Experiments in Transfer Across Multiple Learning Mechanisms

Nicholas A. Gorski NGORSKI@UMICH.EDU
John E. Laird LAIRD@UMICH.EDU
Computer Science and Engineering, University of Michigan, 2260 Hayward Street, Ann Arbor, MI 48109-2121 USA

Abstract

We present three different learning mechanisms for transferring spatial knowledge from one problem to another, related problem: memory-based transfer, search-based transfer, and transfer using reinforcement learning. We describe the approaches and present preliminary results demonstrating successful transfer using these approaches in Soar and testing them in the Urban Combat Testbed.

1. Introduction

The study of transfer in learning is typically restricted to variants of a single learning mechanism. When multiple learning mechanisms are studied, underlying reasons for any differences in performance can be unclear, obscured by differences in task knowledge and the underlying performance system. In this work, we study three different learning mechanisms, all within the same general cognitive architecture (Soar) and all performing the same tasks, minimizing differences outside of the learning systems. The three learning mechanisms that we investigated are: storing map knowledge in short-term working memory; using chunking to learn operator selection rules; and using reinforcement learning to tune operator selection rules.

We developed agents that use these learning mechanisms to solve simple navigation tasks within the Urban Combat Testbed (UT Arlington, 2006). The transfer tasks vary the position of the destination and introduce barricades that require finding new paths.

2. Soar Cognitive Architecture

Soar is designed to be a general cognitive architecture, capable of using a wide range of methods and knowledge to solve problems requiring the integration of many cognitive capabilities underlying general intelligence (Lehman, Laird & Rosenbloom, 2006). Soar encodes long-term procedural knowledge as production rules and uses working memory to store short-term declarative knowledge in a hierarchical graph structure. Rules in Soar propose, select, and apply operators, which in turn perform external actions or modify internal data structures. Impasses from inconsistent or incomplete knowledge lead to subgoals in which additional operators can be proposed, selected and applied. These subgoals can involve planning, hierarchical task decomposition or any other type of reasoning.

Traditionally, Soar has relied on two "learning" mechanisms: modifying elements in working memory; and creating new rules through chunking (Laird, Rosenbloom & Newell, 1986). Recently, Soar has been enhanced with the ability to adjust operator selection rules using reinforcement learning (Nason & Laird, 2005).

Adding and removing elements in working memory via rules is a fundamental feature of Soar. Elements in working memory are not forgotten unless explicitly removed; thus, working memory can be abused by using it to maintain long-term declarative knowledge.

Chunking creates new procedural knowledge encoded as rules when a result is created in a subgoal. The newly created rule summarizes the processing that led up to the creation of the result. When similar situations arise in the future, rules learned during the subgoal will fire before the subgoal even arises, eliminating the need for it entirely. Over time chunking eliminates internal problem solving and moves the system from reflective problem solving to more reactive decision making.

Reinforcement learning (RL) in Soar adjusts the expected reward for proposed operators. Rules that test features of the current state and operator generate numeric preferences for operators when they are proposed. These numeric preferences encode expected reward, and their values are summed during the decision making process to provide a composite prediction of the utility of each operator. Numeric preferences are updated using a variant of Sarsa(0) (Rummery & Niranjan, 1994) and are based on immediate observed reward as well as the discounted expected future reward of the next selected operator.

Previous experimentation has demonstrated Soar's capability to perform in many complex domains (Jones, Laird, Nielsen et al., 1999); including using many forms of learning (Steier, Laird, Newell et al., 1987). Although transfer within and across problems has been explored in Soar, previous work has concentrated exclusively on transfer using Soar's chunking mechanism.

Appearing in the *ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, Pittsburgh, PA.

3. Transfer Learning in Urban Combat

The Urban Combat Testbed (UCT) is a multi-player first-person shooter video game where the main activity (in the available preliminary scenarios) is moving through a map. Through the exposed interface, the agent perceives space as being partitioned into non-overlapping convex polyhedrons, of which there are on the order of 100 in a typical scenario. The agent senses the area (convex polyhedron) containing its current location as well as information such as the vertices of the area, objects contained within the area, and gateways to adjacent areas. Visibility is limited so that only objects in the current area can be perceived.

There are two conceptual levels of navigation for the agent: moving within areas and moving between them. Since areas are convex and completely sensed by the agent, the agent can move within areas directly without any problem solving. For inter-area navigation, the agent must either randomly explore (when it has no prior knowledge of the topology of the areas) or use transferred spatial knowledge to direct its behavior.

