Beating the Defense: Using Plan Recognition to Inform Learning Agents

Matthew Molineaux[1], David W. Aha[2], & Gita Sukthankar[3]

[1]Knexus Research Corporation, Springfield, VA, 22153
[2]Navy Center for Applied Research in Artificial Intelligence; Naval Research Laboratory (Code 5514); Washington, DC 20375
[3]School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816
david.aha@nrl.navy.mil | matthew.molineaux@knexusresearch.com | gitars@eecs.ucf.edu

Abstract

In this paper, we investigate the hypothesis that plan recognition can significantly improve the performance of a case-based reinforcement learner in an adversarial action selection task. Our environment is a simplification of an American football game. The performance task is to control the behavior of a quarterback in a pass play, where the goal is to maximize yardage gained. Plan recognition focuses on predicting the play of the defensive team. We modeled plan recognition as an unsupervised learning task, and conducted a lesion study. We found that plan recognition was accurate, and that it significantly improved performance. More generally, our studies show that plan recognition reduced the dimensionality of the state space, which allowed learning to be conducted more effectively. We describe the algorithms, explain the reasons for performance improvement, and also describe a further empirical comparison that highlights the utility of plan recognition for this task.

1. Motivation and Contributions

Large state spaces pose a challenge for reinforcement learning (RL) algorithms due to the amount of data required to develop accurate action-selection policies. For example, when using the observed state variables, the performance task that we analyze in this paper has a large state space ($4.3*10^9$), which is common for adversarial multiagent environments. Due to this and other characteristics of our task, if we used a simple Q-learning algorithm, then learning an accurate policy would require an inordinately large (and practically infeasible) number of trials.

Case-based reasoning (CBR) methods are an attractive approach for solving this problem because they assume that the same (or similar) actions are best performed among a given set of similar states. When this assumption holds, then generalizing from previous experiences can greatly reduce the number of states that need to be visited, during trials, to learn an accurate policy. Also, CBR methods are comparatively simple to encode, intuitive, and have a good performance record for assisting with reinforcement learning (e.g., Ram & Santamaria, 1997; Sharma *et al*., 2007; Molineaux *et al*., 2008).

Unfortunately, CBR methods are not a panacea; they provide only one part of a solution to this problem. For example, like reinforcement learning algorithms (Barto & Mahadevan, 2003) they learn slowly when state descriptions have high dimensionality because this complicates the task of identifying similar cases. Thus, they can benefit from techniques that reformulate the state space to address this problem (e.g., Aha, 1991; Fox & Leake, 2001).

One method for reformulating the state space involves using *plan recognition* (Sukthankar, 2007) to reveal hidden variables (e.g., concerning opponent intent), which can then be incorporated into the state space used by learning algorithms. This has the potential to transform a partially observable environment into a fully observable environment (Russell & Norvig, 2003).

We investigate the utility of a plan recognition method for reformulating the state space of a case-based reinforcement learning algorithm so as to improve its performance on a complex simulation task. We claim that plan recognition can significantly increase long-term rewards on this task, describe an algorithm and its empirical study that supports this conclusion, hypothesize a reason for its good performance, and report on its subsequent investigation.

Section 2 describes our task environment, which is a limited American football game simulation. We then describe related work in Section 3 before introducing our case-base reinforcement learner, and its plan recognition extensions, in Section 4. Our empirical study, results analysis, and subsequent investigation are described in Section 5. We discuss these in Section 6 and conclude in Section 7.

2. Domain and Performance Task

American football[1] is a game of skill played by two teams on a rectangular field. RUSH 2005[2] is an open-source American football simulator whose teams have only 8 players and whose field is 100x63 yards. We use a variant of Rush that we created (*RUSH 2008*[3]) for our investigation.

