal{P}_{\text{strategic}}$, is a function that assigns a weight to each strategy which is determined by the learning framework as elaborated on in later subsections.

$$P_{\text{strategic}} : S \longrightarrow W, \tag{6}$$

where S is the set of all strategies

Next, a hierarchical learning framework was crafted. Every agent in the game possesses the cognitive framework as shown in Figure 9. First, a strategy, s_i, is passed down from a parent agent in the previous hierarchical level. Based on this strategy and taking certain environment variables into consideration, the tactical cognition and the strategic cognition define the tactical personality, P_{tactical}, and strategic personality, P_{strategy}, respectively. Via the tactical personality, the behavior generator outputs a sequence of actions, B_i, for the agent's current tactical behavior. Similarly, the strategy generator makes use of the strategic personality to produce a strategy, s_{i+1}, for use by the next agent(s). Note that our definition of strategy is synonymous with that of a command decision being passed down from a commander to a subordinate.

4.1. Behavior and Strategy Generators. Again, common to any online learning framework, the problem of exploration versus exploitation needs to be addressed. For the behavior generator, the standard ϵ-greedy algorithm [13] is adopted. This basically means that the generator chooses a random tactical behavior sequence with probability ϵ, and exploits the knowledge to generate the best behavior (actions with the highest weight) otherwise.

For the strategy generator, the selection process follows a softmax rule [13] such that a higher w_i value means a higher chance of being selected, where $w_i \in W$ is the weight assigned to strategy $s_i \in S$. The probability, Pr, of selecting a strategy, s_i, at time t is

$$\Pr(s(t) = s_i) = \frac{e^{w_i/\tau}}{\sum_{k=1}^{n} e^{w_k/\tau}}, \tag{7}$$

where n is the total number of strategies and τ is a temperature variable to control the greediness of the approach [13].

These straightforward selection methods ensure that behavior generation and strategy selection can be done very quickly without interrupting or overlapping with actual game play. Also, a change in the weights directly leads to a different tactical and strategic characteristic for an agent, hence enabling a platform for variability of game play which improves entertainment value. Note that strategy generation uses a softmax rule as opposed to the ϵ-greedy method used in tactical behavior generation. This is because strategically a more constant performance is preferred over variability whereas, tactically, variability is just as important (for the purpose of entertainment value as mentioned). A softmax rule ensures that higher weights would always result in a higher chance of being selected whereas the ϵ-greedy method chooses randomly when exploring.

After the strategies are passed down the hierarchy and each agent has determined its behavioral action, the game is executed until the next reevaluation time, T. This reevaluation time is basically the time step that is set for the agents to reevaluate the strategies and hence their behaviors. It can be a periodic time in the game or milestones like map transitions in an RPG or respawn times in an FPS. After each execution, a reward value, r_T, is generated via a reward function. A particular formulation of the reward function in an RTS setting is given in Section 6. This value, r_T, represents the environment factor as shown in Figure 9 which is passed on to the reinforcement process in the tactical and strategy cognitions.

4.2. Tactical and Strategic Cognitions. The tactical and strategic cognitions each perform a two-stage process to determine the personalities to be used in behavior and strategy selection, namely, reinforcement and command injection. In the reinforcement stage, only the behavior and strategy in use (before the current reevaluation time step) are affected. For each adaptable action, j, in the behavior sequence in use, the update function for its weight, w_j, is

$$w_j = w_j + \alpha(r_T - \overline{r_T}), \tag{8}$$

where α is a positive step-size parameter to control the magnitude of change. $\overline{r_T}$ is a reference point to determine whether the current reward is large or small [13], and it can either be a heuristic value or simply the average rewards over all the previous reevaluation time steps until the current time step. Similarly, if strategy i is the current strategy in use, then the update function for the strategy weight, w_i, is

$$w_i = w_i + \beta(r_T - \overline{r_T}). \tag{9}$$

After the tactical and strategic personalities are updated by the reinforcement process, the strategy received from the previous hierarchy is used to temporarily affect the weight values before the selection processes are performed. This is the command injection stage. If s_{i-1} is the strategy passed down from the agent from the previous hierarchy,

$$\begin{aligned} P'_{\text{tactical}} &= C_{\text{tactical}}(P_{\text{tactical}}, s_{i-1}), \\ P'_{\text{strategic}} &= C_{\text{strategic}}\left(P_{\text{strategic}}, s_{i-1}\right), \end{aligned} \tag{10}$$

where P'_{tactical} and $P'_{\text{strategic}}$ are the new personalities, respectively. The functions C_{tactical} and $C_{\text{strategic}}$ can be rule bases that define the effect of each individual strategy on the current weights or a machine learning system trained to assign changes to each of the weights according to the strategy being received. In this work, the experimental setup follows a rule-base system because the nature of our game requires domain knowledge for each strategy received. Having the functions as rule bases provides an avenue for the inclusion of domain knowledge (specific to different game genres) in our framework. The evaluation of this SAP-based hierarchical learning framework will be shown in the section that follows.

