A Combined Tactical and Strategic Hierarchical Learning Framework in Multi-agent Games

Chek Tien Tan*
Ho-lun Cheng†
National University of Singapore

Abstract

This paper presents a novel approach to modeling a generic cognitive framework in game agents to provide tactical behavior generation as well as strategic decision making in modern multi-agent computer games. The core of our framework consists of two characterization concepts we term as the tactical and strategic personalities, embedded in each game agent. Tactical actions and strategic plans are generated according to the weights defined in their respective personalities. The personalities are constantly improved as the game proceeds by a learning process based on reinforcement learning. Also, the strategies selected at each level of the agents' command hierarchy affect the personalities and hence the decisions of other agents. The learning system improves performance of the game agents in combat and is decoupled from the action selection mechanism to ensure speed. The variability in tactical behavior and decentralized strategic decision making improves realism and increases entertainment value. Our framework is implemented in a real game scenario as an experiment and shown to outperform various scripted opponent team tactics and strategies, as well as one with a randomly varying strategy.

CR Categories: I.2.1 [Artificial Intelligence]: Applications and Expert Systems—Games I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—[Intelligent agents, Multiagent systems]

Keywords: game artificial intelligence, game agent architecture, multi-agent cooperation, tactical behavior, strategic planning, learning

1 Introduction

In the domain of modern game Artificial Intelligence (AI), recent years have seen much work being performed in the area of synthesizing intelligent behaviors in game agents [Hussain and Vidaver 2006; Horswill and Zubek 1999; Geramifard et al. 2006; Khoo and Dunham 2002; McDonald et al. 2006; Spronck et al. 2006; Sweetser 2005; Tan and Cheng 2007; White and Brogan 2006]. 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). Agent behavioral planning can be classified as either *tactical* or *strategic*. "Tactical" refers to low level planning of primitive actions like "shoot", "attack" and "heal" by each agent. "Strategic" encompasses high level planning like deciding on whether to execute a flanking attack or a mass frontal attack as a team. In short, tactical decisions are short-term and individually-based whereas strategic decisions are long-term and team-based.

This paper describes our efforts at modeling a generic cognitive framework in game agents that enables planning in both tactical and strategic aspects. We find a combined planning approach lacking in current work, especially in complex modern game environments. It builds upon an earlier work in tactical behavior generation [Tan and Cheng 2007] to include strategic planning and the integrated framework has been implemented with positive results. Our framework is applicable for the different modern game genres as described above.

1.1 Motivation

Although there are large amounts of research being done in the area of agent planning, current work are more suited for chess-like board games than for modern games. Most work are based on classical theories that focus on the intelligent agent problem in a generic sense. In doing so, they are often too complicated and impractical for use in commercial games, and are focused on theory rather than gameplay. In board games, algorithms like alpha-beta game-tree search can exhaustively enumerate future states to evaluate action outcomes because of the tactical nature and the branching factor is relatively small. Modern game developers are also reluctant to use state of the art intelligent agent algorithms due to the fear of game agents learning to produce strange or even erratic behavior that is totally unexpected. Modern game developers require simplicity whilst achieving speed and effectiveness, hence simple and comprehendible AI like scripting and finite state machines (FSMs) [Orkin 2004b] are still dominant in today's game AI. Whilst we acknowledge that generic and theoretical advancements are of utmost importance in the general AI community as a whole, we also recognize that there is a concurrent need for studies that are implementable in (or can be combined with) the AI of modern commercial games.

Automatically devising a good playing strategy is much more complex in modern games than in classical board games for a variety of reasons. Modern games include a variety of agents, each possessing a vast number of diverse actions, whereas board game pieces are mainly defined by only the type of movement they can make. Additionally, all agents in modern games are able to act simultaneously in real time, in contrast with turn-based board games. Considering the world space, modern game worlds are enormous and continuous compared to the 64 discrete positions of chess or even the 361 positions in Go. These factors make the search space exceptionally larger and it is impossible to exhaust all searches within a reasonable time interval for game playing to be smooth. Take for example if we were to model an RPG in a classical Partially Observable Markov Decision Process (POMDP) [Foka and Trahanias 2007], it would be almost impossible to obtain a working policy without tremendous amounts of heuristics and abstractions, due to the enormous number of belief states. Current successful algorithms solve

*e-mail: tanchekt@comp.nus.edu.sg
†hcheng@comp.nus.edu.sg

to the complexity level of simple poker games [Oliehoek 2005], which is a far fetch from the complexity we get in modern commercial games [Sailera et al. 2007]. In addition to the performance aspect, modern game AI also needs to take into account the entertainment value for human players. As such, research in this area is promising, challenging and interesting.

