An Empirical Evaluation of Automated Knowledge Discovery in a Complex Domain

Jay H. Powell
Indiana University
Dept. of Computer Science
Bloomington IN 47405, U.S.A.
jhpowell@cs.indiana.edu

John D. Hastings
University of Nebraska at Kearney
Dept. of Computer Science & Information Systems
Kearney NE 68849, U.S.A.
hastingsjd@unk.edu

Abstract

Automatically acquiring knowledge in complex and possibly dynamic domains is an interesting, non-trivial problem. Case-based reasoning (CBR) systems are particularly well suited to the tasks of knowledge discovery and exploitation, and a rich set of methodologies and techniques exist to exploit the existing knowledge in a CBR system. However, the process of automatic knowledge discovery appears to be an area in which little research has been conducted within the CBR community. An approach to automatically acquiring knowledge in complex domains is automatic case elicitation (ACE), a learning technique whereby a CBR system automatically acquires knowledge in its domain through real-time exploration and interaction with its environment. The results of empirical testing in the domain of chess suggest that it is possible for a CBR system using ACE to successfully discover and exploit knowledge in an unsupervised manner. Results also indicate that the ability to explore is crucial for the success of an unsupervised CBR learner, and that exploration can lead to superior performance by discovering solutions to problems which would not otherwise be suggested or found by static or imperfect search mechanisms.

Introduction

For problem domains that are either small, well understood, or static, it is usually possible in a relatively timely manner to manually encode the knowledge (e.g. in the form of search-based algorithms, neural nets, or static case bases) necessary for an artificially intelligent system to reason successfully. For the most interesting domains though (i.e. those which are exceptionally complex or dynamic, such as robot navigation in the real world), manually encoding such knowledge is impractical because either significant portions of the domain may change at unknown intervals (which would require timely knowledge updates), the rules of interaction within the given domain are poorly understood, or the knowledge required to succeed within the domain is mostly or entirely unknown. For an intelligent system to succeed within such domains, it is only natural to assume that it must be able to discover and exploit new knowledge, and to adapt to its changing environment as autonomously as possible. Furthermore, for such a system to make the leap into a general purpose, domain-independent reasoner, and closer to a more idealized form of artificial intelligence, when presented with a problem, such a system should be able to properly identify the domain and apply a capable reasoning technique for that domain. As nice as that sounds, just the more targeted goal of automatically discovering knowledge within a specific, but complex or dynamic domain remains a very challenging, non-trivial problem.

One methodology currently under development which aims to address the issues of automatically acquiring knowledge in complex domains is called automatic case elicitation (ACE) (Powell, Hauff, & Hastings 2005). ACE is a learning technique in which a case-based reasoning (CBR) system acquires knowledge automatically and without supervision through repeated real-time exploration and interaction with its environment. Test results in Powell et al. (Powell, Hauff, & Hastings 2005) suggest that in the domain of checkers, experience gained through ACE can substitute for the inclusion of pre-coded model-based knowledge, and that exploration of a problem domain is crucial to the performance of ACE. Further testing in the domain of chess (Kommuri, Powell, & Hastings 2005) revealed that experience alone, without the ability to adapt for differences between new and previous cases, is insufficient in more complex domains. To help decrease the complexity of the state space of chess, the research described in this paper introduces an alpha-beta search algorithm (Slagle & Dixon 1969; Knuth & Moore 1975) into the ACE framework. To avoid an over-reliance on alpha-beta search, while still encouraging exploration, the alpha-beta search was implemented to search only to a shallow depth of 3. To further encourage exploration, alpha-beta search was relaxed so that it would return a set of recommended chess moves, from which one move could be selected at random. In testing the combination of ACE and alpha-beta search in the domain of chess, our results suggest that experience gained by ACE through the process of exploration, with a case memory refined through reinforcement learning, can lead to an improvement over alpha-beta search alone. The results further suggest that relaxations to the case matching component in order to encourage exploration can lead to additional improvements.

Section 2 makes an argument for automated knowledge discovery utilizing case-based reasoning. Section 3 gives a brief description of the ACE algorithm. Section 4 sets forth an experimental evaluation in which several versions of ACE were evaluated against alpha-beta search. Section 5 details the results of our experimentation, in which we show that ACE combined with alpha-beta search consistently outperforms non-learning alpha-beta search. Section 6 closes with a discussion of related work.