The purpose of having a separate reinforcement and command injection stage is to ensure that the framework is able to provide both individual and team behaviors. The reinforcement stage provides for the individual learning of the tactical and strategic personalities according to the reward values. It can also be noted that it might be possible to have what would seem like "conflicting" tactical and strategic decisions. For example, the strategic decision of the team might be to launch an all out *critical attack* but an individual agent might choose to perform *heal* some of the times. In terms of playability, this supposed "conflict" is actually an advantage which adds variability in the NPCs' behaviors. To appear realistic in an attack situation, some NPCs would need to perform healing anyway. Nevertheless, the *heal* action would still be performed much lesser due to the probabilistic effects of the command injection functions as shown above.

4.3. Evaluation. In order to evaluate the combined hierarchical framework in a real-game environment that has a typical strategic component, an RTS scenario is built. A screenshot of the environment is as shown in Figure 10.

4.3.1. Experimental Methodology. The experiments in this section aim at complementing the effectiveness, versatility,

Figure 10: A screenshot of the SAP hierarchical learning framework experimental game environment. The game represents a typical real-time strategy (RTS) game setting. Team A consists of the lighter-colored characters (mainly on the bottom portion of the figure) whilst Team B consists of the dark-colored characters (mainly on the top portion of the figure).

and robustness results shown in Section 3.1, so as to establish two further tests, namely,

(1) generality,

(2) scalability,

of the TAP-based adaptation framework. Generality measures the effectiveness of the framework to cater to varying opponent strategies in a different game genre. Scalability measures how well the framework scales as the number of adaptive agents gets larger. Robustness is an accompanying criterion that also needs to be measured to show consistency of the results. The sections that follow describe the game setups that aim to fulfil these test goals.

4.3.2. Experimental Setup. The test environment consists of two opposing teams with symmetrical initial geometric positions and identical team structures. Each team consists of agents having one base to defend. The team structure of our main experiment is shown in Figure 11. The hierarchy is static with a top level commander agent who directs multiple subteams, with each subteam having a team leader agent and a number of subordinate agents. Neither team would have any tactical or strategic advantage at the start. The experiments are performed in iterations that end when either team wins or a draw occurs. A team wins only when the opposing team's base is destroyed.

The constituents that make up the tactical and strategic personalities are also shown in Figure 11. Tactically, each agent is able to either melee (close range attack with larger damage), shoot (long range attack with lesser damage), or heal an ally agent, much like the agents in the fundamental TAP framework experiments in Section 3.1. The base acts as a turret that has a longer range than agents and can attack a single enemy at a time. Strategically, the options available to the commander agents of each team are common strategies used in RTS games [17]. and are described as follows.

(1) *Hunting attack*. All ally agents would move towards and attack enemy agents first destroying all opponent agents before moving on to destroy the enemy base.

(2) *Critical attack*. All ally agents would move towards the enemy base and try to bring it down. They keep attacking until either the enemy base is destroyed or the whole team is annihilated.

(3) *Flanking attack*. Some ally agents would move towards and attack enemy agents whilst the rest of the ally agents move towards one side of the enemy base and attack it from there.

(4) *All defense*. All ally agents would stay near the ally base and attack any enemy agents that come within range. When all enemy agents are destroyed, they will move towards and attack the enemy base.

(5) *Attack and defend*. Some ally agents would stay near the ally base whilst the rest would move towards and attack the enemy agents or enemy base.

The tactical and strategic cognition basically consists of the reinforcement and command injection stages, respectively. The reward function used for the reinforcement stages is as follows:

$$r_T = \sum_{o_k \in O} \gamma_k |H_T(o_k)| - \sum_{o'_k \in O'} \gamma'_k \left| H_T\left(o'_k\right) \right|, \tag{11}$$

where r_T is the reward at reevaluation time T, O is the set of all objects of the player team (agents, buildings, turrets, or other objects useful in determining the winning chance), and O' is the set of all objects of the computer-controlled team. $H_t(x)$ defines a function that returns the hit points or health of a game object x at time t. γ_k and γ'_k are coefficients to balance the weight of each type of unit, where $\sum_{\text{all k}} \gamma_k = 1$ and $\sum_{\text{all k}} \gamma'_k = 1$.

In this particular experimental setup, $H_t(x)$ simply returns the summation of all the hit points of each agent (including the base agent) in the team. The coefficients γ_k and γ'_k follows a uniform distribution which assigns equal weights to each agent. For the purpose of the experiments, the magnitudes of change and reference point for the reward functions in (9) and (8) are kept simple and uniform. The magnitudes of change are set to $\alpha = 1$ and $\beta = 2$ because the strategies need to be updated faster to induce enough change to be visible to the player. The reference point is set at $\overline{r_T} = 0$ so as to give full weight to the actual rewards calculated in the algorithm.

4.3.3. Experiments and Results. The experiments devised aim to test the generality and versatility criterions as described above. Robustness is an accompanying criterion that is maintained throughout the tests.