Given that modern games involve teams of multiple agents working together with a common goal, there is a need for team-based cooperative planning. In FPS games like Battlefield 2142 [Electronic Arts 2006], a player can form teams with non-player characters(NPCs) to play against computer controlled opponent teams in military-style combat. This is often the case in RPGs like Guild Wars [NCsoft 2006] when a player teams up with NPCs for missions, and opponents are mostly organized in large packs. Similarly in RTS games like Warcraft 3 [Blizzard Entertainment 2006], the computer opponent manages a whole team of units and resources. The AI in these teams is largely scripted and often performs predictably, making the games un-challenging and dull when the player figures out the patterns. Other than multi-agent pathfinding, research in multi-agent coordination is minimal in modern games. Current work mainly focuses on tactical behaviors and not enough attention has been given to strategic decision making. More details are given in the section on *Related Work*.

1.2 Aim

The goal of this paper is to model an integrated framework to enable both tactical and strategic planning capabilities for agents in a team-based environment. To reinforce the cooperative capabilities, we establish a communication and command-passing system amongst the agents. In general, our framework serves as a generic representation of the cognitive ability of an agent in the game. We also consider the factors of performance, speed and entertainment value, three of the most important requirements for game AI to be useful practically [Spronck et al. 2006]. Besides synthesizing immediate behaviors in normal agents, our work provides strategic capabilities to agents with leadership abilities. In the simplest case, each team has a single commander. In more sophisticated games, there may be a hierarchy of commander agents and our framework can be used to serve the different management requirements at each hierarchy of the opposing teams. For example, each team of units the player or computer agent groups up can be incorporated with our framework and automatically function on its own after being given a high level task. This eases the micro-management imposed on the player in RTS games to allow for more strategic gameplay. The framework is also generic enough to be applied other game genres like RPG, FPS and sports games.

1.3 Our Approach

First, each team is organized into a hierarchical tree structure where each agent is a node in the tree and each parent node is able to pass a command to its children. A tree-structured team is better than a graph-like one in terms of performance because the acyclic manner of command-passing facilitates convergence in the learning process. Allowing the team to form a cyclic structure may result in infinite loops as agents affect each other in a back and forth manner. Moreover, a tree structure better reflects real-life combat establishments like in the military.

Each agent in the hierarchy possesses a *tactical personality* drawn from [Tan and Cheng 2007] and a *strategic personality*. Tactical and strategic planning is based on the weights defined in the personalities and updated based on reinforcement learning concepts [Sutton and Barto 1998] as the game proceeds. In addition, we improve multi-agent coordination by introducing a hierarchical command-passing mechanism into the framework. With these methods we aim to:

- enable tactical and strategic adaptivity to the game environment,
- conceive effective and interesting character behaviors, and
- enable a hierarchical command-passing system between agents.

Adaptive learning improves agent performance as the game proceeds whilst behavioral variability makes agents more interesting hence improving the entertainment value. Moreover, since our tactical and strategic selection processes only depend on numeric weights for decision making, the process is very quick in-game. Hence our framework satisfies the three requirements in our goal-speed, performance and entertainment value.

In the remaining sections of this paper, we first review literature related to our work. Then we describe the system architecture in detail. Thereafter, we show our experimental results as empirical proof of our concept, along with an analysis of the results. Lastly, we conclude the paper together with our plans for future work.

