Analogical Learning in a Turn-Based Strategy Game

Thomas R. Hinrichs and Kenneth D. Forbus
Qualitative Reasoning Group, Northwestern University
2133 Sheridan Road
Evanston, IL, 60208
{t-hinrichs, forbus}@ northwestern.edu

Abstract

A key problem in playing strategy games is learning how to allocate resources effectively. This can be a difficult task for machine learning when the connections between actions and goal outputs are indirect and complex. We show how a combination of structural analogy, experimentation, and qualitative modeling can be used to improve performance in optimizing food production in a strategy game. Experimentation bootstraps a case library and drives variation, while analogical reasoning supports retrieval and transfer. A qualitative model serves as a partial domain theory to support adaptation and credit assignment. Together, these techniques can enable a system to learn the effects of its actions, the ranges of quantities, and to apply training in one city to other, structurally different cities. We describe experiments demonstrating this transfer of learning.

1 Introduction

When a novice learns how to play a strategy game, his initial focus of attention is usually on figuring out how things work. Long before building up a repertoire of competitive strategies, he is trying out basic actions, seeing their effects and discovering traps to avoid. We believe analogy plays a key role in this discovery process, especially in mapping across structurally different situations. Yet, analogy by itself cannot account for the very active process that learners go through in exploring a world and constructing explanations. In this paper, we show how experimentation strategies and qualitative reasoning can support planning and analogy in learning resource allocation tasks in a turn-based strategy game.

Games of this sort have several interesting properties: 1) they involve incomplete knowledge of the world, 2) they entail complex interrelationships between actions and observable quantities, 3) goals may be more like optimization problems than states to be achieved, and 4) planning and executing are tightly interleaved. Qualitative representations can serve as a partial domain theory to guide planning and learning at different levels of expertise.

In this paper, we describe how the integration of analogy, experimentation, and a qualitative model of city management can support planning and learning in the game of Freeciv [Freeciv, 2006]. In the rest of this section, we outline the context of this work, providing a brief synopsis of the Freeciv domain, our HTN planner, and analogical reasoning and experimentation. Next we describe how we are using a qualitative model of city economics to support credit assignment and avoid local maxima. We describe some experiments that demonstrate the ability to transfer and adapt prior training on different cities, and we close by discussing related work and future plans.

1.1 The Freeciv Domain

Freeciv is an open-source turn-based strategy game modeled after Sid Meier's series of Civilization™ games [Freeciv, 2006]. The objective of the game is to start a civilization from initial settlers in the Stone Age and expand and develop it until you either conquer the world or win the space race and escape to Alpha Centauri. In either case, the game can be characterized as a race to build your civilization and technological sophistication faster than your opponents. Along the way, there are many competing demands for limited resources, investment, and development. For example, players must improve terrain with irrigation and roads, while avoiding famine and military defeat. Too much emphasis on military preparedness, for example, can make citizens unhappy and therefore less productive. Money must be

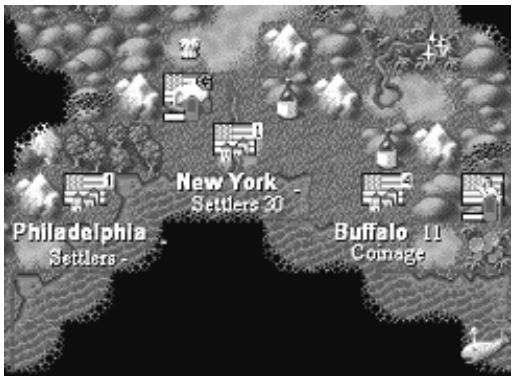

Figure 1: Freeciv

allocated to research into new technologies, such as iron-making and democracy, which enable players to create new improvements to cities, new types of units, and adopt new types of governments, each with their own tradeoffs.

In our current set of experiments, we focus on the subtask of managing cities to optimize their use of resources, build infrastructure, improve terrain, and research technologies. While our planner can direct exploration, city management tasks offer clearer evaluation metrics. We also currently ignore military operations, focusing instead on how to make a rich, productive civilization.

