International Joint Conference on Artificial Intelligence (IJCAI, 2007)

Transfer Learning in Real-Time Strategy Games Using Hybrid CBR/RL

Manu Sharma, Michael Holmes, Juan Santamaria, Arya Irani, Charles Isbell, Ashwin Ram
College of Computing
Georgia Institute of Technology
{manus, mph, jcs, arya, isbell, ashwin} @cc.gatech.edu

Abstract

The goal of transfer learning is to use the knowledge acquired in a set of source tasks to improve performance in a related but previously unseen target task. In this paper, we present a multi-layered architecture named CAse-Based Reinforcement Learner (CARL). It uses a novel combination of Case-Based Reasoning (CBR) and Reinforcement Learning (RL) to achieve transfer while playing against the Game AI across a variety of scenarios in MadRTS™, a commercial Real Time Strategy game. Our experiments demonstrate that CARL not only performs well on individual tasks but also exhibits significant performance gains when allowed to transfer knowledge from previous tasks.

1 Transfer Learning

Transferring knowledge from previous experiences to new circumstances is a fundamental human capability. In general, the AI community has focused its attention on *generalization*, the ability to learn from example instances from a task and then to interpolate or extrapolate to unseen instances of the same task. Recently, there has been growing interest in what has come to be known as *transfer learning*. Transfer learning is somewhat like generalization across tasks; that is, after seeing example instances from a function or task, the transfer learning agent is able to show performance gains when learning from instances of a different task or function. For example, one might imagine that learning how to play checkers should allow one to learn to play chess, a related but unseen game, faster.

One particular taxonomy of transfer learning has been proposed by DARPA [2005]. There are eleven "levels", ranging from the simplest type of transfer (Level 0, or Memorization) to the most complex form of transfer (Level 10, or Differing). The levels measure complexity of the transfer problem in terms of speed, quality, reorganization, and scale. Of note is that the levels emphasize characteristics of the types of problems to which the transfer applies, rather than the representations or semantic content of the knowledge needed for solutions. See Table 1 for the first five transfer levels.

We introduce a mechanism for achieving transfer learning in Real Time Strategy (RTS) games. Our technical contributions are:

- A novel multi-layered architecture named CAse Based Reinforcement Learner (CARL). It allows us to capture and separate the strategic and tactical aspects of our domain, and to leverage the modular nature of transfer learning.
- A novel combination of Case-Based Reasoning (CBR) and Reinforcement Learning (RL). In particular, we use CBR as an instance-based state function approximator for RL, and RL as a temporal-difference based revision algorithm for CBR.

In the next section, we briefly introduce and motivate the choice of RTS games as an application domain. Section 3 details the hybrid CBR/RL approach for transfer learning. We then provide detailed experimental analyses demonstrating validity of the proposed techniques in Section 4. Finally, we describe related work before ending with a discussion of recommendations for future research.

2 Real Time Strategy Games

RTS games are a family of computer games that involve several players competing in real time. Typically, the goal is to kill all the other players in the game or to dominate most of the territories of the game.

RTS games are very popular, and have been created by non-AI developers for a non-AI community. Therefore, RTS games have not been designed as synthetic domains where AI techniques can easily exhibit superior behaviors. These games are interesting as they are characterized by enormous state spaces, large decision spaces, and asynchronous interactions [Aha *et al.*, 2005]. RTS games also require reasoning at several levels of granularity, production-economic facility (usually expressed as resource management and technological development) and tactical skills necessary for combat confrontation. RTS games are even used in real-life applications such as military training systems.

The particular RTS game that we used in our research is MadRTS™, a commercial RTS game being developed for military simulations. Figure 1 shows a source task (left) and target task (right) for achieving Level 4 Transfer ("Extending") in MadRTS. The experimental results presented in this

Transfer Level	Description
0. Memorization	New problem instances are identical to those previously encountered during training. The new problems are solved more rapidly because learning has occurred.
1. Parameterization	New problem instances have identical constraints as Memorization, but have different parameter values chosen to ensure that the quantitative differences do not require qualitatively different solutions.
2. Extrapolating	New problem instances have identical constraints as Memorization, but have different parameter values that may cause qualitatively different solutions to arise.
3. Restructuring	New problem instances involve the same sets of components but in different configurations from those previously encountered during training.
4. Extending	New problem instances involve a greater number of components than those encountered during training, but are chosen from the same sets of components.

Table 1: Transfer Learning Levels 0-4 from a taxonomy of eleven levels.