2 Related Work

Synthesizing intelligent game agents has been one of the core areas in game AI research [Horswill and Zubek 1999; Hussain and Vidaver 2006; Geramifard et al. 2006; Khoo and Dunham 2002; McDonald et al. 2006; Sweetser 2005; White and Brogan 2006], but most studies ignore the strategic aspect. [Khoo and Dunham 2002] devised a stateless FSM system for game agents to efficiently react to environmental changes. [Sweetser 2005] combines influence maps with cellula automata techniques for game agent decision making. Others [Horswill and Zubek 1999; Hussain and 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. [White and Brogan 2006] presented a method that employs a self-organizing Kohonen map coupled with reinforcement learning to produce effective gameplay in multiple agents in RoboCup simulated soccer. These work mainly targeted at devising tactical behavior architectures. Additionally, these studies make use of decision making systems that need to perform heavy computations before each action taken by the agents, with frequent updates to the knowledge in the system which would likely cause disruptions to the actual gameplay. In [Tan and Cheng 2007], they have devised a natural and convenient way to represent actions in a weighted set, in which they call a personality model. They have also alleviated the problem by decoupling the learning system from the action selection mechanism. The intensive learning process only takes place at convenient intervals in the game (like scenario or map transitions in the game) whereas a fast and simple action selection can take place as often as needed. However, their work only plan for primitive action sequences. The work in this paper largely makes use of their personality representation to cater for a combined tactical and strategic planning architecture.

A number of studies also focused on specific areas of planning like pathfinding [Geramifard et al. 2006; Silver 2005] and tactical positioning [Remco Straatman and van der Sterren 2006; Christian J. Darken 2006]. Recent pathfinding work [Geramifard et al. 2006; Silver 2005] concentrated on cooperative, multi-agent pathfinding primarily in RTS games. [Remco Straatman and van der Sterren 2006] provides a simple map representation to aid the game agent in evaluating tactical positions, whilst [Christian J. Darken 2006] combines level annotation with sensor grid algorithm to allow the

agent to dynamically find cover. These studies aim at issues of location and do not consider the actual actions that need to be done after the agents arrive at the location.

A prominent advancement in adaptive agent behavior was the introduction of dynamic scripting [Spronck et al. 2006] where adaptive behavior was defined by choosing from a set of scripted actions. They have shown that dynamic scripting using a learning system that is adapted from reinforcement learning performs very well against computer opponents. Moreover, they balanced the practicality of scripting with the learning capability of reinforcement learning. In this paper we adopt a similar approach for the learning system. However, while their work focuses on defining adaptive tactical behavior in individual agents, our framework includes strategic decision making as well as a hierarchical command-passing mechanism for multiple agents. Also, our planning system reaches into the level of individual actions, whilst theirs are based on scripts. This adds flexibility in the framework.

In terms of strategic AI in modern games, [Sailera et al. 2007] devised a method that searches through all scenarios to conclude a winning strategy for an RTS game environment. They simulate strategy pairs in a fast-forwarded simulation to determine which strategy would give the best winning chance. Though the simulation is supposedly fast-forwarded according to their algorithm, the obvious setback is still that it is a tedious search method that requires simulating each scenario all the way to the end to arrive at a conclusion. In contrast, the strategy selection mechanism in this paper is based on a machine learning approach which is much faster, albeit at the expense of having a learning period.

In summary, current work in modern games focuses on tactical AI or strategic AI in isolation. In modern games where large agent teams exist, an AI that encompasses both aspects would prove very useful.

3 Agent Cognitive Framework

Every agent in the game possesses the cognitive framework as shown in Figure 1. 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.

3.1 Agent Personality

The concept of agent personality was first introduced in [Tan and Cheng 2007] and an illustration can be seen in the top diagram in Figure 2. In their work, when an action selection mechanism is applied, the weight determines the chance of choosing an an action in view of other simultaneous adaptable actions. To reiterate the definition using a similar terminology, if $P'_{tactical}$ is the set of all tactical personalities, the tactical personality of an agent, $P_{tactical} \in P'_{tactical}$, is a function that assigns a weight to each adaptable action.

$$P_{tactical} : A \rightarrow W, \tag{1}$$

where $W \in [W_{min}, W_{max}]$ is the set of all weights with W_{min} and W_{max} being the bounds for the weight values, and A is the set of all adaptable actions.