[1] http://en.wikipedia.org/wiki/American_Football
[2] http://rush2005.sourceforge.net/
[3] http://www.knexusresearch.com/projects/rush

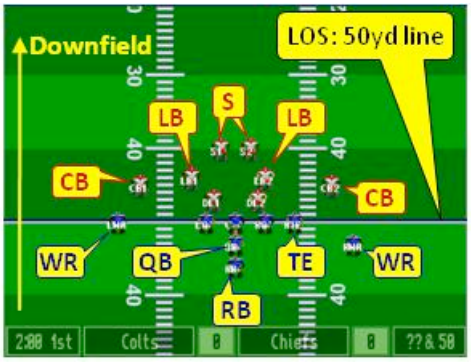

Figure 1: Our study's starting Rush 2008 formation for the offense (**blue**) and defense (**red**), with some player/position annotations.

Our investigation involved learning to control the quarterback's actions on repeated executions of the same offensive play (a *pass*). In this context, the offensive team's players are instructed to perform the following actions:

Quarterback (QB): Given the ball at the start of play while standing 3 yards behind the center of the line of scrimmage (LOS), our QB agent decides whether and when he runs, stands, or throws (and to which receiver).

Running Back (RB): Starts 3 yards behind the QB, runs a pass route 7 yards left and 4 yards downfield.

Wide Receiver #1 (WR1): Starts 16 yards to the left of the QB on the LOS, runs 5 yards downfield and turns right.

Wide Receiver #2 (WR2): Starts 16 yards to the right of the QB a few yards behind the LOS, runs 5 yards downfield, and waits.

Tight End (TE): Starts 8 yards to the right of the QB on the LOS and pass-blocks.

Offensive Linemen (OL): These 3 players begin on the LOS in front of the QB and pass-block (for the QB).

In our investigation, the defense always used plays starting from the same formation, and acts as follows:

Defensive lineman (DL): These 2 players line up across the LOS from the OL and try to tackle the ball handler.

Linebacker (LB): These 2 players start behind the DL, and will blitz the QB, guard a particular zone of the field, or guard an *eligible* receiver (i.e., the RB, WR1, or WR2), depending on the play.

Cornerback (CB): These 2 players line up across the LOS from the WRs and guard a player or a zone on the field.

Safety (S): These 2 players begin 10 yards behind the LOS and assist with pass coverage or chase offense players.

Figure 1 displays the starting formation we used for both teams in each play. All players pursue a set play based on their specific instructions (i.e., for the offense, a location to run to followed by a behavior to execute), except for the QB, whose actions are controlled by our learning agent. However, due to the stochastic nature of the simulator, the play does not unfold the same way each time. Each player possesses unique skills (specified using a 10-point scale) including power, speed, and skill; these affect his ability to handle the ball, block, run, and tackle other players. The probability that a passed ball is caught is a function of the number of defenders near the intended ball receiver, the skills of the ball receiver and the nearby defenders (if any), and the distance in which the ball was thrown.

Figure 2: Six of the QB's eight possible actions. Not shown are **Throw RB** and **Noop**.

The physics of the simulator are simplified. When a player or the ball starts to move, it takes on a constant velocity, with the exception that the ball will accelerate downwards due to gravity. All objects are represented as rectangles that interact when they overlap (resulting in a catch, block, or tackle).

Within RUSH, we examine the task of learning how to control the quarterback's actions so as to optimize yardage gained on a single (repeated) play. At the start of each play, the defense secretly and randomly chooses one of five plays/strategies that begin from the same known formation. These plays are named "Half-and-Half", "Soft Covers", "Pass Blanket", "Hard Blitz", and "Pressure RB". The offensive team always uses the same passing play, as detailed above. Only the QB is controlled. The other player's actions are slightly variable, and they may not run the same path every time, even though they will follow the same general directions. This task is *stochastic* because the other players' actions are random within certain bounds. It is also partially *hidden*: while each player's positions and movements are visible, one of the determinants of those movements (i.e., the defensive strategy) is not observable.