Figure 1: A source task (left) and target task (right) for exhibiting Transfer Learning Level 4. This example is simpler than our actual experiments; however, it demonstrates the same changes in scale that characterizes "Extending". In our experiments, both the number of territories and the number of troops is increased—from three and five, respectively, in the source task to five and eleven in the target task—necessitating a qualitatively different set of tactical decisions.

paper were obtained with much larger maps and with higher numbers of troops, buildings and territories.

3 Approach

Because we are particularly interested in applying transfer learning to RTS games, we have designed an architecture suited to that purpose. In particular, we have implemented a multi-layered architecture that allows us to learn at both strategic and tactical levels of abstraction. In this section, we describe in detail that architecture as well as our approach to case-based reasoning and reinforcement learning.

3.1 System Architecture

CAse-Based Reinforcement Learner (CARL) is realized as a multi-layered architecture similar in spirit to other popular multi-tiered architectures [Albus, 1991]. Upper layers reason about strategy while lower levels are concerned with ever finer distinctions of tactics. The granularity and time scale of the actions generally reduces in the lower layers. Actions for an upper layer A_i are treated as goals by the layer immediately below. Each layer is conceptually identical, as shown in Figure 2. Each layer includes these three representative modules:

- The *planner* makes decisions about action selection at that layer based on the current state.
- The *controller* is a reactive module that provides perception/action interfaces to the layer below.
- The *learner* modifies the representation used by the planner *in situ*, splitting and merging actions, updating features about the actions and states, and so on.

For our experiments, we have defined three levels. The top-most strategic level is currently rather simple (based on a hand-coded strategy). The middle tactical layer uses a hybrid Case-Based Reasoner and Reinforcement Learner. Specifically, this layer makes tactical decisions over an action space of Attack, Explore, Retreat and Conquer Nearest Territory. The lowest layer incorporates a reactive planner, scripted to perform the tasks in a predefined manner specific to MadRTS™. The lower layer responds to the tasks delegated from the tactical layer as if they were goals for the reactive planner.

State Feature	Description
AvgHealth	Average health of the agent's troops that are currently alive
$\%S_p$	Agent's alive troops as a percentage of their initial strength
$\%S_o$	Opponent's troops killed as a percentage of their initial strength
$\%T_p$	Territories owned by the agent as a percentage of the maximum available in the scenario
$\%T_o$	Territories owned by the opponent as a percentage of the maximum available in the scenario

Table 2: Features representing state of the game at the Tactical layer of CARL. At any instance of time, a territory can be either owned by the agent, the opponent or neither of the two playing sides. A side owns a territory if it sends a constant number of troops to that territory and leaves behind a smaller fixed number of troops for patrolling.

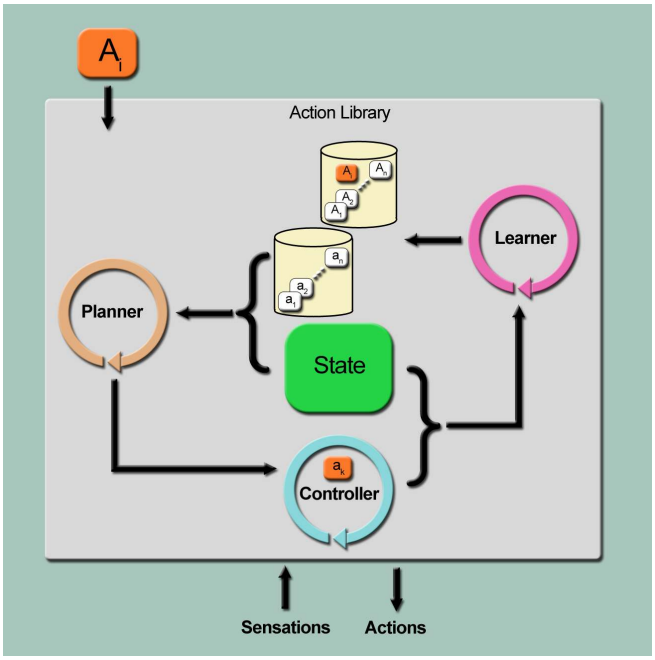

Figure 2: A representative layer in our Transfer Learning architecture CARL. The architecture contains several of these levels that are conceptually similar; the planner, learner, and controller are instantiated in different ways, depending upon the needs of that level.