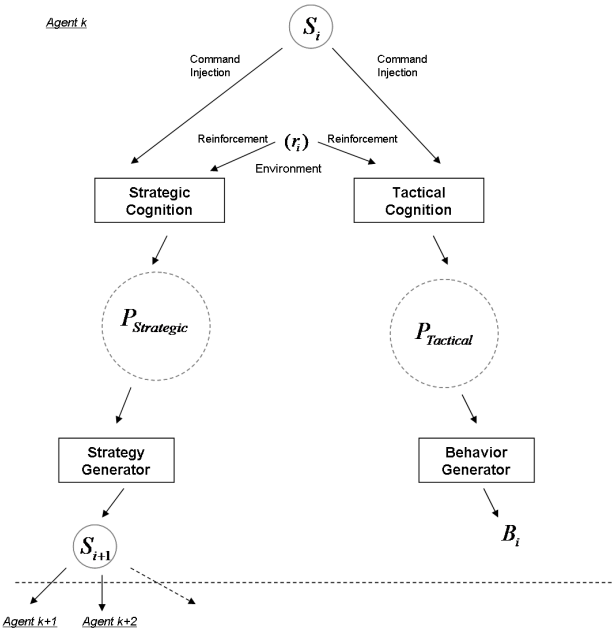

Figure 1: *Agent Cognitive Framework: The tactical behavior and its strategy for the agents in the current hierarchical level is governed by the strategy being generated from the commander agents in the previous hierarchical level. Central to the framework are the adaptive tactical and strategic personalities which control the selection process for the behaviors and strategies for each agent.*

In our new strategic personality (the bottom diagram in Figure 2), the primitive actions in the tactical personality are replaced with high level strategies. The commander has several strategies in mind, each of which is assigned a weight before the start of the decision making process. Similarly, if $P'_{strategic}$ is the set of all strategic personalities, the strategic personality of an agent, $P_{strategic} \in P'_{strategic}$, is a function that assigns a weight to each strategy which is determined by the learning framework as elaborated on in later sub-sections.

$$P_{strategic} : S \rightarrow W, \tag{2}$$

where S is the set of all strategies.

3.2 Behavior and Strategy Selection

Common to any online learning framework, we need to address the problem of exploration versus exploitation, namely, at each loop, the behavior and strategy generators either choose to exploit the currently learnt knowledge and choose an optimal item, or to choose a suboptimal item to create new learning instances and improve that item's assigned weight.

For the behavior generator, as described in [Tan and Cheng 2007], we adopt the standard ϵ-greedy algorithm [Sutton and Barto 1998]. 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 kind of rule [Sutton and Barto 1998] where 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

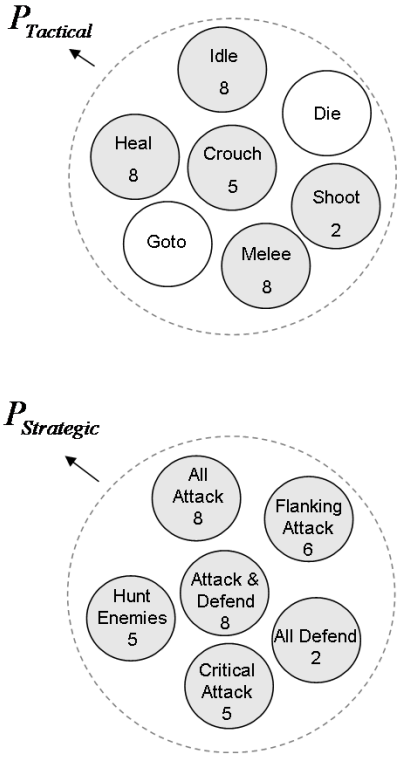

Figure 2: *An example tactical personality (top) and strategic personality (bottom): For the tactical personality, each action is tagged with a relative weight that can be evaluated into a probability of choosing it. The shaded ones are the adaptable actions whilst the unshaded ones are non-adaptable. The strategic personality is similarly defined.*

strategy, s_i, at time t is

$$Pr(s_t = s_i) = \frac{e^{w_i/\tau}}{\sum_{k=1}^{n} e^{w_k}/\tau}, \tag{3}$$

where n is the total number of strategies and τ is a temperature variable to control the greediness of the approach [Sutton and Barto 1998].