The QB can perform one of eight actions (see Figure 2) at each time step during the offensive play. The first four, **Forward**, **Back**, **Left**, and **Right** cause the QB to move in a certain direction for one time step. Three more cause the QB to pass to a receiver (who is running a pre-determined pass route): **Throw RB, Throw WR1, Throw WR2**. Finally, one action causes the quarterback to stand still for a time step: **Noop**. The QB may decide to run the football himself. The quarterback must choose actions until either he throws the ball, crosses into the end zone (i.e., scores a touchdown by gaining 50 yards from the LOS), or is tackled. If the QB passes, no more actions are taken, and the play finishes as soon as an incompletion occurs, an interception occurs, or the successful receiver has been tackled or scores a touchdown.

At the start of each play, the ball is placed at the center of the line of scrimmage (LOS) along the 50 yard line. The agent's *reward* is 1000 for a touchdown (i.e., a gain of at least 50 yards), -1000 for an interception or fumble, or is otherwise ten times the number of yards gained (e.g., 0 for an incomplete pass) when the play ends. A reward of 0 is received for all actions before the end of the play. Touch-

downs, interceptions, and fumbles are relatively rare. Touchdowns occur between 0.01% of the time (for a low performer) and 0.2% of the time (for a high performer). Interceptions and fumbles combined occur between 1% and 3% of the time.

3. Related Work

Plan recognition concerns the task of inferring the goals of an agent and their plan for achieving them (Carberry, 2001). Ours is a simple instantiation of this in which we know the opponents' goals (i.e., minimize yardage gained and gain possession if possible), and few plans are used.

Plan recognition has a long history in CBR research (e.g., Kass, 1991), particularly in the context of adversarial, real-time multiagent games. For example, Fagan and Cunningham (2003) acquire cases (state-action planning sequences) for predicting a human's next action while playing SPACE INVADERS™. We instead focus on predicting the actions of a team of coordinating players. Cheng and Thawonmas (2004) propose a case-based plan recognition approach for assisting players with low-level management tasks in WARGUS. However, they do not observe the adversary's tactical movements, which is our focus. Finally, Lee et al. (2008) use Kerkez and Cox's (2003) technique to create an abstract state, which counts the number of instances of each type-generalized state predicate. On a simplified WARGUS task, their integration of CBR with a simple reinforcement learner performs much better when using the abstract state representation to predict opponent actions. While our approach also performs state abstraction, our states are not described by relational predicates, and this technique cannot be applied to our task.

Several additional CBR researchers have recently investigated planning techniques in the context of real-time simulation games (e.g., Aha *et al*., 2005; Ontañón *et al*., 2007; Sugandh *et al*., 2008). While some employed reinforcement learning algorithms (e.g., Sharma *et al*., 2007; Molineaux *et al*., 2008; Auslander *et al*., 2008), none leveraged plan recognition techniques.

CBR is frequently used in *team* simulation games such as ROBOCUP SOCCER (e.g., Karol *et al.,* 2003; Srinivasan *et al*., 2006; Ros *et al*., 2007). Unlike our own, these efforts have not focused on plan recognition or on alternative approaches for learning a state representation to enhance reinforcement learning behavior. Among more closely-related work, Wendler and Bach (2003) report excellent results for a CBR algorithm that predicts agent behaviors from a pre-defined set. We instead use plan recognition to assist reinforcement learning, and our opponent's behaviors are instead learned via clustering. Finally, Steffens (2005) examines the utility of adding *virtual* features that model the opponent's team and showed that, when weighted appropriately, can significantly increase player prediction accuracies. However, these features were hand-coded rather than learned via a plan recognition method.

There has been limited use of clustering to assist with plan recognition in related tasks. For example, Riley et al. (2002) use a clustering technique based on fitting minimal rectangles to player logs of Robocup simulator league data to identify player *home areas*. A player's home area is defined as the segment of the field where the player spends 90% of the game time. However, knowing a player's home area is insufficient to perform state-space reduction. In general, our use of an EM clustering approach for plan recognition is fairly unique; most related research focuses on determining which plan is being executed rather than the plan's cluster/category.