A Case for Automated Knowledge Discovery

Realistically, a full-blown autonomous reasoner (at least one based on a human) would be able to automatically acquire and exploit knowledge using a variety of reasoning techniques at its disposal. We believe that the key to this idea begins with the automatic acquisition of knowledge in a simple and generically processable form. Case-based reasoning systems are particularly well suited to the tasks of both knowledge discovery and knowledge exploitation, due to the ease with which they can identify a novel situation (i.e. one that is not in the case base) and store that scenario for further reuse. A particularly rich set of highly successful methodologies exist for properly exploiting the knowledge within a case base (Kolodner 1993; Aamodt & Plaza 1994). However, within the CBR community, there appears to have been little research into methodologies that would facilitate the development of a system that could automatically acquire knowledge in an unsupervised manner (Patterson *et al.* 1999; Althoff & Weiss 1991).

One method for automatically discovering knowledge within a CBR framework is automatic case elicitation (ACE). Due to ACE's core philosophy of not requiring pre-coded domain knowledge for its primary reasoning and discovery processes, the ultimate goal is to have ACE be a general methodology for automatic knowledge discovery in complex or unknown domains for which an ACE system can explore its domain and receive feedback on actions that it has taken. Ideally, a system utilizing ACE would make use of generic, domain-independent matching and adaptation algorithms. However, at the writing of this paper, the issues of implementing domain-independent matching and adaptation algorithms in ACE have yet to be fully explored. Despite the lack of attention to date to case matching or adaptation within ACE, we do believe that the capabilities of CBR can be directly extended through the techniques described in this paper to acquire vast amounts of useful knowledge in an autonomous, unsupervised manner. Eventually, we hope to demonstrate the applicability of automated knowledge discovery through ACE to domains for which a comprehensive set of rules or knowledge governing optimal behavior have yet to be discovered.

The research described in this paper does not attempt to solve the matching or adaptation issues. Instead, it aims to determine the value of experience, specifically in the form of cases which might be acquired through ACE's process of exploration, within the domain of chess. Given that ACE worked well in checkers, but not chess, one could infer that ACE would be equally applicable to chess if the complexity could be reduced. Previously, ACE has retrieved only those cases from its library which exactly match the current situation, and has otherwise turned to random actions. Given the complexity of chess (i.e., the massive state space), ACE was seldom able to exactly match new board situations to its stored experience and thus routinely turned to random moves, resulting in less than optimal performance. In order to reduce the complexity (i.e., to reduce the number of possible actions that the system must explore for any given state using trial and error), this research replaces random move selection with a shallow alpha-beta search. Although this change conflicts with the ACE philosophy of not relying on pre-coded domain knowledge, we feel that it is necessary in the short-term simply to gain a better feel for the contribution of experience within the domain of chess.

Automatic Case Elicitation

This paper does not intend to provide in-depth details of the ACE algorithm, for that, the reader is referred to Powell et al. (Powell, Hauff, & Hastings 2005). However, briefly ACE can be described as a learning technique in which a CBR system, coupled with reinforcement learning (Kaelbling, Littman, & Moore 1996; Sutton & Barto 1998), automatically acquires knowledge in the form of cases from scratch during real-time trial and error interaction with its environment without reliance on pre-coded domain knowledge (e.g. rules or cases). ACE does not utilize separate training and testing phases, but instead continually improves its performance through repeated exposure to its environment. For implementation purposes, a case contains an observation or snapshot of the environment, the action taken in response to the observation, and a rating of the success of the applied action in meeting a goal. ACE operates on the sequence of observations (O_1 through O_n) made during interaction with the environment and completes at the point at which the effectiveness of the interaction can be determined (e.g. in chess, the effectiveness of an interaction will be determined at the completion of a game). For each observation of the environment (O_i), the system selects and applies actions suggested by the case library until a change in the environment is observed. In referring to the case library, ACE loops through the set of cases which match the current situation (in decreasing order by rating) and selects the action recommended by the first case whose rating is above a randomly chosen value between 0.0 and 1.0. If the set of matching cases is exhausted (meaning that no cases were selected or that the current situation is sufficiently distinct from previous experience), ACE applies a sequence of new actions at random (ones which might be entirely ineffective) until a change in the environment is observed.