(a) Generality Test. The generality test determines whether the TAP-based framework is still effective in a new game

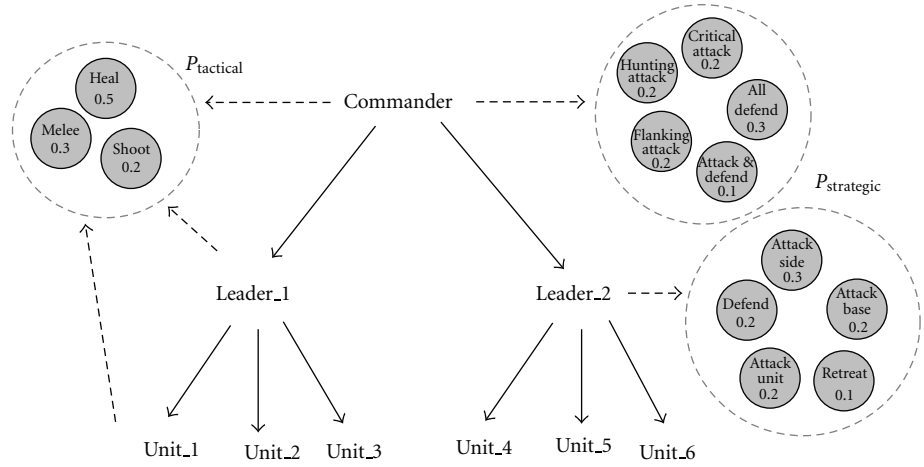

Figure 11: Experimental setup of the SAP hierarchical learning framework tests: units' hierarchy. Each team in the experiment consists of one commander, two subteam leaders, and 6 units. All agents have the same type of tactical personality (with different weights), but the strategic personalities at each hierarchical level are made up of different items.

genre. For the team that incorporates the TAP-based framework (Team A), the tactics and strategies are selected and adapted according to the methodology as depicted in this section. For the opposing team (Team B), the tactical behavior is randomly selected between the 3 types shown. For each experimental setup, one of the 5 strategies is chosen and fixed for Team B. Hence, Team B portrays a team scripted with a proper strategy and at the same time having some variance in their tactical behavior. At the start of each experimental setup, Team A has their tactical and strategic personalities randomly initialized. A separate experimental setup is also performed whereby Team B chooses a random strategy at each iteration.

The main results are as shown in Figures 12 and 13. The graphs show the reward value plotted against the number of iterations. All the agents' hit points as well as the bases' hit points are included in the calculation of the reward as depicted in (11). A positive reward means Team A has won the game, and the larger it is, the larger the margin of success (higher performance), and vice versa.

As it can be seen, all the experiments eventually converge to a stable state where Team A constantly wins. In cases where Team A randomly starts with a poor mix of tactics and strategies (Sets 1, 2, and 3), it loses first and tries out each of the other approaches and eventually finds a winning strategy which is reinforced and constantly applied. In other cases, Team A starts with a relatively good strategy (Sets 4 and 5) which is also reinforced and constantly utilized to win the game. For experiment Set 2, it can be seen that the opponent's strategy (all Team B agents attacking the base at once) is a harder one to beat, and the positive rewards are small in value. Nevertheless, Team A still finds the best approach in the end. In experiment Set 6, Team A also manages to find a winning strategy even though Team B randomly chooses a strategy at every round. After around 400 iterations, there is a spike down probably due to the randomness, but Team A still manages to recover after another 50 iterations. In general, it can be observed that the adaptive hierarchical framework enables the agent to constantly perform better than teams with fixed approaches.

To determine the significance of the results, a single-sample one-tailed Student's t-test [16] is performed on the results. An RTS game is a team-versus-team format whereby each result is a single set of reward values that represent the performance of the TAP-based team against the scripted team. Hence, a single-sample t-test is used. The null hypotheses are that there are no differences between the performance of the teams (no difference with an outcome with mean 0). The alternative hypotheses to be achieved are that the TAP-based teams perform better than the scripted team. Hence, a one-tailed t-test is used here.

The results of the t-tests are as shown in Table 3 where it can be seen that the P-values are mostly lower than .01, which implies that it is highly unlikely that the results have occurred by chance in the majority of the cases. The P-values in the left column are computed considering the full number of iterations from the start. Here, the flanking and random setups have the P-value under .01, which concludes that the results are hardly by chance. However, the all defense has a P-value of .077, meaning that this ambiguity chance is higher, but the significance of the result is still well above 90%. The hunting attack and attack + defense setups seems to have very poor significance with P-values well above .5. As will be seen in the next paragraph, this is attributed to the fact that the learning process prior to convergence is also considered.

The P-values in the right column in Table 3 are computed starting from iteration 250. This right column represents the values after the learning framework has converged (it can be seen from the graphs in Figures 12 and 13 that the framework converges around iteration 250). It can be seen that after convergence, the P-values are all well below .01, hence, it can be concluded that the results are statistically significant at the 1% level. This refutes the insignificance deductions as discussed in the last paragraph.