1.2 HTN Planning

To support performing and learning in the strategy game, we have implemented a Hierarchical Task Network (HTN) planner using the SHOP algorithm [Nau et al., 1999]. In the HTN planner, complex tasks are decomposed into primitive executable tasks. The primitives in Freeciv correspond to packets that are sent to the game server, representing actions such as sending a unit to a particular location or telling a city what to build. Complex tasks are at the level of figuring out what a unit should do on a particular turn, or deciding how to ameliorate a crisis in a city (such as a famine or revolt.) The planner generates plans for each unit and city at every turn and integrates them in a combined planning/execution environment. Planning is invoked partly in an event-driven manner, such that reified events from the game trigger certain decisions. For example, the planning agent does not re-compute its global strategy on every turn, but checks to see if it has acquired any new technologies in the last turn, and only then does it re-evaluate its strategy.

A critical aspect of this game is that it requires planning with incomplete and uncertain information. Terrain is not known until it is explored. The outcomes of some actions are stochastic, for example, village huts may contain barbarians that will kill an explorer, or they may contain gold or new technologies. There is also vastly more information in the game than can be considered within a planning state. Consequently, the planner cannot plan an agent's actions starting with a complete initial state. It must reify information on demand by querying the game state. At the same time, the planner may project the effects of actions such that the planned state deviates from the game state. To reconcile these competing demands, we maintain two contexts (cf., Lenat 1995): a game context that always reflects the incomplete, but correct current state of the game and a planning context in which states are projected forward. Every query for information that cannot be directly answered from the planned state proceeds to query the game state. Before returning such game-state information, it is checked for consistency against the plan state, to ensure that, for example, a unit is not believed to be in two places at the same time. This is a simple heuristic that checks for explicit negations and for inconsistent values of functional predicates.

In addition to reifying on demand, the other way we accommodate incomplete information is through second-order planning primitives. The `doPlan` primitive allows a plan to defer a decision until execution time, thus implementing a kind of conditional plan. Other domain-independent primitives include bookkeeping methods for updating facts in cases, and `doCallback` which suspends a plan until a condition becomes true, at which time it instantiates the plan with the new bindings and resumes execution. This can play an important role in evaluating delayed effects of actions.

1.3 Analogical Learning

A high-level goal of this research is to demonstrate how analogy and qualitative reasoning can support machine learning across increasingly distant transfer precedents. To do this, we are using the Structure Mapping Engine (SME) [Falkenhainer et al., 1989], the MAC/FAC retrieval mechanism [Forbus et al., 1995], and the SEQL generalization system [Kuehne et al., 2000] as core components. These analogical mechanisms are tightly integrated in the underlying reasoning engine and provide the mechanisms for retrieval, comparison, and transfer. The Structure Mapping Engine in particular not only assesses the similarity of a precedent to the current situation, but also projects the previous solution into the new case by translating entities to their mapped equivalents, in the form of candidate inferences. Thus, analogy provides the first level of adaptation automatically.

The unit of comparison and retrieval is a case, and in this approach, cases are not entire games (though some lessons can certainly be gleaned from that granularity), nor even entire cities. Instead, a case is an individual decision in the context of a particular city at a particular moment in time in a given game. For example, cases can capture a decision about what improvements to build, what tiles to work, and at the broader game level, what technologies to research. For each type of decision, there is a set of queries represented in the knowledge base that are designated as possibly relevant to making the decision. There is another set of queries that are relevant to capturing and assessing the case solution. When the decision is acted on in the game, a snapshot of the case is constructed before and after execution and stored in the game context. This case snapshot is used both for analyzing the effects of actions and supporting later analogical transfer.

For retrieval purposes, cases are organized into case libraries from which MAC/FAC can extract the most structurally similar precedent. As the planner attempts to learn decision tasks, it creates and populates libraries for each type of task. After executing a plan, the current relevant facts are queried and stored as a temporal sub-abstraction in the game context. When the result of the action is evaluated with respect to the performance goal, the case is added to the task-specific library of successful or failed cases, as appropriate. A case is considered successful if it both improves the goal quantity and the quantity meets any threshold requirements. By essentially "rewarding" actions that improve the goal quantity, this can be viewed as a type of learning by the method of temporal differences [Sutton, 1988]. However, the threshold provides an additional crite-

rion that helps prevent the accumulation of low-values cases that tend to leave the system stuck in local maxima. In other words, an action that improved a precedent from "terrible" to merely "bad" probably won't help much in the new situation. For maximizing goals, this initial threshold is simply "greater than zero", while for minimizing and balance goals, it is undefined.