At the end of each interaction, a probabilistic reinforcement learning approach rates the effectiveness of each case (acquired or stored) based on the final success of the interaction, providing a means for an ACE system to learn and improve from experience. New cases begin with a rating of 0.5 and tend either toward 1.0 (highly successful) or 0.0 (completely ineffectual) as the system gains experience.

Experimental Methodology

In complex domains there is a certain inherent tradeoff between knowledge and search (Simon & Schaeffer 1992). Effective search algorithms require a rigorous understanding of the rules of interaction of a domain. The amount of search that can be performed in many domains is often limited by the current state of hardware technology (Simon & Schaeffer 1992). For domains that have complex rules of interaction or sufficiently large state spaces, search algorithms can either be difficult to program or may take a significant amount of time to run. For other domains, the rules of the environment may be entirely unknown, rendering the use of search impossible.

Automated reasoning systems that rely on knowledge have one important drawback as well. The process of compiling and encoding knowledge can be difficult and labor intensive. Without a properly developed knowledge base, the effectiveness of a system can be greatly diminished. Often, it may not be known a priori what knowledge is needed for a system to succeed within its environment, suggesting that a system must be able to automatically acquire its own knowledge during its lifetime. To demonstrate the need for automated knowledge acquisition, and the ways in which ACE acquires knowledge automatically, we tested three modified implementations of ACE, or MACE (Modified ACE) for convenience, each initialized with empty knowledge bases, in the domain of chess. The experiments were designed to determine the value of experience in the domain of chess through a combination of ACE and alpha-beta search, the effects of relaxing the strictness of the case matching mechanism, the effects of adding randomness/unpredictability to alpha-beta search, and the effect of learning from an opponent in a complex game environment such as chess.

Two variations of alpha-beta search were used in the experiments. The first form of alpha-beta search (AB) employs the traditional alpha-beta search algorithm with a depth of 3. The second version of alpha-beta search (ABR) also uses a search depth of 3, but instead of statically picking the first top-ranked move like the traditional implementation does, it creates a set of all the top-ranked moves (in the event of a tie), and randomly selects a move from the set. Note that for shallow search depths, it is common for several moves to tie for the top ranking. The reasons for including both versions of alpha-beta search in our testing relate to what the authors perceive to be the deficiencies inherent in search-based algorithms. Search-based algorithms that are entirely rule based have their effectiveness limited by the quality of their rules, heuristics, and evaluation functions. Unless the rules used by such a search based algorithm are optimal, the solutions computed by such algorithms in their domains will be suboptimal. To emphasize the limitations of rule-based search algorithms, we decided to relax the constraints of our implementation of alpha-beta search to randomly select from a list of highly rated moves. The idea behind this approach is that truly optimal moves might be discovered when an algorithm such as alpha-beta search is used to narrow down the state space of a complex domain and suggest a list of possible successful moves. When combined with ACE, alpha-beta search with random move selection helps to guide the process of knowledge acquisition, instead of using knowledge to guide rule-based search algorithms.

Three configurations of MACE were tested in repeated competitions against the two variations of alpha-beta search (AB and ABR). The first configuration of MACE, called MACE1, employs exact matching (i.e., only cases which exactly match the current situation are considered). MACE2 relaxes matching and considers stored cases which differ from the current situation by no more than two distinct board positions. In addition to recording the consequences of their own actions, both MACE1 and MACE2 also capture and contribute intermediate observations about the environment (i.e., intermediate board states and chess moves made by an opponent) to the case base. MACE3 uses exact matching, but does not contribute opponent moves to its case base.

Approach	Description
AB	Alpha-beta search (depth = 3)
ABR	Alpha-beta search (depth = 3) with random top move selection
MACE1	Modified ACE with exact matching, learned opponent moves, and ABR
MACE2	Modified ACE with relaxed matching, learned opponent moves, and ABR
MACE3	Modified ACE with exact matching, no learned opponent moves, and ABR

Table 1: Brief description of the various reasoning approaches evaluated in the experiments.