3.2 Case-Based Reinforcement Learning

Reinforcement Learning (RL) is one natural approach for building interactive agents. In RL, problems of decision-making by agents interacting with uncertain environments are usually modeled as Markov Decision Processes (MDPs). In the MDP framework, at each time step the agent senses the state of the environment, and chooses and executes an action from the set of actions available to it in that state. The agent's action (and perhaps other uncontrolled external events) causes a stochastic change in the state of the environment. The agent receives a (possibly zero) scalar reward from the environment. The agent's goal is to develop an optimal policy that chooses actions so as to maximize the expected sum of rewards over some time horizon.

Many RL algorithms have been developed for learning good approximations to an optimal policy from the agent's experience in its environment. At a high level, most algorithms use this experience to learn *value functions* (or *Q-values*) that map state-action pairs to the maximal expected sum of reward that can be achieved starting from that state-action pair. The learned value function is used to choose actions stochastically, so that in each state, actions with higher value are chosen with higher probability. In addition, many RL algorithms use some form of *function approximation* (representations of complex value functions) both to map state-action features to their values and to map states to distributions over actions (i.e., the policy). See [Sutton and Barto, 1998] for an extensive introduction to RL.

Case-Based Reasoning (CBR) provides an approach to tackling unseen but related problems based on past experience [Kolodner, 1993; Leake, 1996]. It has been formalized into a four-step process: Retrieve, Reuse, Revise and Retain [Aamodt and Plaza, 1994]. One view of this work is that it uses CBR to perform online lazy learning of a value function using RL techniques to solve temporal credit assignment.

Case Representation

Individual cases must represent information about a region in the generalized state space and provide utilities for executing different actions in that region of the state space. In general, the i_{th} case, C_i, is represented as a tuple of four objects

$$C_i =< S_i, A_i, Q_i, e_i >$$

where

- State S_i is a vector composed of features representing game state that the system has already experienced.
- Action set A_i is a list of the possible actions the agent can take at that level in the architecture.
- Utilities Q_i is a vector of utilities for each action in A_i available for the situation described by S_i.
- Eligibility e_i is a real number reflecting the cumulative contribution of the case in previous time steps. Eligibilities are used to support TD(λ) updates during revision. ($0 \leq e_i \leq 1$)

It is important to note that the agent need not have performed a particular action on a particular state for the corresponding cases to provide suggestions for carrying out those actions, i.e. $\forall C_i \in$ Case Library, $\forall a_{ij} \in A_i, \exists q_{ij}$ where $j = 1 \ldots N$ and $N = |A_i|$.

Table 2 enumerates the features used in the experiments reported here. The features are chosen such that they represent the temporal changes in the behavior of the different sides in the game. Note that each feature is represented as a numeric value, so in our case $C_i \in \mathbf{R}^5$.

Retrieval
Because our features are numeric, we can use Euclidean distance as our similarity metric. k-NN$(S_q) = \{C_i \in$ Case Library $\mid d_i \leq \tau_k\}$, where S_q is the queried state, d_i is the similarity (Euclidean distance) of the state knowledge of every case C_i with queried state S_q and τ_k is a task-dependent smoothing parameter. A Gaussian kernel function (K(d_i) = exp$(-d_i^2/\tau_k^2)$) determines the relative contribution of individual cases. The utilities for a queried state is the weighted average of the utilities of the retrieved cases:

$$\hat{Q}_t(S_q, a_{ij}) = \sum_{\forall C_i \in k\text{-NN}(S_q)} \frac{K(d_i)}{\sum_{\forall C_h \in k\text{-NN}(S_q)} K(d_h)} u_{ij}$$

Reuse
The planner chooses the action with the highest expected utility with a probability of $1 - \epsilon$. To encourage exploration, ϵ begins as a large value but decreases monotonically as a function of the game trials and the number of decisions made by the case-based planner during those trials.

The action is passed as a goal for the lower layer to achieve. As noted earlier, the lowest layer is a scripted reactive planner that commands a series of primitive actions that directly affect the MadRTS game engine. For example, given the tactic *explore*, the lower planner would find a subset of idle troops and instruct them to carry out a set of exploratory actions as defined by the game engine's instructions.