One complication with mapping prior solutions to the new problem is that particular choices may not have a clear correspondence in the new problem. When this happens, the mapping process produces an analogy skolem denoting an unmapped entity. We resolve such skolems by collecting the facts that mention that entity in the base case, and treating them as constraints in the problem. We then pick randomly from those choices that satisfy the constraints. An important insight was that it is not always necessary to resolve all skolems in a previous plan, but only those corresponding to the current choice. So for example, if a previous plan was to move a worker from one tile to another, but the current problem is to allocate an available worker, then it is not necessary to resolve the worker's prior location.

1.4 Experimentation

While analogy can be a powerful technique for learning, there must first be cases to retrieve and compare. To bootstrap a case library and ensure a variety of cases, we guide empirical learning via explicit learning goals [Ram and Leake, 1995]. Some learning goals are typically provided at the beginning of a game, such as a goal to learn the effect of the action of allocating workers. Other learning goals may be posted as a byproduct of explaining failures, such as learning the ranges of quantities. These goals may persist across games.

The goal for learning the effects of actions determines how a decision task will be solved when there are insufficient analogical precedents or canned plans. The experimentation strategy it uses makes decisions randomly in order to generate the requisite variation and provide cases that better cover the decision space. This strategy is typically accompanied by additional learning goals to control parameters (by suppressing decisions) in order to try to learn one effect at a time. Such a trial and error approach tends to work best for simple low-level decisions.

Within the context of a single game, poor choices are not revisited blindly, but are recorded as nogoods. Farming the desert typically results in that tile labeled as a nogood. The problem with this is that for some difficult cities, the system may run out of choices, at which point it must revisit the nogoods, but having tried them, it can now order them by performance and choose the best of what remains.

Later, as successful cases accumulate, it becomes possible to solve problems analogically. Still, when analogical transfer fails or runs out of successful precedents, it falls back on experimentation.

2 Exploiting a Qualitative Model

A qualitative model is a partial domain theory that captures influences between quantities. One of the outstanding features of a game like Freeciv is the complexity of the quantitative relationships in the simulation engine. Understanding the relationships is a critical factor in playing the game well. Figure 2 shows a small fraction of our model of cities in Freeciv.

The primary way the qualitative model is currently used is to determine which changes incurred by an action correspond to goals and whether those changes signify success or failure. The model allows the system to trace from local tasks to global goals to assign credit and blame.

The second role of the model is in identifying leaf quantities that could affect outputs, and spawning learning goals as described in 'Overcoming local maxima', below.

The greatest potential role for qualitative models will be in synthesizing higher-level strategies by proposing primitive actions that influence goal quantities. This is on-going work that is outside the scope of this paper.

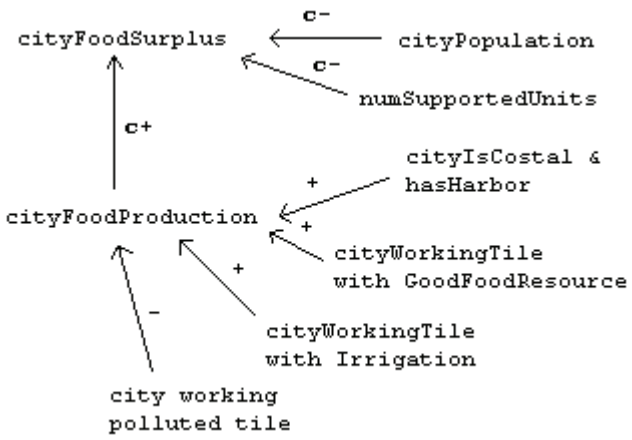

Figure 2: A portion of the city model

2.1 Overcoming local maxima

One of the difficulties in applying analogical learning to optimization problems of this type is that it is easy to fall into a local maxima, where the system performance stops improving and keeps adopting the same precedents over and over. We have two ways to overcome this:

First, when it fails to improve a goal, it attempts to explain why. By traversing the qualitative model, it collects the leaf quantities that ultimately affect the goal. It then posts a learning goal to learn the maximum or minimum values of these quantities (e.g., the food produced on individual tiles). Over several iterations, this provides a simple expectation about what should be attainable. The learner uses this information to estimate the improvement that should be possible by moving a worker from the least productive tile to a maximally productive tile. Then, depending on its current tolerance for risk, it sets the minimally acceptable threshold for its goal quantity. By raising the bar this way, it forces the learner to experiment some more to find a better solution. The tolerance for risk mentioned above is a function of the penalty that will be incurred on a bad decision. For example, when a city grows, its granary starts out

empty. Moving a worker to a less productive location can lead to a catastrophic famine if the granary is empty, but can be easily corrected if the granary is almost full.

The second method (not yet implemented) is by explicitly recognizing the lack of improvement. In the same way that a person can look at a learning graph and recognize a local maxima, so should the system. If the recent trend across games is flat or declining, this could serve as a spur to experiment more and be less satisfied with the existing cases.

3 Transfer Learning Experiments

In order to measure the generality of the learning mechanism, we performed a number of transfer learning experiments. Transfer learning involves training a performance system on one set of tasks and conditions and measuring its effect on learning in a different but related set of tasks and conditions. Three types of improvement are possible: higher initial performance (the "Y intercept"), faster learning rate, and/or higher final (asymptotic) performance. In the experiments we describe here, the system learned to allocate workers to productive tiles, while holding other potentially conflating decisions constant.

To understand the worker allocation task better, it is necessary to understand that cities in Freeciv can only benefit from working the 21 tiles in their immediate region, as shown in Figure 3. Some tiles are more productive than others, by virtue of their terrain type and randomly placed resource "specials". When a city is founded, it has one "worker" citizen available to work the land and produce food. If more food is produced than consumed, then the city's granary slowly fills over a number of turns. When it is full, the city grows by producing another worker that may be assigned to another tile. If, on the other hand, more food is consumed than produced, then when the granary is empty the city experiences a famine and loses a worker. The task we have set for the learner is to learn which tiles will most improve food production. The performance objective is to maximize the total food production at the end of 50 turns. Note that this is strongly affected by how fast the city grows, which in turn depends on making good decisions early and avoiding famine.

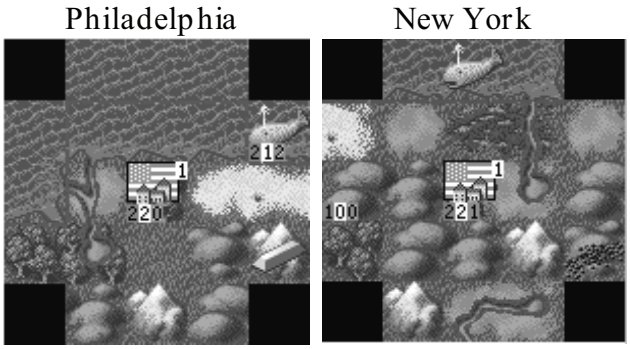

Figure 3: Freeciv Cities

In order to simplify evaluation, we controlled the city production queues to produce only coinage, and constrained the technology research agenda to work towards TheRepublic. Beyond this, it was also necessary to control for the game's unfortunate tendency to allocate workers for you whenever a city grows or shrinks. We did this by intercepting these changes in the low-level packet handler routines and undoing the game's interference before the learning agent ever sees it. In these experiments, we ran 10 training games on one city, "Philadelphia", followed by 10 games on a structurally different city, "New York" (See Figure 3). Each game stopped after 50 turns, which is approximately long enough to show some variation in performance.

Figure 4 shows the learning curves for the trials with prior training and those without. In this case, the trials with prior training have flatlined, because they are essentially at ceiling. The trials without prior training start out at 3 food points (the baseline performance) and rise to 5 after 5 games. This could be viewed as a 60% improvement in Y-intercept.

We next ran the same experiment with the cities reversed in order to determine how sensitive it is to the order of presentation. Here, there is an improvement from 5 to 6 food points, but games without prior training remain fixed at 5 points, while after the third trial, the games with training start to oscillate between 3 and 5 points. What is going on?

