Case-Based Learning in Goal-Driven Autonomy Agents for Real-Time Strategy Combat Tasks

Ulit Jaidee[1], Héctor Muñoz-Avila[1], and David W. Aha[2]

[1]Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA
{alc308, hem4}@lehigh.edu
[2]Naval Research Laboratory (Code 5514); Washington, DC 20375
david.aha@nrl.navy.mil

Abstract. We describe a study on using case-based learning techniques in a goal-driven autonomy (GDA) agent for real-time strategy games. The two case bases in our Learning GDA (LGDA) agent store (1) the expected states that an agent can reach when executing an action in and (2) the next goals to pursue when a discrepancy occurs between the expected and encountered states. We report on an ablation study that demonstrates performance gains using LGDA.

Keywords: Case-based learning, goal-driven autonomy, real-time strategy

1 Introduction

Goal-driven autonomy (GDA) is a goal reasoning model in which agents continuously monitor the current plan's execution and assess whether the encountered states match expectations (Molineaux *et al*., 2010). When GDA agents detect a state discrepancy (i.e., when the expected and actual states mismatch), they consider whether to formulate new goals that, if achieved, would fulfill an objective such as maximizing a long-term reward. This monitoring process and the long-term objective of maximizing a reward is reminiscent of reinforcement learning (RL) (Sutton & Barto, 1994). We claim that, by explicitly representing expectations and discrepancies, GDA agents can adapt to discrepancies in dynamic environments more quickly than can RL agents.

We present a GDA learning algorithm that can learn expectations and discrepancies for combat tasks in a real-time strategy (RTS) game, and use the popular Wargus game engine for our environment (Aha *et al*., 2005; Judah *et al.*, 2010). In combat tasks, players maneuver their units to defeat their opponent's units. This task is complex for two reasons. First, multiple unit types exist (with varying capabilities), and RTS games follow a rock-paper-scissors model that forces players to adequately use friendly units based on their types. For example, archers are long-ranged units that can quickly eliminate footmen, which are slow and can only attack at close range. In turn, fast-moving knights can easily eliminate archers using their melee attack. Finally, footmen can eliminate knights by swarming them. Second, RTS games assign units to tactical roles, where each unit in a group assumes a role based on its type and also the types of other units in its group. For example, footmen can play the role of tanks, archers can play the role of heavy artillery, and knights can

play reserve roles, in case something goes wrong. However, if the group lacks footmen, then knights can play the role of tanks.

These two characteristics pose challenges for methods that learn combat tasks. This difficulty is compounded by other characteristics of RTS games: adversarial agents, real-time action execution, and non-deterministic actions (e.g., a stochastic function determines the amount of damage incurred when one unit attacks another).

We are interested in agents that automatically learn which goals are best to achieve, throughout a game, by reasoning about (1) expectations during plan execution and (2) discrepancies that occur between expected and encountered states. LGDA accomplishes these by learning two case bases: an **expectations case base**, which records the expected states encountered, and a **goal formulation case base**, which records the next goal to pursue when a discrepancy occurs.

In Section 2, we define GDA concepts by using the Wargus game elements and by contrasting them with well-known RL concepts. Section 3 presents our techniques for case-based learning; we contrast these with the RL cycle. Section 4 presents an example of a learning episode. Section 5 presents our empirical evaluation. We discuss related work in Section 6 and future work in Section 7.

2 Elements of the Learning Goal-Driven Autonomy Algorithm

LGDA is given the basic action model Σ: $S \times A \rightarrow 2^S$, where Σ denotes, for each state $s \in S$ and each action $a \in A$, the sets of states $\Sigma(s,a)$ that can be reached when a is executed in s. This is the same as the state transition model learned by RL agents and is equivalent to the action model required by planning agents. For example, state information may include that an archer is within attacking range of an enemy footman. After executing an attack action, the environment state may indicate that the footman is killed, the archer is killed, or both are killed.