Finally, we recently reported successful results when using a *supervised* plan recognition approach to predict the *offensive* team's play (Shore *et al*., submitted), but we did *not* use it as leverage in a subsequent learning task, which is the focus of this paper.

4. Algorithm

Our algorithm is based on the Q(λ) algorithm (Sutton & Barto, 1998); it uses a set of case bases to approximate the Q function and an EM clustering algorithm to add opponent plan information to the state. We call it *Case-Based Q-Lambda with Plan Recognition* (CBQL-PR).

4.1 Plan Recognition Task

In CBQL-PR, plan recognition is an online learning task that clusters the observable movements of all the defensive players into groups. The perceived movement $m \in \boldsymbol{M}$ for each defensive player is the direction that player is moving during a time step, which has nine possible values:

$\boldsymbol{M}$ = {*None, Forward, Left, Right, Back, Forward-Right, Forward-Left, Back-Right, Back-Left*}

Directions are geocentric; *Forward* is always in the direction of play (downfield), and all other directions are equally spaced at 45-degree angles. Clustering is performed after the third time step of each play, so three "snapshots" of the defensive players' movements are used. Thus, 24 features are used to represent defensive plays (i.e., the directions on each of three steps for each of eight defensive players). For the first 1000 trials, examples were added to the batch to be clustered, but the predicted cluster (i.e., the recognized plan) was not used in action selected.

We used the Expectation-Maximization (EM) algorithm from the Weka[1] suite of machine learning software for clustering. EM iteratively chooses cluster centers and builds new clusters until the centers move only marginally between iterations. Membership of an example in a cluster is calculated as the product of the within-cluster frequencies of each value in the feature vector. EM also increases the number of clusters to discover until successive steps decrease the average log-likelihood of instances in a final clustering. We selected EM after reviewing several algorithms; the clusters it found matched the defensive plays over 99% of the time in less than 1000 examples.

[1] http://www.cs.waikato.ac.nz/ml/weka/

4.2 Action Selection Task

CBQL-PR periodically selects an action that either maximizes the expected return (*exploiting* learned knowledge), or improves its knowledge of the value space (*exploring* the environment) so as to maximize the long-term reward.

CBQL-PR uses a set of case bases to approximate the standard RL Q function, which maps state-action pairs to an estimate of the long-term reward for taking an action a in a state s. There is one Q_a *case base* in this set for each action $a \in \boldsymbol{A}$, where $\boldsymbol{A}$ is the set of actions defined by the environment. These case bases support a case-based problem solving process consisting of a cycle of case retrieval, reuse, revision, and retention (Aamodt & Plaza, 1994). For faster retrieval, we use kd-trees to index cases. At the start of each experiment, each Q_a case base is initialized to the empty set; cases are added and modified as new experiences are gathered, which provide new local estimates of the Q function. Cases in Q_a are of the form $<s, v>$, where s is a feature vector describing the state (it contains a combination of integer, real, and symbolic values) and v is a real-valued estimate of the reward obtained by taking action a in state s and then pursuing the current approximation of the optimal policy until the task terminates.

At each time step, a state is observed by the agent, and an action is selected. With probability ε, a random action will be chosen (*exploration*). With probability 1-ε, the algorithm will predict the best action to take (*exploitation*). To do this, it *reuses* each Q_a case base by performing a locally-weighted regression using a Gaussian kernel on the *retrieved k* nearest neighbors of the current observed state s. Similarity is computed using a normalized Euclidean distance function. This produces an estimate of the value of taking action a in the current observed state s. CBQL-PR selects the action with the highest estimate, or a random action if any case base has fewer than 7 nearest neighbors.