When we looked at the saved game cases, we saw that successful cases were transferred and applied both within

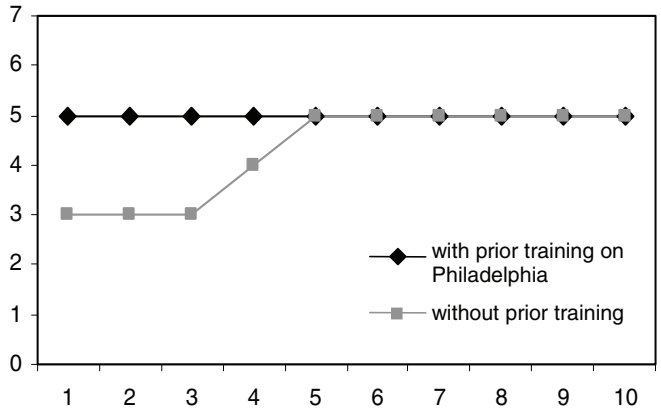

Figure 4: Food production after 50 turns (New York)

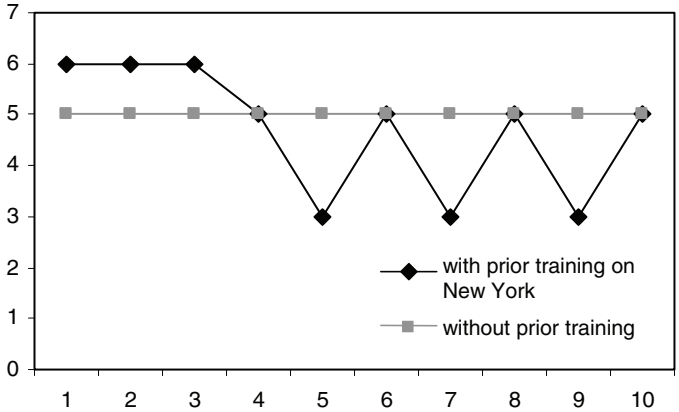

Figure 5: Food production after 50 turns (Philadelphia)

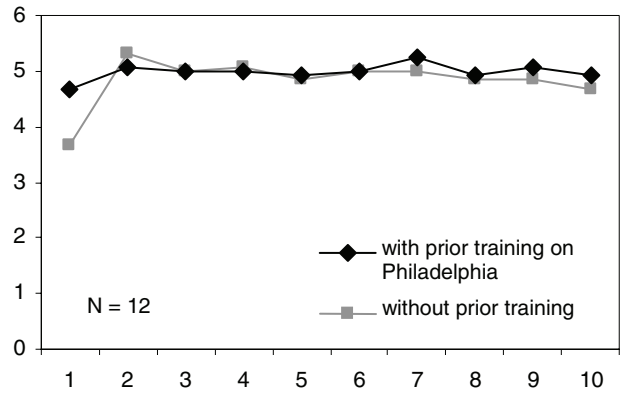

Figure 6: Learning from failure
(average food production in New York after 50 turns)

the same city and across cities. However, in cases with prior training, the system kept making the same mistakes over again. Specifically what was happening was that as it gained experience, it transferred successful cases early in the game, reaching a population of 5 or 6. Yet, there aren't typically more than 3 or 4 uniquely mappable highly productive tiles in a city. So above this size, the system fell back on random choice. This in itself is not terrible, except that the system had no facility for learning from prior failed cases, and so would continue to try farming the desert and suffer famine. Since this was not a case of negative transfer, we modified the experimentation strategy to first retrieve the most similar failed cases and map their choices into nogoods in the current case. We ran 12 sequences of 10 games in each condition and plot the average learning curves in Figure 6. These results show a 36% improvement in initial (Y-intercept) performance, with a P-value of 0.017 and a 95% confidence interval between 1.66 and 0.33, and are therefore statistically significant. We further analyzed the saved cases and determined that the incidence of famine dropped in half. Avoiding poor random decisions has the effect of converging more rapidly to an asymptote and reducing the variability in performance. This suggests that future learning experiments will require more complex, open-ended tasks.