LGDA doesn't know (1) which actions are "best" for any given state, or (2) the probability that, when executing action a on state s, it will reach a state s'. In our example, the agent doesn't know the probability that the archer or footman will kill its opponent. This is reminiscent of RL agents, which use this information to learn the value $V(s)$ of being in any state s, and the value $Q(s,a)$ of executing a in state s.

However, GDA agents also know the goals G that can be pursued. For example, a goal "divide and conquer" may require splitting a force into two groups, with the first advancing towards the enemy from the north and the latter from the south. Furthermore, for each goal g, LGDA is given the policy π_g: $S \rightarrow 2^{[0,1] \times A}$, which indicates how to achieve g by mapping, for every state s, the probability distribution $\pi_g(s)$, which is a collection of pairs (p,a), where p is the probability that action a is taken. Acquiring such policies automatically is possible as demonstrated in Gillespie et al. (2010), which observes a player's actions while pursuing a goal g and records the probability distribution $\pi_g(s)$ for each state s over multiple games.

Initially, LGDA has no knowledge about which goal in G it should pursue at a given time or the semantics of these goals g (i.e., what they achieve). It is also not given the semantics of their policies π_g. However, if it can learn this information, then LGDA could self-determine which goal it should pursue, and could constantly monitor the environment to make sure that the states it encounters are as expected.

One way that LGDA can learn the meaning of the goal g is by inspecting $\pi_g(s)$ for each state s to examine the possible actions and the probability of taking them, and then observe for each action a in $\pi_g(s)$, the states $\Sigma(s,a)$ that could be reached. However, it is difficult to automatically elicit from these inspections anything resembling our description of the "divide and conquer" goal. Instead, LGDA learns (during environment interactions) the **expectations case base** (ECB), which records the expected states encountered, and the **goal formulation case base** (GFCB), which records the next goal to pursue when a discrepancy occurs.

The ECB enhances the state transition model by defining the probability that a state s′ will be reached when taking action a in state s. More precisely, ECB computes the mapping $S \times A \rightarrow 2^{[0,1] \times S}$, where for each state-action pair (s,a), it returns a probability distribution ECB(s,a) as a collection of pairs (p,s'), where p is the probability that state s' is reached when executing action a in state s.

We use a standard feature-value vector representation to represent a state. More precisely, we define a state as an n-tuple vector of values $s = (v_1,...,v_n)$, where each value v_i is of some type T_i. In our Wargus scenarios, armies include multiple unit types. Given this, our states are vectors s=$(f,a,m,k,f',a',m'k')$, which respectively denote the number of footmen, archers, mages, and knights in the army being controlled by our team and those controlled by the opponent. This simple model works surprisingly well in scenarios where no units and buildings can be created, maps have no obstacles and the available units from the opponents are observable.

We define a state discrepancy (or simply a **discrepancy**) as an n-vector of binary values $(b_1,...,b_n)$. Given two states x and s' (i.e., the expected and actual states), the discrepancy value for the coordinate k, b_k, is defined as 1 if x and s' have the same value on coordinate k and 0 if otherwise.

The GFCB denotes, for a goal g being pursued and a discrepancy d, the expected value of pursuing another goal g' given (g,d). More precisely, GFCB computes the mapping $G \times D \rightarrow 2^{[0,1] \times G}$, where for each goal discrepancy pair (g,d), it returns a probability distribution GFCB(g,d) as a collection of pairs (v,g'), where v is the expected value of selecting goal g' when (g,d) occurs.

3 The Feedback Loop in LGDA

Figure 1 presents the feedback loop that our learning agent LGDA follows when performing goal-driven autonomy after training. LGDA's inputs are a collection of goals and a collection of policies, one policy π_g for each goal g. The outputs from training are the expectations case base ECB and the goal formulation case base GFCB; we discuss learning these below. LGDA begins with some given goal $g = g_{init}$ and an initial state $s = s_0$ (Label 1). The following steps are repeated while LGDA interacts with the environment (i.e., in this paper, Wargus). For further details about LGDA please see Jaidee *et al*. (2011):

1. Given s and g, the next action a is computed by executing policy π_g. Thus, a is chosen according to the probability distribution $\pi_g(s)$ (Label 2).
2. Action a is executed in the environment, resulting in a new state s' (Label 3).