UCT contains scenarios specifically designed to test transfer learning. In UCT, a scenario consists of a pair of problems, the source and the target. The goal in all of the preliminary UCT scenarios is to move from a starting location to a second location in the map, containing a flag. In the source problems (and target problems solved without transferred knowledge from the source), the flag location and topology is unknown.

3.1 Approaches

There are three types of knowledge that can transfer from the source to the target problem: the topology of the map, shortest path information for navigating from area to area, and the location of the flag. As we shall see, different learning mechanisms acquire different subsets of these types of knowledge, leading to different levels of transfer.

3.1.1 Memory-based Transfer

Our first approach explicitly stores all spatial knowledge in working memory, including map topology, complete path information from each area to every other area, and the flag location. This requires extensive exploration and computation in the source problem to compute the path information, but eliminates all search in the transfer problem. Complete paths are not stored; rather, a “path element” from a given area consists of a destination area and the gateway in the given area that is on the shortest path to the destination area. To move to a destination area, the agent repeatedly traverses gateways leading to the destination; at no point does it store the complete path in memory. If the path information is absent, the agent uses a strategy of directed exploration in the environment.

The memory-based approach exploits Soar’s unlimited working memory. As there are approximately 10^2 areas in the UCT maps used in these experiments, working memory contains on the order of 10^4 elements when the map is fully explored. Having such a large number of working memory elements impacts the efficiency of the Rete matcher used in Soar, but the matcher is still sufficiently fast so as to provide real-time control in UCT.

3.1.2 Search-based Transfer

Rather than pre-computing paths in the source problem, another approach is to store the topology and flag location but not paths. To find the shortest path in the target problem, an agent can search through the transferred map. In Soar, this search arises from a series of impasses that occur because the agent doesn’t know which area to move to next. A side effect of this search is that chunking learns rules for each decision, effectively encoding a procedural memory of the path. When knowledge of the flag is available, the agent biases its internal search to first consider the gateway nearest to the assumed location of the flag. For this domain, this heuristic is extremely effective.

3.1.3 Reinforcement Learning Transfer

RL learns the expected value of moving into adjacent areas. This is encoded in Soar by having operator selection rules that test for the destination area of each operator. These rules are generated dynamically as the agent explores the world. Through repeated trials, the agent learns the value of moving to every area. The agent receives negative reward for every action not leading to the area containing the flag, and no reward when it reaches the flag; as operator preferences are initialized to 0, this biases the agent to search initially. In contrast to the other approaches, this approach does not involve explicitly storing the map topology or the location of the flag, both of which are implicit in the operator selection rules.

In the transfer tasks, the learning parameter for the agent is set to 1. A high learning rate serves to inform the agent that the domain may have changed and it should update its numeric preferences quickly so as to adapt. In a domain with stochastic actions, it would still be appropriate to initialize the learning rate to be high and then decay it over time; no such decay is necessary with the deterministic actions of our domain. Additionally a negligibly small exploration parameter is used in order to focus the agent on exploiting learned knowledge that it had transferred.

3.2 Scenario Descriptions

We tested our transfer learning approaches on three tasks in UCT. In each, the goal is to reach the flag – the agent is only competing against the clock and there are no adversaries. These tasks are designed to test the first three levels of transfer as defined by DARPA (2005).

The level 0 task tests transfer performance involving memorization; the source and target are identical. The level 1 task tests transfer across problems that are reparameterized; maps are identical across problems but the flag location changes. The level 2 task tests transfer

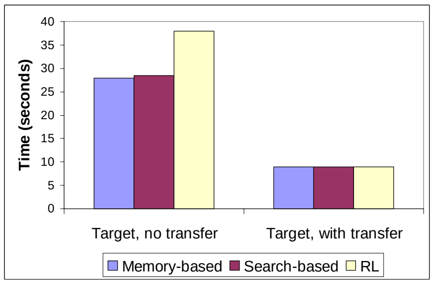

Figure 1, results for original level 0 scenario 2 in UCT (median of 40 trials).

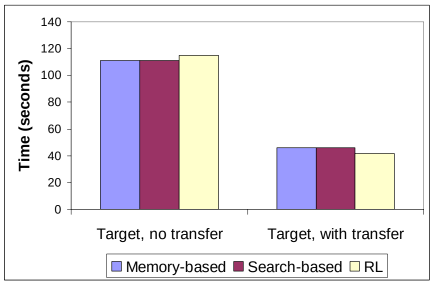

Figure 2, results for a modified level 1 scenario in UCT (median of 40 trials).