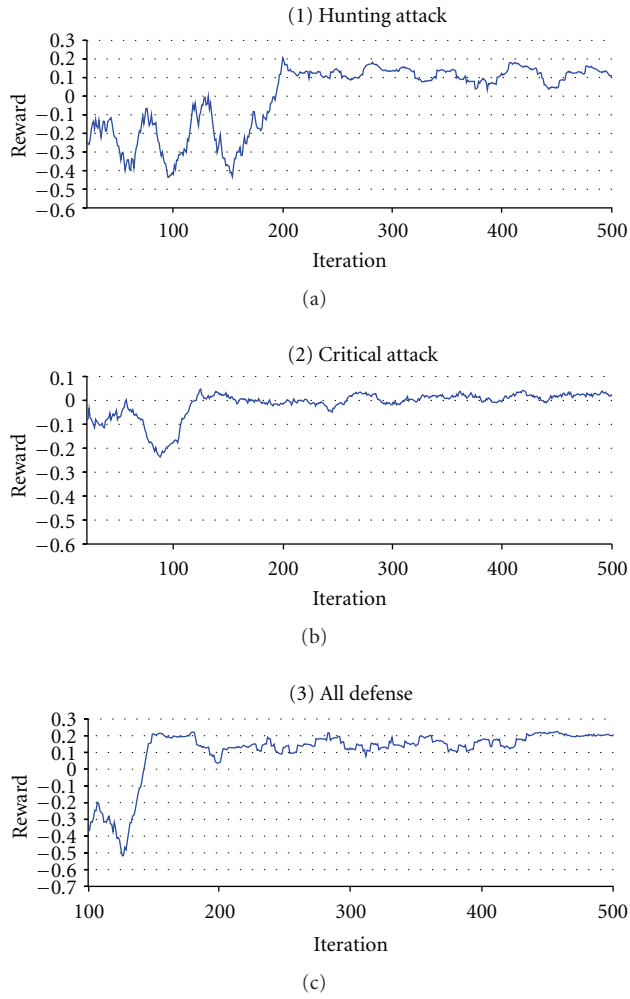

Figure 12: Generality test results for the SAP hierarchical learning framework: experiment sets 1, 2, and 3. The first three of six sets of experimental setups where each setup consists of Team B having a different fixed strategy. Each experiment set has the reward value plotted against the number of iterations. Note that in order to maximize visibility, the scales are not uniform across all graphs.

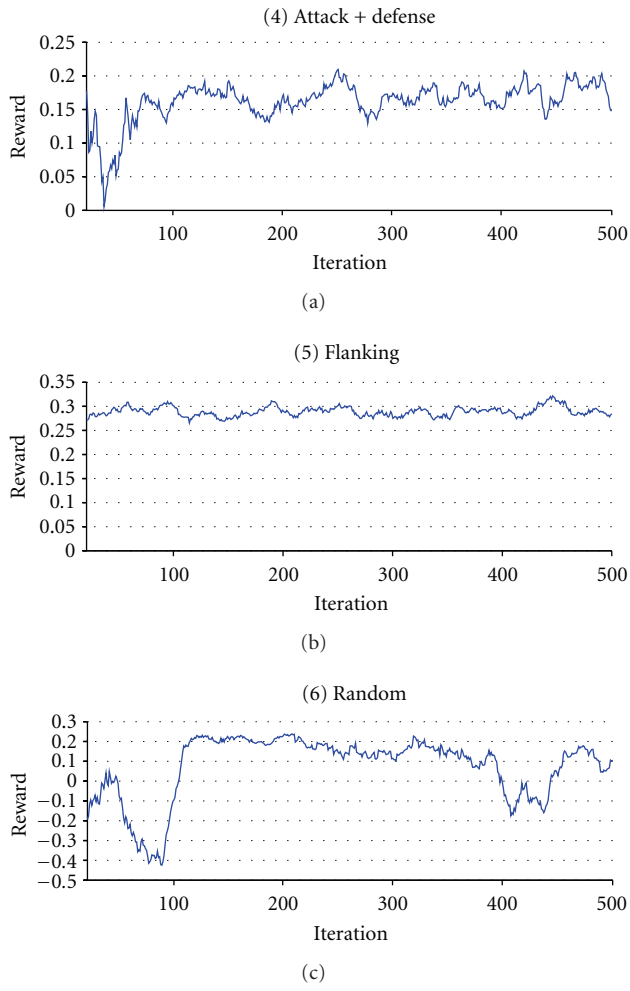

Figure 13: Generality test results for the SAP hierarchical learning framework: experiment sets 4, 5, and 6. The last three of six sets of experimental setups where each setup consists of Team B having a different fixed strategy. The exception is in the 6th set whereby Team B chooses a random strategy in each iteration. Each experiment set has the reward value plotted against the number of iterations. A single point on the graph represents a full game episode. Note that in order to maximize visibility, the scales are not uniform across all graphs.

(b) Scalability Test. To evaluate the scalability of the TAP-based framework, a set of experiments is performed with an increasingly large number of agents. The total time needed for all the agents to complete decision making at each reevaluation step is recorded and averaged over 500 runs for each experiment set.