3. After a fixed amount of time, s' is compared against the expected state x (Label 4). Since expected states have a probability distribution, the expected state is computed as the one with the maximum probability:

$$x = \max_p \{(x,p) \in \mathrm{ECB}(s,a)\}.$$

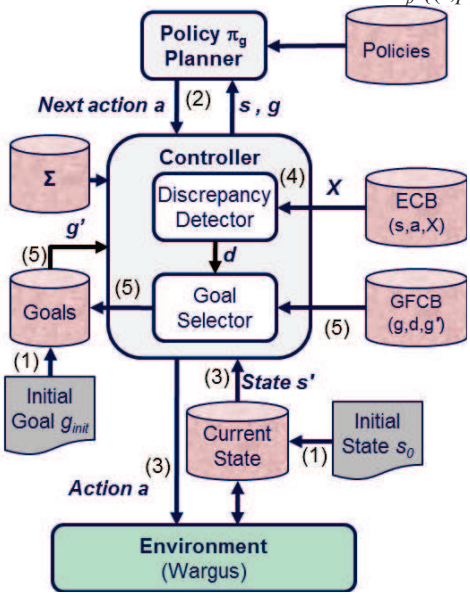

Figure 1: Information flow in LGDA

4. A discrepancy d occurs whenever $x \neq s'$ and it is computed as described in Section 2. When a discrepancy occurs a new goal g' is selected based on (g,d) (Label 5). The new goal is selected with an ε-greedy selection process: with a probability $1-\varepsilon$ a random goal in GFCB(g,d) is selected; with probability ε, the goal with the highest expected value is computed:

$$g' = \max_v \{(g',v) \in \mathrm{GFCB}(g,d)\}$$

Learning the Expectations Case Base. In Step 3, the values of s, a, and s' are used to update the ECB's probabilities for ECB(s,a) by incorporating the new occurrence of (s,a,s'). If an entry (s',p) already exists in ECB(s,a), then its probability is increased (and the probabilities of the other entries are decreased). If the entry (s',p) does not exist in ECB(s,a), then a new one is added with a low probability (and the probabilities of the other entries are slightly decreased).

Learning the Goal Formulation Case Base. In Step 4, the values of g, d, and g' are used to update GFCB via Q-learning (Sutton & Barto, 1998). LGDA maps Q-learning states to (g,d) pairs and maps actions to the next goal g'. Hence, $Q((g,d),g')$ represents the expected value of taking goal g' when g and d are the current goal and discrepancy. The reward is taken from the environment after executing an action. In a Wargus game, the reward is the difference in score between the time that an action a is selected and the time when its execution ends. Hence, our formulation seeks to choose new goals that maximize the game score.

4 Domain Model and Example

To demonstrate case-based learning of GDA agents for real-time strategy combat tasks, we use Wargus for our environment (Figure 2). Wargus is a modified version of Warcraft II®, which is a RTS game developed by Blizzard Entertainment.

Wargus is a multiplayer combat game. In our experiments, we use two competing teams. Each team includes multiple types of units. Units vary in their number of hit points. When a unit is attacked, its hit points are decreased. When a unit's hit points is reduced to zero, it dies (i.e., is removed from the game). When a unit A of one team kills a unit B of another team, the score of unit A's team is increased. For combat tasks, the game ends when all units of one team are eliminated.