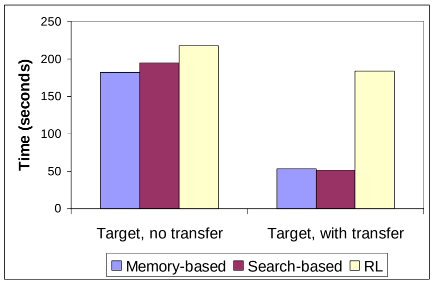

Figure 3, results for original level 2 scenario 1 in UCT (median of 40 trials).

requiring extrapolation; an obstacle is introduced in the target that qualitatively changes the optimal solution.

3.2.1 Implementation Details

At levels 0 and 1, the topology is identical in source and target problems so that route information transfers perfectly. In the original level 1 UCT scenario, the flag changes location within an area which is equivalent to level 0 for our agent. We created a more difficult scenario: the flag is moved three areas away from its original location. When the memory- and search-based agents reach the original flag location in the target, they initiate a breadth-first search of surrounding areas to find the flag.

At level 2, the flag location is unchanged but a gateway on the previously shortest path to the flag is blocked, which can be detected only from an adjacent area. The memory-based approach uses its stored path information to navigate up to the blocked gateway. Similarly, the search-based approach generates a path based on its stored map topology and follows it up to a blockage. Both approaches then initiate a model-based search to find and then follow a new path to the flag. The RL approach uses a high learning rate to quickly update the expected values of areas and force exploration to find a new path.

4. Evaluation

We evaluated our three approaches on the previously mentioned scenarios testing transfer at levels 0, 1, and 2 using two metrics. Given that T_T is time spent in the problem with transfer and that T_N is time spent with no transfer, a ratio of times indicating improvement is:

$$\text{Time ratio} = T_N / T_T$$

By measuring the best time in which a human expert could complete each task from a repeated set of trials, we allow for another ratio which uses this optimal time (O):

$$\text{Transfer ratio} = (T_N - O)/(T_T - O)$$

4.1 Level 0 (Memorization)

As shown in figure 1, transferred knowledge improves performance using all three approaches as expected. A human expert can complete this problem in 7 seconds.

4.2 Level 1 (Reparameterization)

Although level 1 is slightly more difficult than level 0, the agents perform similarly (figure 2). The implicit search performed by the RL agent serves the same role as the other agents' breadth-first search to find the new flag location. The breadth-first search should scale better as the distance between flags from source to target increases. A human expert can complete this problem in 17.4 seconds.

4.3 Level 2 (Extrapolation)

As seen in figure 3, the memory- and search-based approaches continue to show significant transfer at level 2. While the RL agent shows a slight performance gain from transfer, the qualitatively different path required in this scenario demonstrates that RL is weak when drastically different paths are required but the flag location is known. A human expert can complete this problem in 24.5s.

However, figure 4 shows that the RL agent does demonstrate transfer when measured over repeated episodes of the same task. These results were collected by simulating movement through the UCT map and thus measure the number of areas visited by the agent rather than elapsed time. With transfer, the agent performs near-optimally after approximately 10 episodes; without transfer, the same performance is not reached until the 40th episode.

The search-based approach also improves on a successive episode. In the first target episode, the agent performs a

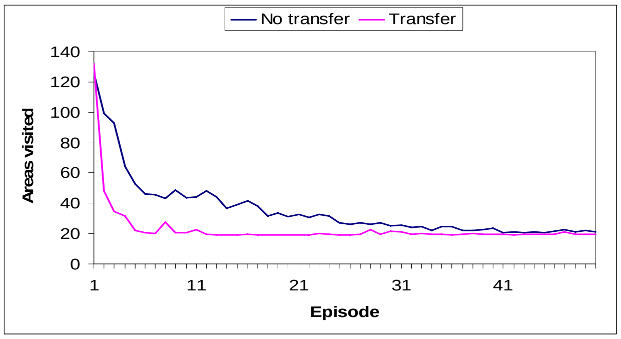

Figure 4, results of the RL agent on successive episodes in a simulated level 2 target problem. Results are averaged over 20 trials, each consisting of 50 episodes. Optimal is 19 areas.

model-based search and chunks the results. These chunks eliminate all search in the second episode, improving performance. As time spent internally reasoning is dominated by time spent moving in UCT, this improvement isn't significant; in other problem-solving domains it could be.