These straightforward selection methods ensure that behavior generation and strategy selection can be done very quickly without interrupting or overlapping with actual gameplay. 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 gameplay 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 we prefer a more constant performance over variability whereas tactically, variability is as important as constant performance (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 game or milestones (for example map transitions in an RPG or respawn times in an FPS). After each execution, a reward value, r_T, is generated via an error function

$$r_T = \sum_{o_k \in O} \gamma_k |H_T(o_k)| - \sum_{o'_k \in O'} \gamma'_k |H_T(o'_k)|, \tag{4}$$

for the particular strategy, i, is used before this reevaluation time, where 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 weightage of each type of unit, where $\sum_{\text{all k}} \gamma_k = 1$ and $\sum_{\text{all k}} \gamma'_k = 1$. This value, r_T, represents the environment factor as shown in Figure 2 which is passed on to the reinforcement process in the tactical and strategy cognitions.

3.3 Reinforcement and Command Injection

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) is affected. For each adaptable action, j, in the behavior sequence in use, the update function for its weight, w_j, is

$$w_j = w_j + \alpha(r_T - \overline{r_T}), \tag{5}$$

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 [Sutton and Barto 1998], 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

$$w_i = w_i + \beta(r_T - \overline{r_T}). \tag{6}$$

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, we can define

$$P'_{tactical} = C_{tactical}(P_{tactical}, s_{i-1}) \text{ and} \tag{7}$$

$$P'_{strategic} = C_{strategic}(P_{strategic}, s_{i-1}), \tag{8}$$

where $P'_{tactical}$ and $P'_{strategic}$ are the new personalities respectively. The functions $C_{tactical}$ and $C_{strategic}$ 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 paper, 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.

4 Empirical Evaluation and Discussion

In order to evaluate our framework in a real game environment, we implement a typical action game scenario built on the Truevision3D 6.5 game engine (a screenshot of the environment is as shown in Figure 3). While this scenario is close to that of an FPS, the framework can be applied to other games of different genres.

4.1 Game Mechanics

Our 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 4 with a top level commander directing multiple sub-teams, each with 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. A draw happens when both teams have no more agents alive but both bases are not destroyed.

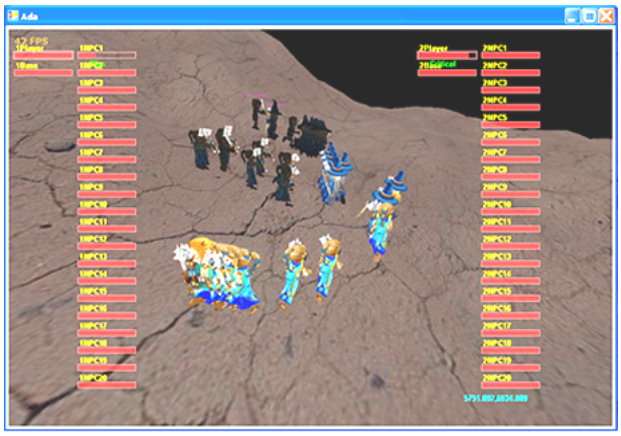

Figure 3: *Screenshot of Test Environment: Team A consists of the lighter colored characters (mainly on the bottom side of the figure) whilst Team B consists of the darker colored characters.*

The constituents that make up the tactical and strategic personalities are also shown in Figure 4. 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. The base acts as a turret that has a longer range than agents and can attack a single enemy at a time. Strategically, the strategies available to the commander agents of each team are

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 to and attack the enemy base.
5. **Attack & Defend**. Some ally agents would stay near the ally base whilst the rest would move towards and attack the enemy agents or enemy base.

For the team incorporated with our framework (Team A), the behavior and strategies are selected and adapted according to the techniques depicted in this paper. For the opposing team (Team B), the

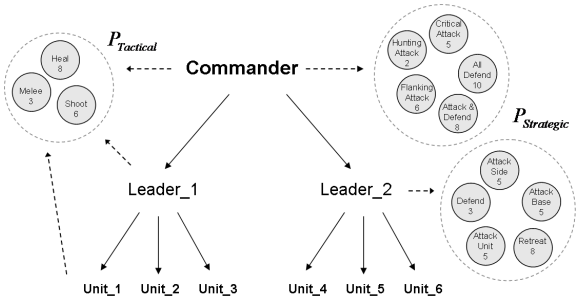

Figure 4: *Experimental Setup of Units' Hierarchy: Each team in the experiment consists of one commander, two sub-team leaders and 6 units. All agents have the same type of tactical personality (with different weights), but the strategic personalities at each hierarchical level are made up of different items.*

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. We also include an experimental setup whereby Team B chooses a random strategy at each iteration.