The scalability results are tabulated as shown in Table 4. To investigate the growth factor, the corresponding graph is plotted in Figure 14. Although the time required is increasing in a roughly linear fashion with a factor of 0.0001 as per agent (taking the rough gradient from the graph in Figure 14), it still only takes slightly more than one millisecond for around 100 agents, which is rather fast. This means the framework can be implemented in modern games with a large number of intelligent agents.

5. Temporal Links in TAP

Advancing from the fundamental TAP model concept, this section introduces a temporal notion to the model to further improve its generality in more game genres (like an RPG game). To interpret other agents behaviors, capturing sequential information is essential, apart from 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. Sections 3.1 and 4 ignore this temporal information which seems to possess two problems. The first is a lack of descriptive power which might deter adaptation performance. The second problem is that agents

Table 3: Generality test results for the SAP hierarchical learning framework: tables of mean and t-test P-values.

	Mean values	
	Start from iteration 0	Start from iteration 250
Hunting attack	−0.002	0.118
Critical attack	−0.015	0.014
All defense	0.017	0.165
Attack + defense	0.162	0.172
Flanking	0.289	0.289
Random	0.074	0.090
	t-test P-values	
	Start from iteration 0	Start from iteration 250
Hunting attack	.607	<.010
Critical attack	.999	<.010
All defense	.077	<.010
Attack + defense	<.010	<.010
Flanking	<.010	<.010
Random	<.010	<.010

Table 4: Scalability test results for the SAP hierarchical learning framework: table of average decision making times.

Number of agents	0	9	21	51	101
Average decision time (s)	0	0.00093	0.00362	0.00471	0.01185

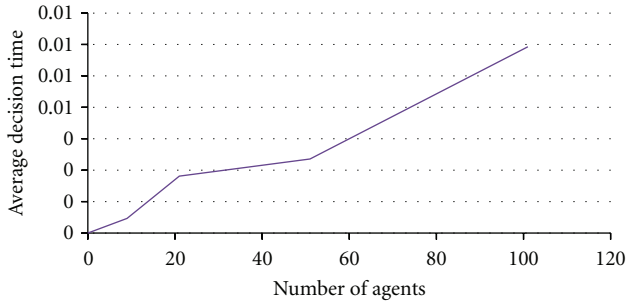

Figure 14: Scalability test results for the SAP hierarchical learning framework: plot of average decision time versus number of agents. The average time required for decision making is plotted against the number of agents. The decision time increases roughly linearly with the number of agents.

might produce erratic behavior when actions of a different genre are added to their adaptable set.

Hence, an alternative reformation of the model is performed by including directed edges between each action as shown in Figure 15. Instead of placing weights on each action, the weights are placed on each edge to denote a preference of that transition. Note that Figure 15 is only illustrative and does not show all the links for viewing clarity. Formally, if $\mathcal{P}$ is the set of all TAPs, the TAP, $P_k \in \mathcal{P}$, of an agent k, is a function that assigns a value, W, to each action pair

$$P_k : A \times A \longrightarrow W, \tag{12}$$

where A is the set of all allowable actions. Now, the model captures preferences of more descriptive tactical patterns rather than only previous action preferences. As explained, this enables more action genres to be adaptable and improves adaptation performance, which will be shown in the experiments. By including temporal knowledge in the model, it is also hoped that the flexibility and accuracy in interagent adaptation will be improved. In a nutshell, the temporal TAP aims to improve adaptation performance and accuracy.

When coupled with the TAP-based adaptation as described in Section 3, the learning cycle (adaptation, execution, and reinforcement) is the same as that shown in Figure 2. The only differences are in the inputs and outputs of the adapter in the adaptation stage. If the adapter is implemented as a feed-forward neural network as that shown in Figure 5, instead of having the neurons as the values on each action (node), now the neurons represent the values on each transition (directed edge).

Similar to the fundamental TAP concept, the adaptation process in the temporal TAP will be decoupled from the action-planning process. After the adaptation process, transition weights will be generated to produce an action-transition graph as shown in the right-hand diagram in Figure 15. After that, a separate action-selection mechanism will be used to generate the actual behaviors (action sequences). The specific implementation is again not the main focus of this paper but the current mechanism is a softmax selection probabilistically weighted by all the outgoing transitions at each action.

5.1. Evaluation. To evaluate the temporal TAP-based framework, a typical battle scenario is created in an RPG game scenario. An RPG game is chosen as each game character possesses a good variety of different action genres which can

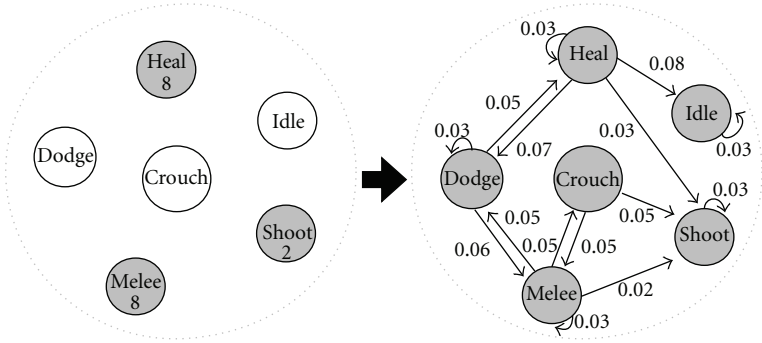

Figure 15: An example, TAP based on temporal links. In the new TAP model (right), a weighted topology depicting the preferential sequence of actions is added to the previous personality model (left).

effectively evaluate the enhancement claims of the temporal TAP model. A screenshot of the game environment is as shown in Figure 16.