4 Related Work

This work is part of a government-funded program on Transfer Learning. Other groups are pursuing similar goals in somewhat different ways. ICARUS [Langley et al., 2005] and SOAR [Nason and Laird, 2005] have both been applied to learning in real-time game environments, using Markov Logic Networks and Bayesian techniques, respectively. ICARUS has also been extended to include an HTN planner to generate hierarchical skills that can be executed in the architecture. Unlike ICARUS, our system does not learn hierarchical skills, but instead employs analogy to transfer instances of decisions.

The type of learning our system is doing could be viewed as a kind of reinforcement learning [Kaelbling et al., 1996], in so far as it involves unsupervised online learning and requires exploration of the space of actions. In reinforcement learning, successful actions are rewarded so that they will be preferred in the future. Here, successful actions are rewarded by adding them to the library of successful cases. Yet the emphasis is clearly different, because the process we have described is knowledge intensive, model-driven, and instance based. Derek Bridge [2005] has also explored the relation between reinforcement learning and CBR, in the context of recommender systems.

Qualitative models have been used in planning previously, notably Hogge's [1988] TPLAN, which compiled a qualitative domain theory into planning operators, Forbus' [1989] action-augmented envisionments, which integrated actions into an envisioner, and Drabble's [1993] EXCALIBUR planning and execution system that used QP theory with a hierarchical partial-order planner. The strategy game domain is more complex than any of the domains tackled in previous efforts. Our use of HTNs was inspired by Muñoz-Avila & Aha [2004], who used HTN planning in a real-time strategy game.

Prodigy/Analogy tightly integrated analogy with problem solving [Veloso, 1994]. Prodigy's core means-ends analysis strategy is perhaps more amenable to inferential tasks than the kinds of optimization goals we face. Liu and Stone [2006] also apply Structure Mapping to transfer learning in the domain of RoboCup soccer.

Other researchers have also used Freeciv as a learning domain. Ashok Goel's group at Georgia Tech has applied model-based self-adaptation in conjunction with reinforcement learning [Ulam et al, 2005]. We believe that analogy will better support distant transfer learning, and that qualitative models will ultimately permit strategic reasoning in ways that their TKML models will not.

5 Summary

This paper has presented initial results on integrating analogy, experimentation, and qualitative modeling in a system for planning, executing, and learning in strategy games. We suggest that qualitative models provide an intermediate level of domain knowledge, comparable to what a novice human player might start with. We described the use of analogy to compare before and after snapshots in order to extract the effects of actions. On the basis of experimental results, we implemented a minor change to the plans to enable learning from failures. This led to a pronounced improvement in behavior.

Clearly, learning resource allocation decisions is only a first step to learning and transferring abstractions and high-level strategies. We are investigating the role of qualitative models for composing new plans. We also intend to develop more elaborate strategies for pursuing learning goals.

This system does not currently construct explicit generalizations. Consequently, this can be viewed as a kind of transfer by reoperationalization [Krulwich et al., 1990], though by spawning learning goals in response to failures, we are part way to explanatory transfer. A next step for us will be to employ SEQL to construct generalizations and rules once the conceptual distinctions are sufficiently captured in the case base.

Another important near-term goal is to extract and reason about trends, deadlines and inflection points. This is a critical requirement for learning the early warning signs of impending problems and learning how to compensate for them before they become severe.

Ultimately, we expect the techniques developed from this effort to be applicable to the reflective control of agents in a Companion Cognitive System [Forbus and Hinrichs, 2006]. Strategizing and directing agents in a game is similar to the problem of coordinating computational resources, learning from interacting with a human partner, and maintaining an episodic memory of previous reasoning sessions.

Acknowledgments

This research was supported by DARPA under the Transfer Learning program. We thank Phil Houk, Jon Sorg, Jeff Usher, and Greg Dunham for their programming contributions.

References

[Bridge, 2005] Derek Bridge. The Virtue of Reward: Performance, Reinforcement and Discovery in Case-Based Reasoning. Invited talk presented at ICCBR 2005.

[Drabble, 1993] Brian Drabble. EXCALIBUR: A Program for Planning and Reasoning with Processes. Artificial Intelligence 62(1)1-40.