Once that action is executed, a reward r and a successor state s' are obtained from the RUSH 2008 environment. This reward is used to improve the estimate of the Q function. If the case is sufficiently novel (more than a distance τ from its nearest neighbor) a new case is *retained* in Q_a with state s and $v = r + \gamma \max_{a \in A} Q_{a'}(s')$, where $Q_{a'}()$ denotes the current estimate for a state in Q_a and $0 \leq \gamma < 1$ is the discount factor. This update stores an estimate of the value of taking action a in state s based on the known value of the best successor state and action. If the case is not sufficiently novel, the 7 nearest neighbors are *revised* according to the current learning rate α and their contribution β to the estimate of the state's value (determined by a normalization over the Gaussian kernel function, summing to 1). The solution of each case is updated using:

$$v = v + \alpha\beta\,[r + \gamma \max_{a \in A} Q_{a'}(s') - Q_a(s)].$$

Finally, the solutions (values) of all cases updated earlier in the current trial are updated according to their λ-eligibility:

$$v = v + (\gamma\lambda)^t \alpha\beta\,[r + \gamma \max_{a \in A} Q_{a'}(s') - Q_a(s)],$$

where t is the number of steps between the earlier use and the current update, and $0 \leq \lambda < 1$ is the trace decay parameter.

4.3 State Definitions

In addition to CBQL-PR, we investigate two non-clustering variants of the algorithm which do not perform plan recognition, $CBQL_{base}$ and $CBQL_{opt}$; they differ only in their representation of the state. $CBQL_{base}$ uses the time step and eight features from the set M described in Section 4.1 (i.e., the directional movements of the eight defensive players for the most recent time step), In contrast, CBQL-PR instead uses only the predicted cluster and the time step. Before its third turn, CBQL-PR's second feature takes on a distinct value indicating *no prediction*. We compared CBQL-PR with $CBQL_{base}$ to examine whether clustering improves CBQL-PR's performance over RL alone.

$CBQL_{opt}$ uses an optimized 5-dimensional state description which includes four real-valued features that are intuitively helpful in the QB's decisions of when and where to throw. It uses three features to indicate each eligible receiver's distance from the nearest defensive player (indicating how well each is covered). A fourth feature denotes the QB's distance from the closest defensive player (indicating the likelihood that he will be tackled imminently). The final feature is the current time step. This state representation more closely resembles a conventional RL state, containing features selected for easy disambiguation of the right action to use, rather than capturing opponent plans. See Table 1 for more details.

5. Evaluation

Our empirical study focuses on analyzing how the state representation affects the performance of a case-based Q(λ) algorithm on a task in RUSH 2008. We hypothesized that clustering via plan recognition would yield a state representation that significantly improves performance. We used the experimentation platform LIET, the Lightweight Integration and Evaluation Testbed. LIET is a free tool we developed that can be used to evaluate the performance of agents on tasks in integrated simulation environments. LIET managed communication between RUSH 2008 and CBQL, ran the experiment protocol, and collected results.

We assessed performance in terms of two metrics: asymptotic advantage and regret. Aysmptotic advantage is defined as the difference between the asymptotic performances of two algorithms, which we compute by averaging the performance achieved during the final 10 testing periods. The second metric, regret (Kaelbling *et al*., 1996), is the integral difference between the performance curves of two algorithms. To normalize, the regret is divided by a bounding box defined by the most extreme values in each

Table 1: Summary of the algorithms used in the first experiment.