5.1.1. Experimental Methodology. The experiments in this section aim to further complement the results shown in Sections 3.1 and 6, so as to reinforce two tests, namely, the

(1) generality,

(2) improvement,

claims of the TAP-based adaptation framework. The generality test here measures the effectiveness of the framework to cater to yet another game genre (which is RPGs). Improvement is measured here to test how well the temporal enhancements improve the TAP framework. Once again, robustness also needs to be measured to show consistency of the results. The sections that follow describe the game setups that aim to fulfil these test goals.

5.1.2. Experimental Setup. The game is modeled to mimic the complexity and randomness of a modern game; hence, different action genres that are vastly different in nature are created. Each agent can be one of 3 classes of characters typically found in RPG games, and each can possesses a possibly different AI controller. Every agent possesses the basic actions *idle* and *dodge*, but each character class possesse different personal attributes and actions like to *slash* and *shout* for the warrior and to *heal* and *mind blast* for the priest.

Various aspects of uncertainty found in a typical RPG game are synthesized so as to show practical applicability of this temporal algorithm. All class-specific actions have a certain chance of whether it will succeed, miss, or inflict critical (double) damage. The amount of damage or healing inflicted is calculated based on the class attributes, weapon attributes, and character attributes. In addition, all attacks have a range and cool-down time as well.

5.1.3. Experiments and Results. The experiments devised aim to test the generality and effectiveness criterions as described above. Robustness is an accompanying criterion that is maintained throughout the tests.

(a) Generality Test. Here another basic proof of concept of the TAP-based framework is demonstrated in a new RPG game genre. An RPG game consists of complex game environments with a larger variety of different action genres as compared to the FPS or RTS game. The game is set up as a battle between two teams with the same setup. Each team consists of 1 PC and 5 NPCs which are made up of 2 warriors, 2 mages, and 2 priests. The first team will be using the enhanced TAP-based framework (TTAP team), whereas the other team uses synthetically scripted behavior (scripted team) in a series of runs.

The results are as shown in Figure 17. The S value on the y-axis is basically a score value that is the normalized difference between the hitpoints of the TTAP and scripted teams. It is a value between -0.5 and 0.5, with the former that is TTAP team has been wiped out with all scripted team agents having full hitpoints, and vice versa. It can be seen that the score is improving as the game proceeds that is the adaptation does occur to make the agents perform better.

A single-sample one-tailed Student's t-test is performed on the results to establish significance. As the results are based on a single-reward value that represents the performance of the TAP-based team against the scripted team, a single-sample t-test is used here. The null hypothesis is that there is no difference between the performance of the teams (no difference with an outcome with mean 0). The alternative hypothesis to be achieved is that TTAP team performs better than scripted team. Hence a one-tailed test is used. As shown in Table 5, the P-value obtained is less than .01, which means that the claim is significant.

(b) Improvement Test. Here the experiments are set up to test whether the improvement made to the temporal TAP model is effective, in comparison with the fundamental TAP model (as depicted in Section 3.1). To make this comparison, the previous game scenario is duplicated. As with the experiment in Section 3.1, 6 agents are used and the actions *melee*, *shoot,* and *heal* recreated and made adaptable. The team using the temporal TAP-based framework (TTAP team) is then set to pit against a team using the old framework (TAP Team), both with untrained adapters at first. The results are as shown in Figure 18.

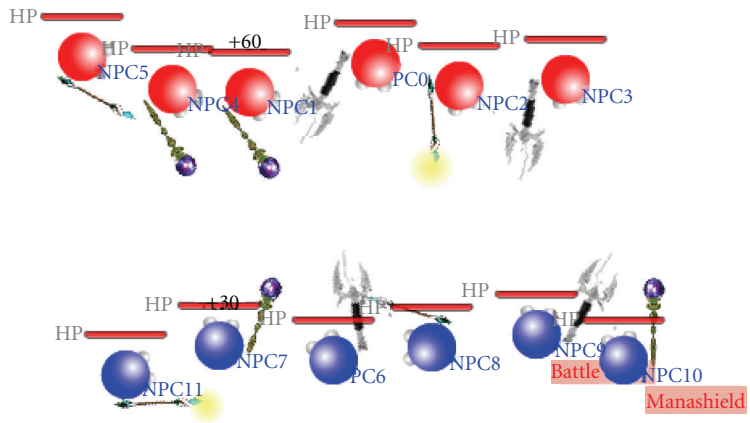

Figure 16: A screenshot of the experimental game scenario used in evaluating the TAP based on temporal links. Two teams of agents consisting of warriors, mages and priests pit against each other. They can be differentiated by the weapons they carry.