Table 1, Summary of time and transfer ratios at all levels.

METRIC	LEVEL	MEMORY	SEARCH	RL
TIME RATIO	0	3.1	3.2	4.2
TIME RATIO	1	2.4	2.4	2.7
TIME RATIO	2	3.4	3.8	1.2
TRANSFER RATIO	0	10.5	10.8	15.5
TRANSFER RATIO	1	3.3	3.3	4.0
TRANSFER RATIO	2	5.5	6.2	1.2

4.4 Further Discussion

There are a few questions that arise from these results. Why don't the agents ever achieve optimal performance? The emphasis in our agent design is on learning, and the agents' low-level movement is not optimized.

Why present median times? We present medians due to the long-tailed (and multi-modal) nature of the underlying distributions. While the results for agents not transferring in the target have high variance (with standard deviations as much as 100s or more), the medians are robust against high-valued outliers and present a fair and accurate depiction of the agents' observed behaviors. Results for agents using transferred knowledge have low variance.

Are there any differences between the memory- and search-based approaches? Even though the results for the search- and memory-based approaches are equivalent, they are so similar because the time to move through the environment dominates the time to reason internally. The results would be very different, and favor the memory-based approach, on pure reasoning tasks.

Why didn't RL do better on level 2? It transfers only what is required to perform well on a particular task. The other methods transfer the topology, paths, and flag location; the RL approach transfers only the expected value of moving to an area. Therefore, the former two should scale better on more complex tasks. However, the RL agent would natively handle domains with stochastic actions, while the others would need similar statistical knowledge to be competitive. The RL agent has the best time and transfer ratios for level 0, so isn't it the best? One oddity with these metrics is that they can reward inefficiency. In this case, it has the best scores because its performance is worst when not transferring.

What is the best method? While time and transfer ratios are informative, they are only part of the story for deciding which approach to use. Another consideration is the training required in source scenarios. The search-based approach requires several hundred seconds to exhaustively explore the map; the memory-based approach requires twice that time as it must enumerate all paths. The RL agent requires significantly more time: hundreds of repeated episodes of the source problem.

5. Future Work

Our first step will be to store all map information in Soar's newly created semantic memory system instead of working memory. The semantic memory is optimized for storing large bodies of declarative facts, while working memory is optimized for maintaining the current state of the agent and matching that against the system's rules. Beyond that we plan to explore additional learning mechanisms (episodic memory and hierarchical clustering) on the remaining DARPA transfer levels.

Acknowledgments

This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and managed by the Air Force Research Laboratory (AFRL) under contract FA8750-05-2-0283. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of DARPA, AFRL, or the United States Government.

References

DARPA (2005). *Transfer Learning* (BAA 05-29).

Laird, J.E., Rosenbloom, P.S., Newell, A. (1986). Chunking in Soar: The Anatomy of a General Learning Mechanism. *Machine Learning, 1*, 11-46.

Lehman, J.F., Laird, J., Rosenbloom, P. (2006). A Gentle Introduction to Soar, an Architecture for Human Cognition: 2006 Update. Retrieved May 2, 2006, from http://ai.eecs.umich.edu/soar/sitemaker/docs/misc/GentleIntroduction-2006.pdf

Jones, R.M., Laird, J.E., Nielsen, P.E., Coulter, K.J., Kenny, P.G., Koss, F.V. (1999). Automated Intelligent Pilots for Combat Flight Simulation. *AI Magazine, 20*, 27-41.

Nason, S., Laird, J.E. (2005). Soar-RL: Integrating Reinforcement Learning with Soar. *Cognitive Systems Research, 6*, 51-59.

Rummery, G.A., Niranjan, M. (1994). On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University.

Steier, D.S., Laird, J.E., Newell, A., Rosenbloom, P.S., Flynn, R., Golding, A., Polk, T.A., Shivers, O., Unruh, A., Yost, G.R. (1987). Varieties of Learning in Soar. *Proceedings of the Fourth International Machine Learning Workshop* (pp. 300-311). Irvine, CA.

UT Arlington CSE AI Research Group (2006) Retrieved March 23, 2006, from http://gameairesearch.uta.edu/