To evaluate the scalability of our framework, we have also ran the experiments 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.

4.2 Results and Discussion

The main results are shown in Figure 5 on the last page of this paper. The graphs show the reward value plotted against the number of iterations. We chose to include all the agents' hit-points as well as the bases' hit-points for the calculation of the reward as depicted in Equation 4. 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 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 4), 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 3 and 5) which is also reinforced and constantly utilized to win the game. For experiment Set 2 we can see 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. At 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, we can observe that our adaptive framework enables the agent to constantly perform better than teams with fixed approaches.

In Figure 6, the graph shows the decision making times needed for experimental runs involving different number of agents. Although the time required is increasing in a roughly linear fashion, it still only takes slightly more than one millisecond for 100 agents, which is rather fast. This shows that the framework can be implemented even for massive modern games with a large number of intelligent

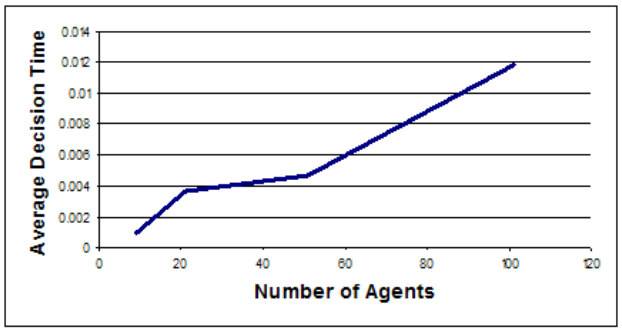

Figure 6: *Scalability Test Results: The average time required for decision making is plotted against the number of agents. The decision time increases roughly linear with the number of agents.*

agents.

5 Conclusion and Future Work

We have presented a generic cognitive multi-agent framework for both tactical and strategic planning which is shown to exhibit better performance against various scripted opponent team tactics and strategies, as well as one with a randomly varying strategy. Advancing from the work done by [Tan and Cheng 2007], we have introduced a new strategic personality concept as well as established a command-passing mechanism for team hierarchy. The combined personalities provide a means for adaptation both towards the feedback from the environment as well as the strategy command passed down from the previous hierarchical level. Our method also decouples action and strategy selection from the resource intensive learning process, hence reducing disruptions to actual gameplay. Moreover, the tactical flexibility in behavioral selection and decentralized strategic planning process introduce interesting and variable character behaviors, increasing the entertainment value for the player.

As a next step, we hope to investigate a generic system to represent the tactical and strategic cognitions. The system should ideally be able to automatically provide the weight differences to the personalities, given a new strategy. To do so, we need to formally define the strategy space. Also, as we did not evaluate the improvements we achieved in terms of entertainment value, we hope to do so in the future, using a user survey method to record the gaming experience of players for our analysis.

References

ANDRADE, G., RAMALHO, G., SANTANA, H., AND CORRUBLE, V. 2005. Automatic computer game balancing: A reinforcement learning approach. *In Proceedings of the Autonomous Agents And Multi Agent Systems Conference* (July), 1111,1112.

BLIZZARD ENTERTAINMENT, 2006. Warcraft III. Accessed April 12, 2006. Available via http://www.blizzard.com/war3/.

BLIZZARD, 2006. Diablo ii. Accessed April 12, 2006. Available via http://www.blizzard.com/diablo2/.

BLIZZARD, 2006. World of warcraft. Accessed April 12, 2006. Available via http://www.worldofwarcraft.com/.

BURO, M., 2007. ORTS - A Free Software RTS Game Engine. Accessed March 20, 2007. Available via http://www.cs.ualberta.ca/ mburo/orts/index.html.

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. *Findin Cover in Dynamic Environments*, first ed. Charles River Media, Hingham, Massachusetts.