Figure 2: Snapshot of a Wargus Game

Teams are initially composed of n units of k different types. **States** are 2k-tuples $(x_1,x_2,\ldots x_k,y_1,y_2,\ldots,y_k)$ where x_i and y_i are the current number of units of type i on our team and the opponent team, respectively. The first half of the vector encodes the number of units for the first team of each type t_i and the second half the types of units of each type $t_{i\text{-}k}$ for the second team. The states include information about only the current units. Thus, each value is an integer in the range $[0,n]$, and the number of units per team also remains in the range $[0,n]$. In our experiments we have 5 types: footmen, archers, knights, mages, and ballistae. ***Actions*** are n-tuples $(a_1, a_2,\ldots a_{n-1}, a_n)$, where $a_i \in \{0,1,\ldots,k\text{-}1,k\}$ such that 0 indicates that the unit does nothing and $a_i \geq 1$ indicates that unit i will attack an opponent unit of type a_i.

The ***utility*** U of state s is defined by the function $U(s) = F(s)–E(s)$, where $F(s)$ is our team's score and $E(s)$ is the enemy's score. The ***discrepancy*** between states s and s' is a $2k$-dimensional vector $(v_0,v_1,\ldots v_{k-1},v_k,v_{k+1},\ldots ,v_{2k-2},v_{2k-1})$, where v_i is true (i.e., 1) if s and s' have the same value in coordinate i and false otherwise.

Example. Suppose each team has four units of two different types, the current goal g is *HiMaxHPTeam* (see Table 1 for a description of the goals) and the current state state s is (2,2,1,2). Suppose that according to policy π_g the next action a is (0,0,0,0) and according to ECB the expected state x for (s,a) is (2,2,0,2). Suppose the resulting state s' after executing a in s is (1,2,1,2). LGDA will calculate the discrepancy d between the current and expected states. Here, d is (false, true, false, true). The next goal g' is chosen by retrieving it from GFCB(g,d), thus closing the loop.

5 Experimental Evaluation

We used the task of winning Wargus games to investigate the hypothesis: *LGDA can significantly outperform ablated agents that use only RL or only CBR, respectively*.

Table 1: List of goals and their associated policies

<table>
<tr><th>Goals</th><th>Description of Policies</th></tr>
<tr><td>HIMAXHPTEAM</td><td>All units attack opponent units with the highest hit points first (ballistae for our experiment), then attack opponent units with second highest hit points (knights), etc.</td></tr>
<tr><td>HIRANGETEAM</td><td>All units on our team attack opponent units with the highest range of attack first (ballistae in our experiment), and then attack opponent units with the second highest range of attack (archers and mages), etc.</td></tr>
<tr><td>GOOFYTEAM</td><td>Different unit types on our team attack different kinds of enemy units as indicated in the following list:
<table>
<tr><th>Type of units</th><th>Attacking list</th></tr>
<tr><td>footman</td><td>knight→footman→archer→ballista→mage</td></tr>
<tr><td>ballista</td><td>footman→knight→archer→mage→ballista</td></tr>
<tr><td>knight</td><td>archer→mage→ballista→knight→footman</td></tr>
<tr><td>archer</td><td>knight→mage→archer→footman→ballista</td></tr>
<tr><td>mage</td><td>knight→footman→archer→ballista→mage</td></tr>
</table></td></tr>
<tr><td>KAMIKAZETEAM</td><td>Different unit types on our team attack different kinds of enemy units as indicated in the following list:
<table>
<tr><th>Type of units</th><th>Attacking list</th></tr>
<tr><td>footman</td><td>ballista→mage→archer→footman→knight</td></tr>
<tr><td>ballista</td><td>knight→ballista→mage→archer→footman</td></tr>
<tr><td>knight</td><td>footman→ mage→archer→ballista→knight</td></tr>
<tr><td>archer</td><td>mage→knight→footman→ballista→archer</td></tr>
<tr><td>mage</td><td>archer→footman→knight→ballista→mage</td></tr>
</table></td></tr>
<tr><td>TYPEDESCENDTEAM</td><td>All units on our team attack opponent enemy units in the same order: mages→archers→knights→ballistae→footmen.</td></tr>
</table>