Each of the variations of MACE is backed by ABR and began play with empty knowledge bases. The addition of ABR to ACE requires the following slight change to the action selection algorithm described in section 3: when the set of matching cases is empty (i.e., the system has encountered a novel situation), the action selection algorithm invokes the ABR algorithm, which returns a list of top-ranked moves, and then an action is randomly selected from this set. This step differs from previous implementations of ACE which generated a new random action at this point.

The reasoning systems played each other through XBoard (a graphical user interface for chess) using a modified version of code distributed with TIELT (Aha & Molineaux 2004). The first set of experiments involved running a series of 10 chess matches with 50 games per match. The matches paired { {MACE1,MACE2,MACE3} x {AB,ABR} }. The averaged results of the first set of experiments appear in Figure 1. The second set of experiments with the same pairings involved running a series of 10 chess matches with 500 games per match. The second set of experiments also included a pairing of ABR against AB in order to provide a point of comparison for the performance of the MACE agents. The results for the second set of experiments appear in Figures 2 and 3. For the results illustrated in Figure 2, draws were counted as 0.5 win and 0.5 loss.

Results

As illustrated in Figures 1 and 2, MACE was most effective in its play against the static AB player. For short matches of 50 games, MACE1 produced the most success against AB. However, for longer matches MACE2 and MACE3 were superior to MACE1. It appears that learning from the opponent and retrieving and applying exactly matching cases performs well when the learning time is limited, but is not the best approach when the period of interaction is extended. In analyzing a trace of MACE2 in its matches against AB, it appeared that MACE2 was less effective in short matches because the relaxed matching mechanism often retrieved dissimilar cases which were then applied out of context. Obviously, this suggests that cases which were close lexical matches were not always good matches. However, given a longer period of exploration and experience, MACE2 was able to successfully learn how to retrieve and apply non-exactly matching cases. The fact that MACE2 demonstrated the top performance in longer matches is a key result and suggests that the matching component can be relaxed or weakened to a certain extent, and that a period of exploration and refined experience can overcome early misapplications of actions and ultimately lead to successful actions and solutions not originally provided by alpha-beta search. This finding further validates the claim that an exploratory agent can successfully find and exploit actions which would not be suggested by pre-compiled search mechanisms.

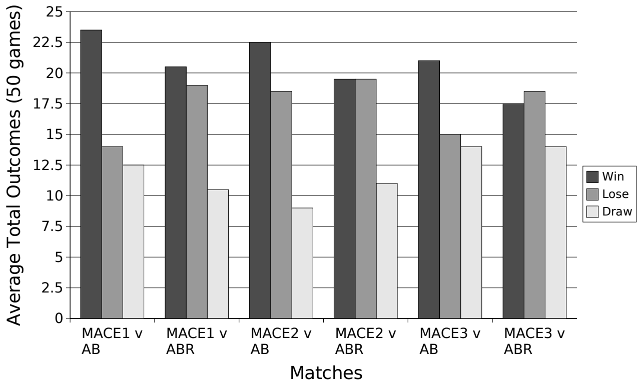

Figure 1: Averaged results of a series of 10 chess matches with 50 games per match.

For long matches, it is possible that MACE3 is more successful than MACE1 either because MACE3 is not attempting to learn from an inferior opponent or because the technique used to record and reuse opponent moves in MACE1 needs additional thought.

Notice in Figure 2 that the MACE players held a large advantage over the ABR player in the matches against AB. In addition, ABR was only slightly better than AB on average, indicating that the MACE players were not receiving a distinct advantage against AB simply by making use of ABR algorithm.

MACE was less effective against the randomization employed by ABR. This result is not surprising given that a more unpredictable opponent would require a player to find not one, but possibly several winning strategies. In short matches, MACE1 performed the best against ABR, although the average margin of victory was narrow. MACE1 was followed in performance by MACE2 and MACE3. The performance of the three configurations of MACE versus ABR in longer games were very similar to each other (and therefore not included in Figure 2), with MACE2 performing best followed by MACE3, and MACE1. Note that the ranking of the MACE configurations in long matches versus ABR mirrors that seen in the long matches against AB.