CO-OP, S., 2006. Sven co-op. Accessed September 15, 2006. Available via http://www.svencoop.com/.

ELECTRONIC ARTS, 2006. Battlefield 2142. Accessed December 20, 2006. Available via http://battlefield.ea.com/battlefield/bf2142/.

EXLUNA, INC. 2002. *Entropy 3.1 Technical Reference*, January.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 15–22.

FOKA, A., AND TRAHANIAS, P. 2007. Real-time hierarchical POMDPs for autonomous robot navigation. *In Proceedings of Robotics and Autonomous Systems*, 561,571.

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*.

HUNICKE, R., AND CHAPMAN, V. 2004. AI for Dynamic Difficult Adjustment in Games. *In Proceedings of the Challenges in Game AI Workshop, Nineteenth National Conference on Artificial Intelligence*.

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* (July), 785,792.

JOBSON, D. J., RAHMAN, Z., AND WOODELL, G. A. 1995. Retinex image processing: Improved fidelity to direct visual observation. In *Proceedings of the IS&T Fourth Color Imaging Conference: Color Science, Systems, and Applications*, vol. 4, 124–125.

KARTCH, D. 2000. *Efficient Rendering and Compression for Full-Parallax Computer-Generated Holographic Stereograms*. PhD thesis, Cornell University.

KHOO, A., AND DUNHAM, G. 2002. Efficient, realistic npc control systems using behavior-based techniques. *In AAAI Technical Report*, 02–01.

LANDIS, H., 2002. Global illumination in production. ACM SIGGRAPH 2002 Course #16 Notes, July.

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., AND FULK, D. 2000. The digital michelangelo project. In *Proceedings of SIGGRAPH 2000*, ACM Press / ACM SIGGRAPH, New York, K. Akeley, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 131–144.

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* (June).

MICHAEL E. TIPPING, C. M. B. 1999. Probabilistic principal component analysis. *Technical Report NCRG/97/010, Neural Computing Research Grou* (September).

NCSOFT, 2006. Guild wars. Accessed April 12, 2006. Available via http://www.guildwars.com/.

OLIEHOEK, F. 2005. Game Theory and AI: A unified Approach to Poker Games. *Masters Thesis*, 561,571.

ORKIN, J. 2004. *Applying Goal-Oriented Action Planning to Games*, first ed. Charles River Media, Hingham, Massachusetts.

ORKIN, J. 2004. *Finite State Machines*, first ed. Charles River Media, Hingham, Massachusetts.

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.

PARK, S. W., LINSEN, L., KREYLOS, O., OWENS, J. D., AND HAMANN, B. 2006. Discrete sibson interpolation. *IEEE Transactions on Visualization and Computer Graphics 12*, 2 (Mar./ Apr.), 243–253.

PARKE, F. I., AND WATERS, K. 1996. *Computer Facial Animation*. A. K. Peters.

PELLACINI, F., VIDIMČE, K., LEFOHN, A., MOHR, A., LEONE, M., AND WARREN, J. 2005. Lpics: a hybrid hardware-accelerated relighting engine for computer cinematography. *ACM Transactions on Graphics 24*, 3 (Aug.), 464–470.

REMCO STRAATMAN, A. B., AND VAN DER STERREN, W. 2006. *Dynamic Tactical Postion Evaluation*, first ed. Charles River Media, Hingham, Massachusetts.

SAILERA, F., BURO, M., AND LANCTOT, M. 2007. Adversarial planning through strategy simulation. *In Proceedings of the IEEE Symposium on Computational Intelligence and Games* (April).

SAKO, Y., AND FUJIMURA, K. 2000. Shape similarity by homotropic deformation. *The Visual Computer 16*, 1, 47–61.

SIERRA, 2006. No one lives forever 2. Accessed April 12, 2006. Available via http://nolf.sierra.com/.

SIERRA, 2007. First encounter assault recon. Accessed December 23, 2007. Available via http://www.whatisfear.com/fear.html.

SILVER, D. 2005. Cooperative pathfinding. *In Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment conference*, 117,122.

SPRONCK, P., SPRINKHUIZEN-KUYPER, I., AND POSTMA, E. 2004. Difficulty Scaling of Game AI. *In Proceedings of GAME-ON 2004: 5th International Conference on Intelligent Games and Simulation*, 33,37.