5.1 Experimental Setup

Game configuration. We ran Wargus on two maps: a medium-sized map (64 x 64 cells) with 8 units per team and a large map (128×128 cells) with 32 units per team. In the medium map, each team had 4 footmen, 1 archer, 1 knight, 1 mage, and 1 ballista. For the large map, each team had 16 footmen, 4 archers, 4 knights, 4 mages, and 4 ballistae. The set of goals used by LGDA and their associated policies are described in Table 1. A game is won by the first team to reach a predetermined score

limit. Scores are computed by Wargus by adding the number of enemy units killed (weighted according to the type of unit) and subtracting the number of own units killed (with the same weights). We set the score limit of the medium map to 200 points and 1000 points for the large map.

We compared LGDA versus the following **agents**: Retaliate (Smith *et al*., 2007), which performs Q-learning, and the ablation Random GDA (RGDA), which replaces LGDA's ε-greedy goal selection procedure with a random selection procedure.

All agents (LGDA, Retaliate, and RGDA) use the same model for *S* and *A*. The learning agents use the same utility function *U*. The definitions for *S*, *A*, and *U* are given in Section 4. Scores were averaged over 10 games.

This study addresses our hypothesis: it directly compares LGDA versus the other agents. We trained each learning agent versus the five policies described in Table 1. LGDA received as input these policies. We recorded results before and after each training repetition of LGDA versus each of the two other agents, continuing until their relative performance stabilized. Knowledge learned during testing was flushed between games. Our performance metric is state utility, as defined in Section 4. For the ε-greedy goal selection procedure, we set ε=0.1 for the large map for LGDA agents: it will choose the next best goal 90% of the time and randomly select a goal 10% of the time. For the medium map, we set ε=0.3 because games on that map end quickly and we want to encourage LGDA to explore other goals. For this same reason we ran 40 iterations for the medium map to obtain a better estimate of asymptotic performance. In contrast, 20 rounds were sufficient to observe this for the large map.

5.2 Experimental Results

Figures 3 and 4 show the results for the medium and large maps. The ordinate plots the utility and the abscissa plots the number of training iterations that have been completed with the 5 policies shown in Table 1.