Algorithm	Plan Recognition?	State	State Example
$CBQL_{base}$	×	Instantaneous direction of each defender + time step	<Back, None, Back-Right, Back, Left, Left, Back-Left, Forward, 3 >
$CBQL_{opt}$	×	Distance from receiver to nearest defender (3) + distance from QB to nearest defender + time step	<1.5, 6.0, 3.0, 0.9, 3>
CBQL-PR	✓	Predicted cluster + time step	<cluster_5, 1>

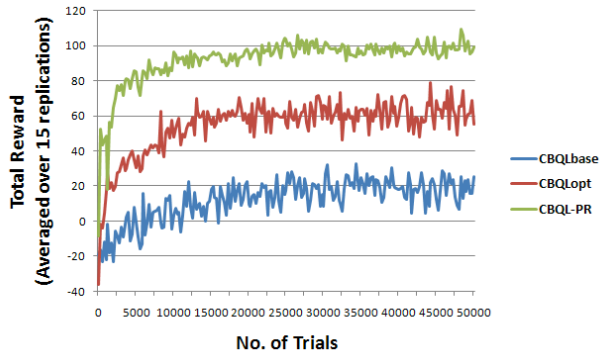

Figure 3: Comparison highlighting the utility of our plan recognition approach for a case-based reinforcement learner on the task of controlling the *Rush 2008* QB's actions.

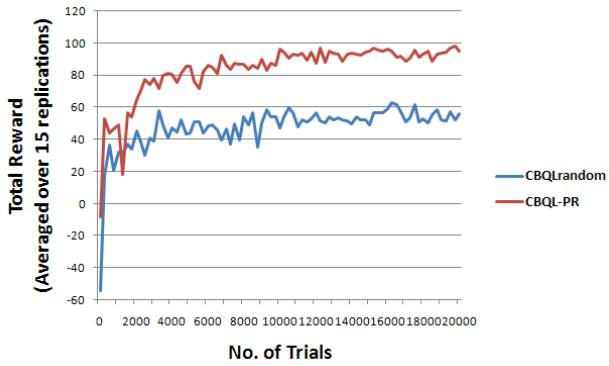

Figure 4: When using the same arity for the state representation, randomly predicting the defensive play performs poorly vs. using plan recognition's predictions.

dimension from both curves. The domain metric measured is the total reward, as defined in Section 2.

We compared three variants of the learning algorithm, each using one of the different state representations defined in Section 4.3. In particular, only CBQL-PR employs a plan recognition method. For each algorithm, we used the following values for the constants to update the case base: learning rate α=0.2, discount factor γ=0.999, exploration parameter ε=0.2, trace decay λ=0.9, neighbor count k=7, and distance threshold τ=0.001. Both α and ε were decreased asymptotically to 0 over time.

Each algorithm variant was tested against the same set of five defensive plays. Each of these plays denotes a different set of behaviors for the defensive team, and for each play, there is a distinct optimal offensive strategy. As the environment is stochastic, a series of actions may produce different rewards if attempted on successive trials.

We ran ten replications of our experiment for each agent. Experiments lasted for 100,000 training trials, with a random defensive strategy selected at the beginning of each training trial. After every 250 training *trials*, we tested the algorithm ten times against each defensive strategy. Each point in Figure 3 is the average performance over one testing period. On average, CBQL-PR found more clusters than actual plays; the mean was 6.6. However, no cluster corresponded to more than one play; rather, multiple clusters sometimes were found corresponding to the same play. Also, predictions were found to be accurate 100% of the time in the limit; each time a particular cluster was predicted, the defense was using the same play.

The results confirm our hypothesis that ***pattern recognition can significantly improve the performance of case-based RL on this task***. That is, CBQL-PR significantly outperformed $CBQL_{base}$ and $CBQL_{opt}$ in k-step regret (*vs.* $CBQL_{base}$=.578, p <.0001 and *vs.* $CBQL_{opt}$= .271, p<.0001) and asymptotic advantage (*vs.* $CBQL_{base}$=82.9, p<.0001 and *vs.* $CBQL_{opt}$=39.0, p<.0001).

A key distinguishing characteristic between CBQL-PR and its variants is its smaller state dimensionality (i.e., two rather than five or nine). To test the hypothesis that this is not the sole reason for its improved performance, we also evaluated $CBQL_{random}$, a variant whose first feature is randomly chosen from the interval [1, # *defensive plays*], and whose second feature is the time step (results in Figure 4). CBQL-PR statistically outperforms this version also (*regret vs.* $CBQL_{random}$=.272, p<.0001; *asymptotic advantage vs.* $CBQL_{random}$=46.5, p<.0001), confirming our hypothesis.