Figure 17: Generality test results for the temporal TAP adaptation framework: plot of S against number of iterations. A single point on the graph represents a full game episode. S depicts the performance difference in TTAP team versus scripted team.

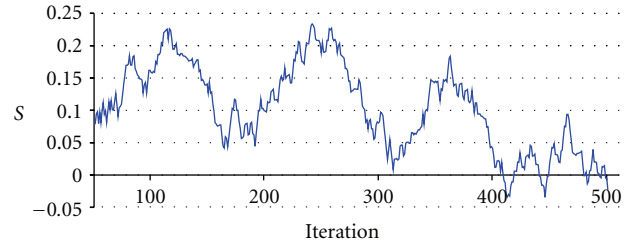

Figure 18: Experimental results of the temporal TAP adaptation framework over the old TAP framework: improvement test. plot of S against number of iterations. A single point on the graph represents a full game episode. S depicts the performance difference in TTAP team versus TAP team.

Table 5: Generality and improvement test results of the temporal TAP adaptation framework: table of t-test P-values.

	TTAP versus scripted	TTAP versus TAP
t-test P-value	<.010	<.010

As shown in the graph, TTAP Team consistently performs better than TAP Teams but as it gets closer to 500 rounds in the performance seem, to converge to a draw state. This is probably due to the fact that TAP team has an AI of its own that also adapts as the game proceeds. In general the line stays above 0 majority of the time with an average of 0.092, which means TTAP team wins more as a whole. This shows that the new temporal TAP framework is performing just slightly better, and seems to converge to a draw.

Similar to the previous generality experiments, the single-sample one-tailed Student's t-test is performed on the effectiveness results to establish significance. The null hypothesis is that there is no difference between the performance of the teams (no difference with an outcome with mean 0). The alternative hypothesis to be achieved is that TTAP team performs better than TAP Team. As shown in Table 5, the P-value obtained is less than .01, which means that the probability that the results are due to chance is also low here.

6. Conclusion

In this paper, the TAP representation has been described along with its use in an adaptation framework. In its basic form, it is an agent personality representation that captures a sufficient historical knowledge to allow inter-agent adaptation. One major advantage is that the knowledge represented is independent of expert knowledge. It also has the advantage of being a uniform representation that does not distinguish itself between PCs and NPCs, making adaptation more generic (not only player-specific adaptation) as required by the application domain. In growingly popular massive multiplayer online games (MMOGs), for example, *Guild Wars* (http://www.guildwars.com/), 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.

Various methodology alternatives and enhancements are also shown and evaluated to show the effectiveness, versatility, generality, scalability, and robustness claims of the TAP-based adaptation framework. The evaluations form a wide range of common modern game genres, whereby the TAP

framework has shown to enable plausible interagent adaptive behavior in various forms. This includes the fundamental TAP evaluated in an FPS, the strategic SAP evaluated in an RTS, and the temporal TAP evaluated in an RPG. In all, the TAP model and framework is shown to be a credible formulation to enable adaptation from one agent to another. However, there are important issues that should be discussed, all of which can lead to further evaluations in future work. These are highlighted in the final paragraphs that follow.

One issue is that most of the experiments involve comparing against a scripted adversary. It might be argued that the results might be dependant on how well the scripts are made. However, it should be pointed out that these "scripted" adversaries are actually carefully crafted such that their behaviors represent the nonadaptive versions of the same framework. They are not actually scripted in the sense that they are given specific handcrafted sequences of actions. For example, the adversaries in the main TAP experiment performed in Section 3.1 contain the exact same TAP model and the same action selection mechanism. The only difference is that the TAP model is fixed and nonadaptive. Hence, the scripting behavioral space is the same action space in which the adaptive agents utilize. The versatility tests also make sure that the conclusions can be made across varying values of each "scripted" action value in the TAP model. These measures ensure that the experiments are not based on a single biased version of a handcrafted script. Nevertheless, larger scale experiments involving a larger number of possible action value combinations can be performed in the future to further support the conclusions.

Next, although using the action transitions seems to be conceptually better than the actions themselves, experiments have shown otherwise. The current conclusion is that when the temporal TAP is being used in the adaptation process, it suffers from the curse of dimensionality as the number of weights to be adapted increases at an alarming rate. This is due to the fact that the number of weights to be adapted is directly proportional to the number of agents and the number of actions each agent possesses. More clearly, the number of adaptive weights (number of inputs to the neural network) is $N \times k^2$, where N is the total number of agents and k the number of actions each agent has. A single increase in N means a k^2 increase in the number of weights to be adapted. Hence, this makes the framework somewhat infeasible for large team sizes. Also, there exists a limitation in terms of topology design. If the designer decides on placing self-loops on actions, then he must be prepared to cater for the possibility that the agent might learn to keep doing the same action over and over again, which might look erratic to the player. Moreover, the improvements over the original nontemporal TAP framework is not that great (as shown in Figure 18), so a stronger case needs to be established before this increase in complexity can be justified. As of now, it seems like the original simple TAP shown in Section 1 is enough to perform the job, and that the increase in complexity seems redundant. Possibly more kinds of experiments on other games can be run here to further strengthen this claim. Related to scalability, a complete set of experiments can also be devised in future work to estimate the time complexities. Some initial experiments (like the scalability tests in Section 4) have been performed, but much more can be done in order to obtain estimates on actual bounds on the number of acceptable weights (and hence actions and agents) without sacrificing model efficiency.