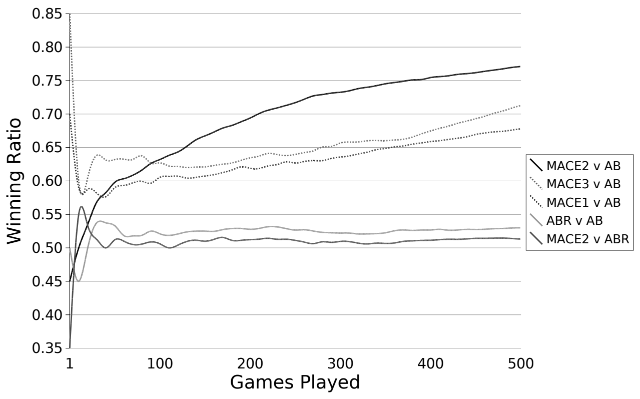

Figure 2: Averaged winning ratios of MACE1, MACE2, and MACE3 against Alpha-Beta Search in a series of 10 chess matches with 500 games per match.

Closer inspection of the 500 game matches reveals some interesting tidbits not illustrated in the summarized figures. Configurations that had the greatest success were most adept at finding move sequences that would guarantee wins for the remainder of the games in a match. Given that XBoard alternates the colors/sides of its opposing players, the most successful players had to find two different winning sequences per match. In the matches between MACE and AB, MACE2 and MACE3 were each able to find winning sequences for both colors in five of the ten matches, while MACE1 found both winning sequences in only three matches. Furthermore, MACE3 was able to find winning sequences for at least one color in all ten matches, while MACE2 found a sequence in nine matches, and MACE1 in eight. Overall, the lengths of the winning sequences ranged from four moves up to 33 moves. MACE2 found twelve different sequences, followed by MACE3 with ten, and MACE1 with eight. Interestingly, in just one match, MACE3 found and continually reapplied two alternate winning sequences when playing as black, and one sequence when playing as white. In terms of the speed with which the approaches were able to find at least one winning sequence, MACE3 was the fastest, finding a sequence in one match in the 10th game, followed by MACE2 in the 21st game, and MACE1 in the 41st game. None of the MACE players were able to find a guaranteed winning technique when matched against the ABR players.

Figure 3 illustrates the case-use ratios for each version of MACE against both ABR and AB. The case-use ratio is calculated as the total number of times that a CBR system utilizes actions suggested by its case base divided by the total number of actions attempted. The total number of actions attempted is the sum of the total number of applied CBR actions and the total number of actions applied to novel situations (e.g., random actions or actions suggested by an alternate reasoner such as alpha-beta search). The case-use ratio is a simple means for illustrating the degree with which a CBR system utilizes its experiences. In addition, our experience has shown case-use ratio to be a good indicator of the performance of an ACE system. As demonstrated in Figure 3, the utilization of the case base increased dramatically for each configuration of MACE throughout the matches with the regular AB players, increasing most dramatically during the first seventy-five games, suggesting that ACE agents are able to quickly adapt to the strategies of new opponents. Of the three MACE configurations, the case-use ratio of MACE2 was by far the best. Notice that the rankings of the case-use ratios for the MACE players versus AB mirrors the rankings of their winning ratios. The higher case-use ratio for MACE2 can be attributed in part to the relaxed matching mechanism which makes it more likely that a "matching" case will be found, but is also a product of MACE2's superior performance. MACE3 ranked higher than MACE1 in matches against AB, although their rankings were reversed in the matches against ABR. All three configurations of MACE struggled to effectively reuse their case base when playing against ABR. This is not surprising when one considers that ABR is more of an unpredictable opponent, one that introduces more novel board states than AB, which in turn requires more effort to learn effective counter strategies.

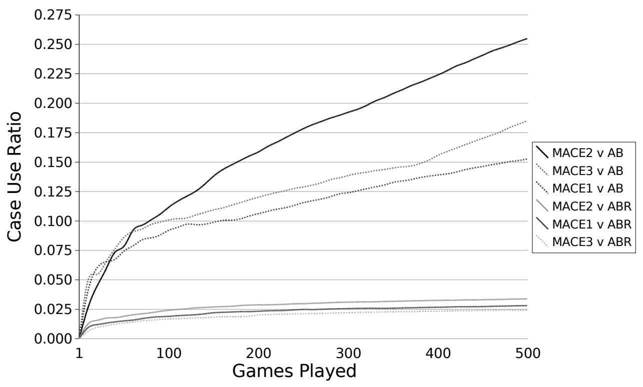

Figure 3: Averaged case-use ratios of MACE1, MACE2, and MACE3 against Alpha-Beta Search in a series of 10 chess matches with 500 games per match.