SPRONCK, P., PONSEN, M., SPRINKHUIZEN-KUYPER, I., AND POSTMA, E. 2006. *Adaptive Game AI with Dynamic Scripting*. Springer Netherlands, 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*. The MIT Press, Cambridge, Massachusetts.

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.

WALLACE, N. 2004. *Hierarchical Planning in Dynamic Worlds*, first ed. Charles River Media, Hingham, Massachusetts.

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* (June).

WIKIPEDIA, 2006. Game balance. Accessed December 20, 2006. Available via http://en.wikipedia.org/wiki/Gamebalance.

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.

YEE, Y. L. H. 2000. *Spatiotemporal sensistivity and visual attention for efficient rendering of dynamic environments*. Master's thesis, Cornell University.

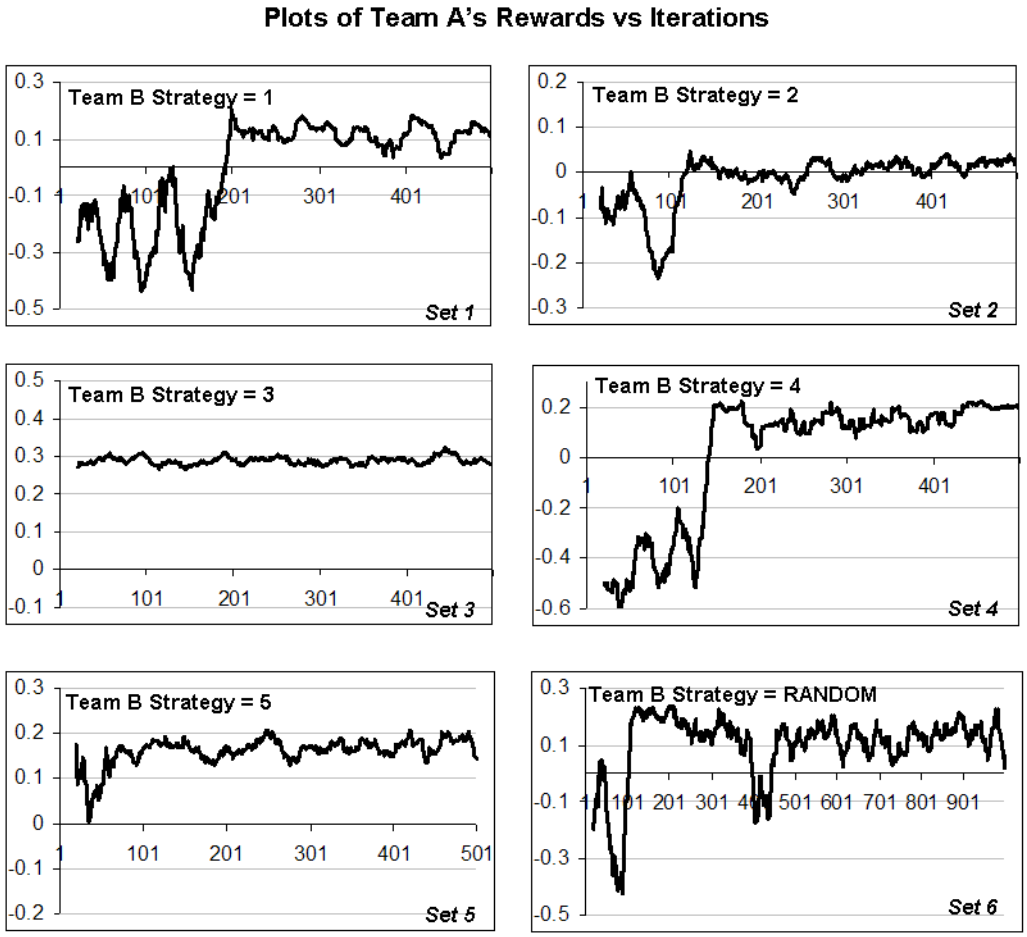

Figure 5: *Experimental Results: Each experiment set has the reward value plotted against the number of iterations. In general Team A (with our AI) is always able to converge to a set of good tactics and strategies for a winning approach.*