In the evaluations, some of the results also show that a relatively large number of iterations are needed before learning can converge. In the experiments shown in this chapter, the TAP model has been utilized in model-free frameworks. This means that the NPCs will initially look stupid and try all sorts of erratic actions before finally appearing intelligent when the learning evolves substantially. This might not be that feasible in modern games where the NPCs do not have such liberty. A possible resolution is to introduce some model-based elements in the architecture in future work.

In the TAP adaptation framework, the core of the adaptation lies in the implementation of the adaptor. As mentioned, the work in this paper focuses on the framework and not the implementation, hence, the current implementation limits itself to a single type of artificial neural network as the implementation of the adaptor. Perhaps a possible future work is to focus on the implementation of other network configurations as well as other supervised, semisupervised and nonsupervised methods so that they can be compared empirically to determine the feasible qualities of each.

Lastly, it seems that variants to the fundamental TAP architecture needs to be created when applying it to different game genres. In the RTS game application, a SAP model was introduced as an extension to cater to the hierarchical team-based game play. Similarly, in the RPG game application, a temporal version was introduced to cater to the increase in action genres. Hence, a major step in future work is to generalize the formalization of the model so as to cater to a broader genre of game play. A first step might be to combine the notion of tactical and strategic actions as well as perhaps allow the option of toggling action and transition weights. In view of the longer term, possibly a clearer definition of the generic computer game space needs to be considered. In all, this presents a plethora of opportunities to extend this work to greater heights.

References

[1] W. van der Sterren, "Being a better buddy: interpreting the player's behavior," in *AI Game Programming Wisdom 3*, pp. 479–494, Charles River Media, Cambridge, Mass, USA, 1st edition, 2006.

[2] M. Sharma, M. Mehta, S. Ontan, and A. Ram, "Player modeling evaluation for interactive fiction," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference Workshop on Optimizing Player Satisfaction*, 2007.

[3] D. Thue, V. Bulitko, M. Spetch, and E. Wasylishen, "Interactive storytelling: a player modelling approach," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference*, pp. 43–48, 2007.

[4] H. Barber and D. Kudenko, "Dynamic generation of dilemma-based interactive narratives," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference*, pp. 2–7, 2007.

[5] M. S. El-Nasr, "Interaction, narrative, and drama: creating an adaptive interactive narrative using performance arts theories," *Interaction Studies*, vol. 8, no. 2, pp. 209–240, 2007.

[6] J. Donkers and P. Spronck, "Preference-based player modeling," in *AI Game Programming Wisdom 3*, pp. 647–659, Charles River Media, Cambridge, Mass, USA, 1st edition, 2006.

[7] D. Thue and V. Bulitko, "Modeling goal-directed players in digital games," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference*, pp. 285–298, Marina del Rey, Calif, USA, 2006.

[8] G. N. Yannakakis and M. Maragoudakis, "Player modeling impact on players entertainment in computer games," in *User Modeling*, vol. 3538 of *Lecture Notes in Computer Science*, pp. 74–78, Springer, Berlin, Germany, 2005.

[9] C. T. Tan and H. Cheng, "Personality-based adaptation for teamwork in game agents," in *Proceedings of the 3rd Conference on Artificial Intelligence and Interactive Digital Entertainment*, pp. 37–42, Palo Alto, Calif, USA, 2007.

[10] C. T. Tan and H. Cheng, "A combined tactical and strategic hierarchical learning framework in multi-agent games," in *Proceedings of the ACM SIGGRAPH Sandbox Symposium on Videogames*, Los Angeles, Calif, USA, 2008.

[11] C. T. Tan and H. Cheng, "TAP: an effective personality representation for inter-agent adaptation in games," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference*, Los Angeles, Calif, USA, 2008.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pp. 318–362, The MIT Press, Cambridge, Mass, USA, 1986.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, Mass, USA, 1998.

[14] I. Szita, M. Ponsen, and P. Spronck, "Effective and diverse adaptive game AI," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 1, pp. 16–27, 2009.

[15] P. Spronck, "A model for reliable adaptive game intelligence," in *Proceedings of the International Joint Conference on Artificial Intelligence Workshop on Reasoning, Representation, and Learning in Computer Games*, pp. 95–100, 2005.

[16] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, UK, 2nd edition, 1992.

[17] F. Sailera, M. Buro, and M. Lanctot, "Adversarial planning through strategy simulation," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, Honolulu, Hawaii, USA, April 2007.