In summary, the results suggest several things. First, the primary result is that experience through the use of ACE backed by alpha-beta search for novel situations can lead to an improvement in performance in the domain of chess over the use of alpha-beta search alone. Second, it appears that a relaxation in the strictness of case matching allows MACE more freedom to explore and ultimately improves its effectiveness given sufficient interaction with its environment, by finding and exploiting actions which would not otherwise be suggested by an imperfect search mechanism. It is possible that a more intelligent matching technique could lead to further improvements. Third, the results show that ACE can routinely overcome a static opponent, and that adding a degree of unpredictability to the opponent presents ACE with a greater challenge in the exploration for winning strategies. Fourth, it seems that learning from an opponent may not be entirely necessary when the opponent is inferior.

Related Work

Several papers have described the application of CBR to chess including Flinter and Keane (Flinter & Keane 1995) and Sinclair (Sinclair 1998) who each use CBR for chess play by automatically generating case libraries from sets of pre-existing grandmaster games. Cases in ACE differ from these approaches in that cases are not derived from a set of grandmaster games, but instead originate from actual interaction with the environment.

The automatic generation of cases from predefined expert knowledge has received some attention. For example, SHOP/CCBR (Mukkamalla & Muñoz-Avila 2002) automatically acquires cases from manually entered project plans. Shih (Shih 2001) integrates CBR and sequential dependency to learn bridge play from existing games. In contrast, ACE does not compile cases from manually entered or existing data, but instead acquires knowledge automatically through the experiences of the agents who learn completely from scratch.

In the context of CBR, learning, and games, von Hessling and Goel (von Hessling & Goel 2005) describe a technique in which a game playing agent uses Q-learning (Watkins 1989), a form of reinforcement learning, to learn the appropriate action for a given state, and then abstracts and encapsulates action selection within a reusable case for application to other similar states. Our approach differs at this point in that our cases contain concrete observations of the environment, not abstractions.

Gabel and Riedmiller (Gabel & Riedmiller 2005) describe how cases can be used for approximating the state value function in reinforcement learning (specifically temporal difference methods) in complex domains or continuous domains with a state space of infinite size. Demonstrated in the domain of robot soccer, cases include compiled attributes (e.g., the ball's and player's velocities). Our approach is distinguished in that our cases include a snapshot of the environment with no compiled features.

In an approach similar to that described in this paper, DeJong and Schultz (DeJong & Shultz 1988) describe a technique for designing and implementing architectures for extending the capabilities of non-learning problem solvers through integration with a knowledge base. Demonstrated by the system GINA, actions in novel situations (those not stored in the system's knowledge base) are suggested by an underlying problem solver and afterward stored in the system's knowledge base. Unlike the implementation discussed in this paper, GINA does not rely upon exploration in the form of randomly selected top-rated moves from the underlying problem solver.

Goodman (Goodman 1994) describes the use of off-line built decision-tree induction projectors to predict the outcome of various actions during game play in Bilestoad. ACE differs in that agents learn in real time and projection is not coded as a separate step but is instead encapsulated within individual case ratings.

Conclusion

To successfully reason in a domain, a system must have a well defined set of rules describing how to interact with its environment, or a system must have a knowledge base with a sufficient amount of knowledge to facilitate successful reasoning. Unfortunately, many domains are complex enough that the rules of interaction for that domain are not well understood, or it is unknown what knowledge is required to successfully reason within the domain. In such domains a reasoning system would ideally be able to automatically acquire its own knowledge, which it can exploit at a later time. CBR is an excellent methodology for exploiting domain knowledge. CBR can also be used to successfully automate the process of knowledge discovery. One general method for automatically acquiring knowledge within a CBR framework is automatic case elicitation (ACE). ACE relies on real-time trial and error interaction with its environment to acquire a knowledge base diverse enough to succeed in a given domain. In complex domains, the process of automatically discovering knowledge can be very time consuming. To facilitate this process, ACE was combined with an alpha-beta search algorithm to help narrow down the state-space through which a system would need to search. Experimental results indicate that the process of exploration is crucial for an ACE system's performance, and that exploration can lead to superior performance by discovering solutions to problems which would not otherwise be suggested or found by a static or imperfect search mechanism.