6. Discussion

We showed that using recognized plans in the state representation improves the performance of our case-based reinforcement learning algorithm on a simulated American football task. We compared the performance of our algorithm using multiple representations, and the version using plan recognition achieves the highest asymptotic performance. It also learns more quickly, achieving the highest performance found by the runner-up in 10% of the time.

This performance improvement is primarily due to two advantages of CBQL-PR's state space formulation. The first is its lower dimensionality, while the second is that the opponent's plans, which are important in explaining variations in performance, are identified.

While useful, this algorithm does not dominate in all situations. Other experiments (not discussed in this paper), showed that CBQL-PR does not outperform $CBQL_{opt}$ against all possible defenses. In particular, when a single series of actions performs well against all defenses, $CBQL_{opt}$ performs as well as or better than CBQL-PR. However, CBQL-PR may perform well in other domains where broader opponent strategies can be grouped into sets to be understood better. In future work, we will test CBQL-PR against a larger range of opponent strategies. We will also extend our work to cover the full game of American football, including choice of offensive play with different starting conditions (e.g., distance from goal). Also, we will investigate learning a more detailed representation of plays, which will allow us to generalize over similar plays.

7. Conclusions

Plan recognition methods can be a powerful ally for machine learning techniques. We investigated the utility of a clustering algorithm for distinguishing opponent plans in a multi-agent simulation of plays from an American football

game. By replacing a low-level feature representation with a learned, accurate prediction of the opponent's plan, this type of plan recognition can significantly increase the performance of a case-based reinforcement learner on an agent control task. We conjecture that similar approaches can improve the performance of learning algorithms on a large variety of tasks, and in particular for tasks that can benefit from the predictions of other agents' actions.

Acknowledgements

Thanks to DARPA (#07-V176) and the Naval Research Laboratory for supporting this work, and to the reviewers for their helpful comments.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications,* **7**, 39-59.

Aha, D.W. (1991). Incremental constructive induction: An instance-based approach. *Proceedings of the Eighth International Workshop on Machine Learning* (pp. 117-121). Evanston, IL: Morgan Kaufmann.

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.

Auslander, B., Lee-Urban, S., Hogg, C., & Muñoz-Avila, H. (2008). Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning. *Proceedings of the Ninth European Conference on Case-Based Reasoning* (pp. 59-73). Trier, Germany: Springer.

Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems,* **13**(4), 341–379.

Carberry, S. (2001). Techniques for plan recognition. *User Modeling and User-Adapted Interaction,* **11**(1-2), 31-48.

Cheng, D.C., & Thawonmas, R. (2004). Case-based plan recognition for real-time strategy games. *Proceedings of the Fifth Game-On International Conference* (pp. 36-40). Reading, UK: University of Wolverhampton Press.

Fagan, M., & Cunningham, P. (2003). Case-based plan recognition in computer games. *Proceedings of the Fifth International Conference on Case-Based Reasoning* (pp. 161-170). Trondheim, Norway: Springer.

Fox, S. & Leake, D. (2001). Introspective reasoning for index refinement in case-based reasoning. *Journal of Experimental and Theoretical Artificial Intelligence*, **13**(1), 63-88.

Kaelbling, L.P., Littman, M.L., & Moore, A.W. (1996). Reinforcement learning: A survey. *JAIR*, **4**, 237-285.

Karol, A., Nebel, B., Stanton, C., & Williams, M.-A. (2003). Case based game play in the RoboCup four-legged league: Part I the theoretical model. In D. Polani, B. Browning, A. Bonarini, & K. Yoshida (Eds.) *RoboCup 2003: Robot Soccer World Cup VII* (pp. 739-747). Padua, Italy: Springer.