Revision
Our revision phase uses RL-style updates to the utilities Q_i for actions A_i chosen by the agent. In particular, a temporal difference (λ) [Sutton, 1988] update ensures that previous errors are incorporated in future decisions.

$$\triangle Q_{ij} \leftarrow \alpha(r_{t+1} + \gamma \hat{Q}_{t+1} - \hat{Q}_t) e_i \quad \forall C_i \in \text{Case Library}$$

where $\hat{Q}_{t+1}$ and $\hat{Q}_t$ are the utilities of the actions chosen by the decision cycle of the agent for state representation S_i at time instances $t+1$ and t, respectively; r_{t+1} is a reward; e_i is an eligibility trace; α is the gradient-descent based utility update factor; and constant γ is the Q-learning discount rate. The reward achieved in each decision cycle is a linear combination of temporal changes in the state features. Eligibilities represent the cumulative contribution of individual state and action combination in the previous time steps. After factoring in ϵ, the exploration factor, the final action selected by the decision cycle has its eligibility increased in those cases that combined to form the k-NN. This helps perform corresponding utility updates based on the case's past contributions:

$$e_i \leftarrow \begin{cases} \gamma \frac{K(d_i)}{\sum_{\forall C_h \in k\text{-NN}(S_q)} K(d_h)} & \text{the chosen action} \\ \lambda \gamma e_i & \text{otherwise} \end{cases}$$

The remainder of the cases in the library receive a scaling down for eligibilities of all their actions by a factor of $\lambda\gamma$.

Retention
New cases are added to the case library only if the distance of the closest neighbor d_{min} is larger than the threshold parameter τ_d. Thus, τ_d acts as both a smoothing parameter and as a mechanism for controlling the density of cases in memory.

4 Evaluation and Analysis

We evaluate our hybrid CBR/RL transfer learning agent in the MadRTS™ game domain. Our evaluation focuses on examining the hypothesis that the agent should do well with learning a given task in MadRTS, and should also be able to learn better across tasks by transferring experience.

To quantify the learning that is being done by the agent on a single task, we compute a performance metric at the end of each completion of a single game on a particular map. The metric is computed as a linear combination of five features: average health of the remaining agent troops, time elapsed in the trial, enemy troops killed, agent's troops alive and territories owned by the agent as a percentage of their respective initial strengths. Given the learning curve T, representing the performance of the agent when acting directly on the target task and the transfer learning curve T/S, representing the performance of the agent after transferring the knowledge from the source task, we use the following three metrics to determine the quality of transfer learning:

- Jump Start - Difference in the mean value of the first ten data points (trials) of T from the mean value of the first ten data points of T/S
- Asymptotic Gain - Difference in the mean value of the last ten data points (trials) of T from the mean value of the first ten data points of T/S
- Overall Gain - Difference in the mean value of the all the data points (trials) of T from the mean value of all the data points of T/S

These, in fact, measure different parameters about the transfer. For example, it is possible for two curves to have the same overall gain but for the initial advantage of transfer (Jump Start) to be vastly different.

4.1 Experiments

In this section, we present the results of transfer at levels 3 and 4. All our experiments are carried out with large 64×64 tile maps, with at least six territories to conquer and at least nine troops fighting from each side. To exhibit transfer learning at level 4, the target scenario consisted of an increase in the map components (number of territories, enemy troops and buildings) of at least 1.5. The experimental setup for testing the transfer learning at level 3 involves mirroring the initial ownership of territories when moving from source scenario to target scenario. Part of the difficulty is that whereas in the source scenario, the agent's troops start with initial positions in adjacent territories (large army), in the target task, the new territories are separated by a significant distance coupled with geographical obstructions. This forces the agent to develop new sequences of tactics: forces are now two smaller battalions of troops rather than one large army.

A trial terminates only after all the agent's troops are dead or all the opponent troops are dead and the remaining agent troops have conquered the maximum number of territories possible (given their current strength). Conquering a territory requires that a troops need to reach the territory, kill any opponent troops, and leave behind a minimum number of troops.

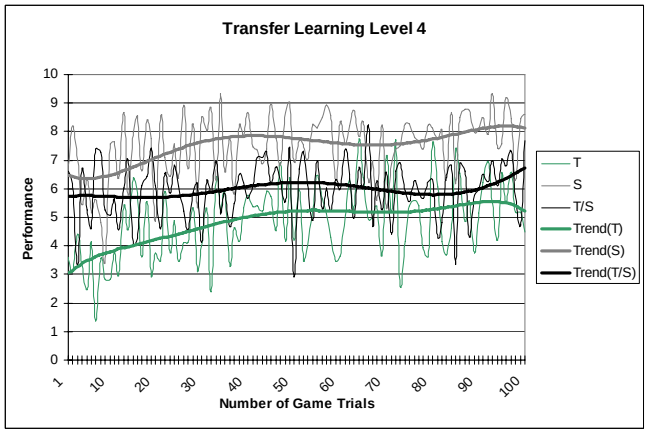

Figure 3: Transfer Learning on Level 4. T, S and T/S are the respective performance of the learning curves obtained as average over five iterations of the agent solving Target scenario alone, Source scenario alone and Target scenario after transfer of knowledge over the Source scenario. Trend(*) is a smooth fitting of the points in the curve *.