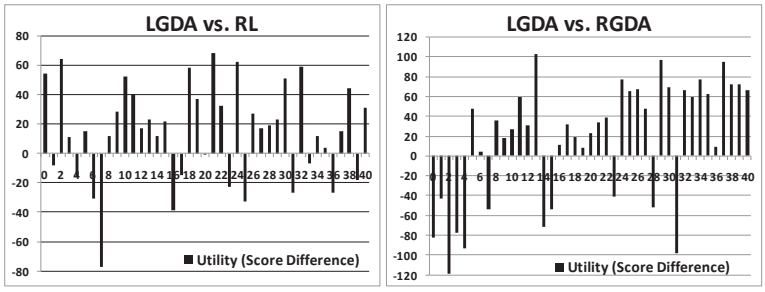

Figure 3: Results for 8 versus 8 units on a medium-sized map

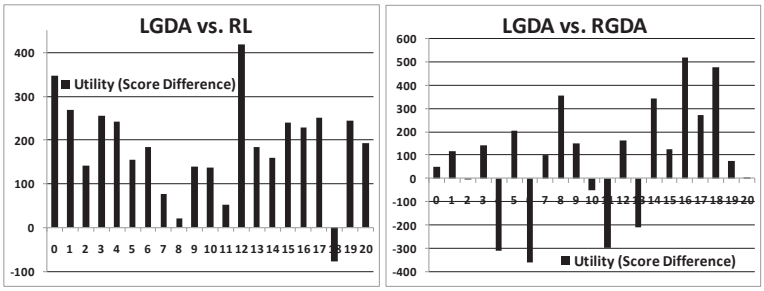

Figure 4: Results for 32 versus 32 units on a large map

The results validate our hypothesis for LGDA versus Retaliate but not for LGDA versus RGDA. For the former, the mean of the underlying distribution of their difference in utility values in the results for the medium map was 14.3 ± 10.1 and in the large map this was 183.8 ± 49.9, both at the 95% confidence level. Thus, this supports our hypothesis that ***LGDA outperforms its RL-only ablation on these maps***. However, the results versus RGDA are 17.2 ± 19.1 and 88.1 ± 110.1, respectively; LGDA is ***not*** *guaranteed* to have a positive utility value vs. RGDA with high confidence. The reason for this is that the given policies are competent and hence, randomly selecting a goal will always select a competent policy. We do observe that, as the number of training iterations increases, LGDA's performance improves compared to RGDA; LGDA won the last 9 medium map games and the last 7 large map games. This indicates that LGDA is fine-tuning its goal selection behavior asymptotically.

6 Related Work

The LGDA algorithm used in this paper is identical to the one described in (Jaidee *et al*., 2011). However, in this paper we instead apply LGDA to combat tasks in real-time strategy games, which is a more complex task than we previously investigated. Weber et al. (2010) also use GDA for real-time strategy games, but their algorithm does not learn expectations. Their cases map discrepancies (between the current state and the goal the agent is trying to achieve) to new goals, which are represented as states, and they use 1-nearest neighbor to compare the current state with recorded cases to perform goal selection. LGDA instead learns expectations, discrepancies, and goals to achieve. Furthermore, goals can be state abstractions (e.g., win the game) and LGDA could map a discrepancy to multiple goals.

Other approaches investigating reasoning techniques for RTS games include reinforcement learning methods (Balla & Fern, 2009) and case-based reasoning (Szczepanski & Aamodt, 2009). LGDA combines both of these methods.

There have been several recent contributions on goal reasoning, including research on goal management in cognitive architectures (Choi, 2010), goal generation (Hanheide *et al*., 2010), and meta-reasoning (Cox, 2007). Applications have included simulated robots (Meneguzzi & Luck, 2007), first-person shooters (Muñoz-Avila *et*

al., 2010), and Navy training simulators (Molineaux *et al*., 2010). We are instead focusing on combat tasks for RTS games.

In contrast to the original model of goal-driven autonomy (Molineaux *et al.,* 2010), LGDA does not perform discrepancy explanation, and we have ignored the topic of goal management. We leave these topics for future research.

GDA is also related to systems that interleave planning and execution (Ambros-Ingerson & Steel, 1988). Those systems detect discrepancies when the state reached after executing the next action doesn't match the action's state. In contrast, LGDA doesn't have knowledge about the action's expectations. Also, those systems replan by pursuing the *same* goals. In GDA, the goals themselves change as a result of resolving observed discrepancies.

7 Conclusions and Future Work

Our study demonstrates the use of case-based techniques to learn knowledge that permits LGDA agents to perform effectively on combat tasks for real-time strategy games. Our algorithm, LGDA, integrates a case-based reasoning method with a reinforcement learning algorithm. Our empirical study demonstrates significant performance gains for LGDA compared to its RL-only ablation, and encouraging but not significant gains versus its CBR-only ablation, on two Wargus scenarios.

In our future work we want to explore more robust representations for the discrepancies. Currently they are simple binary vectors indicating whether the current and expected states match on each feature. Potential extensions could include a degree of dissimilarity. Also, in our current framework the set of goals is fixed. We plan to explore learning new goals when the opportunity arises.

References

Aha, D.W., Klenk, M., Muñoz-Avila, H., Ram, A., & Shapiro, D. (Eds.) (2010). *Goal-Directed Autonomy: Notes from the AAAI Workshop* (W4). Atlanta, GA: [http://www.cse.lehigh.edu/~munoz/gda/]

Aha, D.W., Molineaux, M., & Ponsen, M. (2005). Learning to win: Case-based plan selection in a real-time strategy game. *Proceedings of the Sixth International Conference on Case-Based Reasoning* (pp. 5-20). Chicago, IL: Springer.