Kass, A. (1991). Adaptation strategies for case-based plan recognition. *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society* (pp. 161-166). Chicago, IL: Lawrence Erlbaum Associates.

Kerkez, B., & Cox, M.T. (2003). Incremental case-based plan recognition with local predictions. *International Journal on Artificial Intelligence Tools: Architectures, languages, algorithms*, **12**(4), 413-464.

Lee, J., Koo, B., & Oh, K. (2008). State space optimization using plan recognition and reinforcement learning on RTS game. In *Proceedings of the International Conference on Artificial Intelligence, Knowledge Engineering, and Data Bases*. Cambridge, UK: WSEAS Press.

Molineaux, M., Aha, D.W., & Moore, P. (2008). Learning continuous action models in a real-time strategy environment. *Proceedings of the Twenty-First Annual Conference of the Florida Artificial Intelligence Research Society* (pp. 257-262). Coconut Grove, FL: AAAI Press.

Ontañón, S., Mishra, K., Sugandh, N., & Ram, A. (2007). Case-basd planning and execution for real-time strategy games. *Proceedings of the Seventh International Conference on Case-Based Reasoning* (pp. 164-178). Belfast, N. Ireland: Springer.

Ram, A., & Santamaria, J.C. (1997). Continuous case-based reasoning. *Artificial Intelligence*, **90**(1-2), 25-77.

Riley, P., Veloso, M., & Kaminka, G. (2002). An empirical study of coaching. In H. Asama, T. Arai, T. Fukuda, & T. Hasegawa (Eds.) *Distributed Autonomous Robotic Systems 5*. Berlin: Springer.

Ros, R., López de Mántaras, R., Arcos, J.L., & Veloso, M.M. (2007). Team playing behavior in robot soccer: A case-based reasoning approach. *Proceedings of the Seventh International Conference on Case-Based Reasoning* (pp. 46-60). Belfast, N. Ireland: Springer.

Russell, S., & Norvig, P. (2003). *Artificial intelligence: A modern approach* (2nd ed.). Upper Saddle River, NJ: Prentice Hall.

Sharma, M., Holmes, M., Santamaria, J.C., Irani, A., Isbell, C., & Ram, A. (2007). Transfer learning in real-time strategy games using hybrid CBR/RL. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*. [http://www.ijcai.org/papers07/contents.php]

Srinivasan, T., Aarthi, K., Meenakshi, S.A., & Kausalya, M. (2006). CBRRoboSoc: An efficient planning strategy for robotic soccer using case-based reasoning. In *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation* and the *International Conference on Intelligent Agents, Web Technologies, and Internet Commerce*. Sydney, Australia: IEEE Press.

Steffens, T. (2005). Similarity-based opponent modelling using imperfect domain theories. (2005). In G. Kendall & S. Lucas (Eds.) *Proceedings of the Symposium on Computational Intelligence and Games* (pp. 285-291). Essex, UK: IEEE Press.

Sugandh, N., Ontañón, S., & Ram, A. (2008). Real-time plan adaptation for case-based planning in real-time strategy games. *Proceedings of the Ninth European Conference on Case-Based Reasoning* (pp. 533-547). Trier, Germany: Springer.

Sukthankar, G. (2007). *Activity recognition for agent teams* (Technical Report CMU-RI-TR-07-23). Doctoral dissertation: Robotics Institute, Carnegie Mellon U., Pittsburgh, PA.

Shore, H.L., Blues, J.J.E., & Blues, E.J. (submitted). Recognizing opponent intent in Rush Football. Manuscript submitted for publication.

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

Wendler, J., & Bach, J. (2003). Recognizing and predicting agent behavior with case-based reasoning. In D. Polani, B. Browning, A. Bonarini, & K.Yoshida (Eds.) *RoboCup 2003: Robot Soccer World Cup VII* (pp. 729-738). Padua, Italy: Springer.