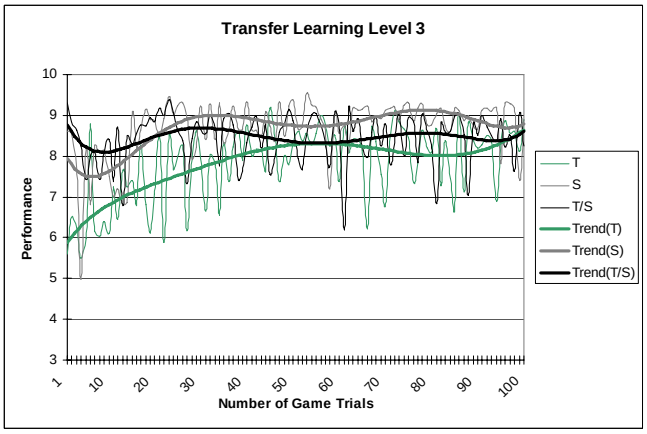

Figure 4: Transfer Learning on Level 3.

TL Metric	TL Level 4	TL Level 3
Jump Start	2.608*	1.995*
Asymptotic Gain	0.870*	-0.004
Overall Gain	1.118*	0.594*

Table 3: Results on Transfer Learning Performance Metrics for Levels 4 and 3. * indicates that the results are statistically significant with a confidence of at least 98%.

In these experiments, we use the value of the Q-Learning discount rate constant $\gamma = 0.9$, the Temporal Difference constant $\lambda = 0.6$, the exploration factor $\epsilon = 0.7$, gradient-descent based utility update factor $\alpha = 0.6$, and number of nearest neighbors $k = 5$. The task dependent smoothing parameter T_k is maintained larger than the density parameter T_d, enabling several cases to contribute to the value of the query point while, at the same time, ensuring that large portions of the domain are covered with fewer cases. We analyze the agent's performance by averaging over five independent iterations. Each task was executed for a hundred trials with the worst case running time of 60 seconds per trial.

4.2 Results

The results are shown in Figure 3 for TL level 4 and in Figure 4 for TL level 3. Curves T, S and T/S represent the agent's performance as a function of the completed game trials with learning only on the target scenario; learning only on the source scenario; and learning on the target scenario after learning on the source scenario, respectively. For clarity, we also show trend lines that are smooth fits for each curve. Both Trend(S) and Trend(T) show a continual increase in performance with S appearing easier than T. The vertical axis in Figures 3 and 4 is the performance of the agent (P) measured at the end of each trial (a game of just Source or Target task): $P = 2 * AvgHealth + 2 * \%S_o + 3 * \%S_p + 3 * \%T_p + 50/TimeElapsed$ (see Table 2). TimeElapsed is a quantity measured as a function of the time scale in the RTS game. Completion of each task is guaranteed to take more than 50 time units. Hence, the best that anyone (our agent, a hard-coded agent, random player) can do in each trial is a performance of 11.

Both Figures 3 and 4 show that T/S has a significant jump start and performs much better overall than T. Table 3 summarizes all of the metrics. Using the signed rank test, the confidence of the overall gain for transfer learning level 3 and 4 is 99.95%. All the resultant values, but one, have been verified to be at confidence of at least 98%. The only exception is the asymptotic gain of Level 3, suggesting that curves T and T/S have converged.

The results show effective jump start transfer learning. In Level 4 this appears to be because the agent learns when to exploit the *attack* tactic. This gave the agent a valuable method for performing well immediately, even for the increased number of enemy troops in the target scenario. Further, reusing the *conquering nearest territory* tactic from the existing case library worked well despite the larger number of territories. In case of TL level 3, the agent benefits from the learned use of the *explore* tactic. This proved useful with the player's troops starting as two geographically separated battalions. When acting without prior learning, the agent takes more trials to discover the use of that exploration tactic to unite smaller forces.

5 Related Work

There has been an increasing interest in transfer learning recently. For example, given a task hierarchy, Mehta *et al.* [2005] transfer value functions within a family of variable-reward Markov Decision Process (MDPs). Their primary objective was speedup learning, and demonstrated on a discretized version of an RTS game. Rosenstein *et al.* [2005] investigate whether transfer should be attempted when there is no prior guarantee that the source and the target tasks are sufficiently similar. To the best of our knowledge, there are no existing case-based reasoning approaches that address the problem of transfer learning.