[Falkenhainer et al., 1989] Falkenhainer, B., Forbus, K., and Gentner, D. The Structure-Mapping Engine: Algorithm and Examples. Artificial Intelligence 41(1):1-63, 1989.

[Forbus, 1989] Kenneth D. Forbus. Introducing Actions into Qualitative Simulation. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. Detroit, MI. pp. 1273-1278, 1989.

[Forbus et al 1995] Kenneth D. Forbus, Dedre Gentner, & Keith Law. MAC/FAC: A model of similarity-based retrieval. Cognitive Science 19:141-205.

[Forbus and Hinrichs, 2006] Kenneth D. Forbus and Thomas R. Hinrichs. Companion Cognitive Systems: A step towards human-level AI. AI Magazine, vol. 27(2):83-95

[Freeciv, 2006] Freeciv official website http://www.Freeciv.org/.

[Gentner, 1983] Dedre Gentner. Structure-Mapping: A theoretical framework for analogy, Cognitive Science 7(2):155-170, 1983.

[Hogge, 1988] John C. Hogge. Prevention techniques for a temporal planner. Proceedings of the Seventh National Conference on Artificial Intelligence, pages. 43-48.

[Kaelbling et al., 1996] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research 4:237-285, 1996.

[Kamps and Peli, 1995] Jaap Kamps and Gábor Peli. Qualitative Reasoning beyond the Physics Domain: The Density Dependence Theory of Organizational Ecology. Proceedings of QR95, pages 114-122, 1995.

[Krulwich et al., 1990] Bruce Krulwich, Gregg Collins, and Lawrence Birnbaum. Cross-Domain Transfer of Planning Strategies: Alternative Approaches. In Proceedings of the Twelfth Annual Conference of the Cognitive Science Society, pages 954-961, 1990.

[Kuehne, et al., 2000] Sven Kuehne, Kenneth D. Forbus, Dedre Gentner, and Bryan Quinn. SEQL: Category learning as progressive abstraction using structure mapping. In Proceedings of the Twenty Second Annual Conference of the Cognitive Science Society, pages 770-775, August, 2000.

[Langley et al., 2005] Pat Langley, Dongkyu Choi, and Seth Rogers. Interleaving Learning, Problem-Solving, and Execution in the ICARUS Architecture. Technical Report, Computational Learning Laboratory, CSLI, Stanford University, CA. 2005.

[Lenat, 1995] Douglas B. Lenat. CYC: A large-scale investment in knowledge infrastructure. Communications of the ACM 38(11):33–38, 1995.

[Liu and Stone, 2006] Yaxin Liu and Peter Stone. Value-Function-Based Transfer for Reinforcement Learning Using Structure Mapping. In Proceedings of the Twenty-First National Conference on Artificial Intelligence, pages 415-420, Boston, MA, 2006.

[Muñoz-Avila, and Aha, 2004] Hector Muñoz-Avila and David Aha. On the Role of Explanation for Hierarchical Case-Based Planning in Real-Time Strategy Games. In Proceedings of ECCBR-04 Workshop on Explanations in CBR, 2004.

[Nason and Laird, 2005] Shelley Nason and John E. Laird. Soar-RL, Integrating Reinforcement Learning with Soar. Cognitive Systems Research, 6(1), pp.51-59, 2005.

[Nau et al.,1999] Dana S. Nau, Yue Cao, Amnon Lotem, and Hector Muñoz-Avila. SHOP: Simple hierarchical ordered planner. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, pages 968-973, 1999.

[Ram and Leake, 1995] Ashwin Ram, and David Leake, eds., Goal-Driven Learning, MIT Press / Bradford Books, Cambridge MA. 1995.

[Sutton, 1988] Richard S. Sutton. Learning to Predict by the Methods of Temporal Differences. Machine Learning, 3:9-44, 1988.

[Ulam et al., 2005] Patrick Ulam, Ashok Goel, Joshua Jones, and William Murdoch. Using Model-Based Reflection to Guide Reinforcement Learning. IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games. Edinburgh, 2005.

[Veloso, 1994] Manuela Veloso. Planning and Learning by Analogical Reasoning. Lecture Notes in Artificial Intelligence No. 886. Springer-Verlag Berlin, 1994.