Ambros-Ingerson, J. A., & Steel, S. (1988). *Integrating planning, execution and monitoring. Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 83-88). Minneapolis,MN: AAAI Press.

Balla, R.K., & Fern, A. (2009). UCT for tactical assault planning in real-time strategy games. *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence* (pp. 40-45). Pasadena, CA: AAAI Press.

Choi, D. (2010). Reactive goal management in a cognitive architecture. In (D.W. Aha *et al*., 2010).

Cox, M.T. (2007). Perpetual self-aware cognitive agents. *AI Magazine,* **28**(1), 23-45.

Gillespie, K., Karneeb, J., Lee-Urban, S., & Muñoz-Avila, H. (2010). Imitating inscrutable enemies: Learning from stochastic policy observation, retrieval and reuse. *Proceedings of the Eighteenth International Conference on Case Based Reasoning* (pp. 126-140). Alessandria, Italy: Springer.

Hanheide, M., Hawes, N., Wyatt, J., Göbelbecker, M., Brenner, M., Sjöö, K., Aydemir, A., Jensfelt, P., Zender, H., and Kruijff, G-J. (2010). A framework for goal generation and management. In (D.W. Aha *et al*., 2010).

Jaidee, U., Munoz-Avila, H., Aha, D.W. (2011). Integrated learning for goal-driven autonomy. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence*. Barcelona, Spain: AAAI Press.

Judah, K., Roy, S., Fern, A., & Dietterich, T. (2010). Reinforcement learning via practice and critique advice. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. Atlanta, GA: AAAI Press.

McGinty, L., & Smyth, B. (2003). On the role of diversity in conversational recommender systems. *Proceedings of the International Conference on Case-Based Reasoning* (pp. 276-290). Trondheim, Norway: Springer.

Meneguzzi, F.R., & Luck, M. (2007). Motivations as an abstraction of meta-level reasoning. *Proceedings of the Fifth International Central and Eastern European Conference on Multi-Agent Systems* (pp. 204-214). Leipzig, Germany: Springer.

Molineaux, M., Klenk, M., & Aha, D.W. (2010). Goal-driven autonomy in a Navy strategy simulation. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. Atlanta, GA: AAAI Press.

Muñoz-Avila, H., Jaidee, U., Aha, D.W., & Carter, E. (2010). Goal-driven autonomy with case-based reasoning. *Proceedings of the Eighteenth International Conference on Case-Based Reasoning* (pp. 228-241). Alessandria, Italy: Springer.

Powell, J., Molineaux., M., & Aha, D.W. (2011). Active and interactive discovery of goal selection knowledge. In *Proceedings of the Twenth-Fourth Conference of the Florida AI Research Society.* Palm Beach, FL: AAAI Press.

Smith, M., Lee-Urban, S., & Muñoz-Avila, H. (2007). RETALIATE : Learning winning policies in first-person shooter games.*Proceedings of the Seventeenth Innovative Applications of Artificial Intelligence Conference* (pp. 1801-1806). Vancouver (BC), CA: AAAI Press.

Sutton, R.S., & Barto, A.G. (1998). *Reinforcement learning: An introduction*. MIT Press: Cambridge, MA.

Szczepanski, T., & Aamodt, A. (2009). Case-based reasoning for improved micromanagement in real-time strategy games. In S.J. Delaney (Ed.) *Case-Based Reasoning for Computer Games: Proceedings of the ICCBR Workshop*. Seattle, WA.

Weber, B., Mateas, M., & Jhala, A. (2010). Applying goal-driven autonomy to StarCraft. In *Proceedings of the Sixth Conference on Artificial Intelligence and Interactive Digital Entertainment*. Stanford, CA: AAAI Press.