CBR and RL have also been used in other domains involv-

ing real-time, continuous-valued attributes, including [Santamaria *et al.*, 1998] for robot navigation and [Gabel and Riedmiller, 2005] for robot soccer. Santamaria *et al.* use a case-based function approximator to learn and generalize the long-term returns of the agent across continuous state and action spaces. Unlike previous approaches, our approach explicitly decomposes the decision-making tasks into several levels of abstraction (motor, tactical, and strategy) and applies the hybrid CBR/RL decision-making algorithm at the tactical level.

Aha *et al.* [2005] provide one of the first CBR systems that can defeat dynamic opponents in an RTS. Unlike their approach, we did not require representation of state spaces using pre-determined lattices. Our system is capable of various levels of transfer learning across classes of problems with varying complexity in the application domain.

6 Discussion

We have presented an approach for achieving transfer learning using a hybrid case-based reasoning and reinforcement learning algorithm. Further, our multi-layered architecture (CARL) provides a useful task decomposition, allowing the agent to learn tactical policies that can be reused across different problem instances with similar characteristics, thereby accomplishing higher levels of transfer learning.

Our experiments demonstrate online transfer learning in the continuous state space of real time strategy games, using a real RTS game as a testbed. Specifically, we have shown that our agent's capability to perform transfer learning is not just limited to speeding up learning, but can also lead to either better or the same overall final performance in complex scenarios.

We plan to extend our architecture, replacing the lowest layer with a module that learns new actions and passes them up to the next layer. Similarly, the middle layer should pass new actions to the strategic level so that it may perform non-trivial reasoning. Our definition of tactics should also be expanded to include actions that change internal state variables. For example, one might imagine an action that changes the weights on the state features as used by the reward function. The system might learn the appropriate time to update the weights of the state features that signify troop strengths as a mechanism for implementing strategically important goals.

We would also like to refine the case-based approximator. For example, CBR would benefit from better similarity metrics (*e.g.* Mahalanobis distance) and a domain-specific indexing scheme for the large number of cases in its library.

Finally, we would like to solve tasks requiring more complex forms of transfer. For example, target tasks should be recognized as similar to a number of different source instances only when they are reformulated through a well-specified transformation that CARL would need to discover.

Acknowledgements

We would like to thank Santi Ontañón for valuable feedback and the anonymous IJCAI reviewers for a variety of suggestions on this paper. We acknowledge the support of DARPA under contract No. HR0011-04-1-0049. We also thank the Navy Research Laboratory for providing the interface TIELT used between CARL and the RTS game as well as game developers, MadDoc Software, for providing TIELT-Enabled MadRTS™.

References

[Aamodt and Plaza, 1994] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1):39–59, 1994.

[Aha *et al.*, 2005] David Aha, Matthew Molineaux, and Marc Ponsen. Learning to win: Case-based plan selection in a real-time strategy game. In *ICCBR*, pages 5–20, 2005.

[Albus, 1991] J. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):473–509, 1991.

[DARPA, 2005] DARPA. Transfer learning proposer information pamphlet (baa # 05-29). www.darpa.mil/ipto/solicitations/closed/05-29_PIP.htm, 2005.

[Gabel and Riedmiller, 2005] Thomas Gabel and Martin Riedmiller. Cbr for state value function approximation in reinforcement learning. In *ICCBR*, 2005.

[Kolodner, 1993] Janet Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, San Mateo, CA, 1993.

[Leake, 1996] David Leake. *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press/MIT Press, Menlo Park, CA, 1996.

[Mehta *et al.*, 2005] Neville Mehta, Sriraam Natrajan, Prasad Tadepalli, and Alan Fern. Transfer in variable-reward hierarchical reinforcement learning. In *NIPS'05 Workshop, Inductive Transfer: 10 Years Later*, 2005.

[Rosenstein *et al.*, 2005] Michael Rosenstein, Zvika Marx, Leslie Kaelbling, and Thomas Dietterich. To transfer or not to transfer. In *NIPS'05 Workshop, Inductive Transfer: 10 Years Later*, 2005.

[Santamaria *et al.*, 1998] Juan Santamaria, Richard Sutton, and Ashwin Ram. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, 6(2), 1998.

[Sutton and Barto, 1998] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[Sutton, 1988] Richard Sutton. Learning to predict by the methods of temporal difference. *Machine Learning*, 3:9–44, 1988.