References

Aamodt, A., and Plaza, E. 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications* 7(1):39–59.

Aha, D. W., and Molineaux, M. 2004. Integrating learning in interactive gaming simulators. In Fu, D., and Orkin, J., eds., *Challenges in Game Artificial Intelligence: Papers from the AAAI Workshop (Technical Report WS-04-04)*, 49–53. AAAI Press.

Althoff, K.-D., and Weiss, S. 1991. Case-based knowledge acquisition, learning and problem solving for diagnostic real world tasks. In *Proceedings of the European Knowledge Acquisition Workshop*, 48–67.

DeJong, K. A., and Shultz, A. C. 1988. Using experience-based learning in game-playing. In *Proceedings of the Fifth International Conference on Machine Learning*, 284–290. San Mateo, California: Morgan Kaufmann.

Flinter, S., and Keane, M. T. 1995. On the automatic generation of case libraries by chunking chess games. In *Case-Based Reasoning Research and Development, LNAI 1010*, 421–430. Springer Verlag.

Gabel, T., and Riedmiller, M. A. 2005. Cbr for state value function approximation in reinforcement learning. In Muñoz-Avila, H., and Ricci, F., eds., *Case-Based Reasoning Research and Development, LNAI 3620*, 206–221. Springer.

Goodman, M. 1994. Results on controlling action with projective visualization. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, 1245–1250. Menlo Park, Calif.: AAAI Press.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. P. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–285.

Knuth, D. E., and Moore, R. W. 1975. An analysis of alpha-beta pruning. *Artificial Intelligence* 6(4):293–326.

Kolodner, J. 1993. *Case-based reasoning*. San Mateo, Calif.: Morgan Kaufmann.

Kommuri, S. N.; Powell, J. H.; and Hastings, J. D. 2005. On the effectiveness of automatic case elicitation in a more complex domain. In Aha, D. W., and Wilson, D., eds., *Proceedings of the Sixth International Conference on Case-Based Reasoning (ICCBR-05) Workshop on Computer Gaming and Simulation Environments*, 185–192. Chicago, Illinois: Springer.

Mukkamalla, S., and Muñoz-Avila, H. 2002. Case acquisition in a project planning environment. In *Advances in Case-Based Reasoning, LNAI 2416*, 264–277. Springer-Verlag.

Patterson, D. W.; Anand, S. S.; Dubitzky, W.; and Hughes, J. G. 1999. Towards automated case knowledge discovery in the m^2 case-based reasoning system. *Knowledge and Information Systems* 1(1):61–82.

Powell, J. H.; Hauff, B. M.; and Hastings, J. D. 2005. Evaluating the effectiveness of exploration and accumulated experience in automatic case elicitation. In Muñoz-Avila, H., and Ricci, F., eds., *Case-Based Reasoning Research and Development, LNAI 3620*, 397–407. Springer.

Shih, J. 2001. Sequential instance-based learning for planning in the context of an imperfect information game. In *Case-Based Reasoning Research and Development, LNAI 2080*, 483–501. Springer-Verlag.

Simon, H. A., and Schaeffer, J. 1992. The game of chess. In Aumann, R., and Hart, S., eds., *Handbook of Game Theory*, volume 1. Elsevier Science Publishers. 1–16.

Sinclair, D. 1998. Using example-based reasoning for selective move generation in two player adversarial games. In *Advances in Case-Based Reasoning, LNAI 1488*, 126–135. Springer-Verlag.

Slagle, J. R., and Dixon, J. K. 1969. Experiments with some programs that search game trees. *Journal of the ACM* 16(2):189–207.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

von Hessling, A., and Goel, A. K. 2005. Abstracting reusable cases from reinforcement learning. In Aha, D. W., and Wilson, D., eds., *Proceedings of the Sixth International Conference on Case-Based Reasoning (ICCBR-05) Workshop on Computer Gaming and Simulation Environments*, 227–236. Chicago, Illinois: Springer.

Watkins, C. J. 1989. *Learning from delayed rewards*. Ph.D. Dissertation, Cambridge university.