On the Effectiveness of Automatic Case Elicitation in a More Complex Domain

Siva N. Kommuri, Jay H. Powell and John D. Hastings

University of Nebraska at Kearney
Dept. of Computer Science & Information Systems
Kearney NE 68849, U.S.A.
sivakom@gmail.com, hueljh@hotmail.com, hastingsjd@unk.edu

Abstract. Automatic case elicitation (ACE) is a learning technique in which a case-based reasoning system acquires knowledge automatically from scratch through repeated real-time trial and error interaction with its environment without dependence on pre-coded domain knowledge. ACE represents an alternative to manually constructed case bases and domain specific techniques, and is generally applicable to any domain for which knowledge can be obtained from a series of observations of an environment (e.g., checkers or massively multiplayer games). A priority is placed on maintaining the flexibility necessary to learn new domains with only negligible manual configuration. We found during testing that the current approach to ACE with a reliance on experience and exploration, while quite capable in the domain of checkers, did not perform adequately in the exponentially more complex domain of chess. Our results suggest that experience alone, without the ability to adapt for case differences between new and prior cases, is insufficient in more complex domains.

1   Introduction

Automatic case elicitation (ACE) [1, 2] is a learning technique in which a case-based reasoning (CBR) system acquires knowledge automatically through repeated real-time trial and error interaction with its environment. ACE represents an alternative to manually constructed case bases and domain specific techniques. Knowledge is acquired from scratch through an ACE system's interaction with its environment without dependence 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.

ACE is generally applicable to any domain for which knowledge can be obtained from a series of observations of an environment. We expect this approach to be just as easily applied to checkers as to massively multiplayer games. In order to remain relatively domain independent, a priority is placed on maintaining the flexibility and generality necessary to learn new domains with only a minimal amount (e.g., a few minutes) of manual configuration, and not by training ACE on a preselected set of "good" information or by guiding it through the learning process. Peak performance in each individual test domain is the eventual but not immediate goal. Intermediate performance is used to guide our research and suggest areas for improvement in ACE's

learning ability, but not to point out areas of domain specific knowledge which should be explicitly encoded. As such, our philosophy contrasts with research which is focused on precisely "solving" a test domain through domain specific techniques such as search.

The power and flexibility of automatic case elicitation is demonstrated by Powell et al. [2] in which the results suggest that in the domain of checkers, experience 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. This paper extends our previous work by applying ACE to chess in order to determine its effectiveness in a more complex domain and to suggest areas for improvement. Our results suggest that experience alone, without the ability to adapt for case differences between new and previous cases, is insufficient in the domain of chess.

We briefly detail automatic case elicitation in Section 2, and follow in Section 3 with test results that demonstrate the shortcomings of ACE in the domain of chess. We close in Section 4 with the introduction of a proposed planning approach to ACE.

2   Predictive Approach to Automatic Case Elicitation

To date, our work on automatic case elicitation has viewed the task of real-time interaction with an environment (made up of a sequence of discrete observations $O_1,...,O_n$) as a predictive task in which cases identical to the current observed situation (O_i) are used to predict the action most likely to lead to the satisfaction of a final goal (at a final observation O_n). The effectiveness of each acquired case is evaluated using a probablistic reinforcement learning approach which provides a means for an ACE system to explore its environment as well as learn and improve from its experiences. Cases contain only the directly observed situation (i.e., the different types of objects and their locations in the environment), a suggested action, and a continually refined global rating of the action for all of the uses of the action in the situation.

Cases in ACE are abstract only in the sense that each contains a global rating of all uses of an action in a situation, rather than the outcome of a specific use of the action for a specific instance of the situation. These cases differ significantly from the highly compiled cases described by Sinclair [3] which were not learned from scratch, but were instead derived from a set of grandmaster games. In contrast, the only cases learned by ACE are those originating from actual interaction with the environment, not from a set of select interactions manually chosen with an understanding of the domain. For a more detailed description of the predictive ACE algorithm and a discussion of related work, please see Powell et al. [2].

3   Preliminary Application of Automatic Case Elicitation to Chess

The primary purpose of the research described in this paper is to apply automatic case elicitation to the game of chess in order to determine its effectiveness in a domain more complex than checkers and to suggest directions for further research. ACE is evaluated through CHEBR [2], a system in which CBR agents utilize automatic case elicitation to learn and test their expertise in a problem domain using the previously mentioned predictive ACE approach. CHEBR was formerly tested in the domain of checkers. Due

Fig. 1. Screenshot of a Chess Match Between Two CHEBR-chess Players

to the flexibility of ACE, CHEBR was easily applied to the domain of chess (we will call this application CHEBR-chess). The performance of CHEBR-chess was tested in repeated sessions which pitted CHEBR-chess systems A and B against one another for 335 games (approximately six hours of game time on an AMD Athlon XP1800+ processor). Both systems began play with empty knowledge bases. The systems played each other through a modified version of XBoard (a graphical user interface for chess) which could acknowledge the validity of machine moves, and interfaced with XBoard using a modified version of code distributed with TIELT [4]. Figure 1 shows an XBoard screenshot of two CHEBR-chess players in competition after black mated white. In keeping with the ACE philosophy of not pre-coding domain knowledge, the CHEBR-chess players did not have knowledge about what constitutes a valid action, but instead depended on XBoard to validate their moves.[1]

Figure 2 displays the number of moves made per game by each of the CHEBR-chess systems during the 335 game competition. The results are shown using a spline interpolation of the data. Similar results were observed in repeated tests with a slight variability caused by the use of random move generation. Notice that the number of moves was often quite high because the players routinely played games of attrition in which they moved until one player had lost all pieces except the king. Notice also

---

[1] Depending on XBoard to validate moves is not an ideal approach as chess rules are not easily configurable or even fully implemented in XBoard. In the future, we expect that our approach will depend on game rules implemented in TIELT.
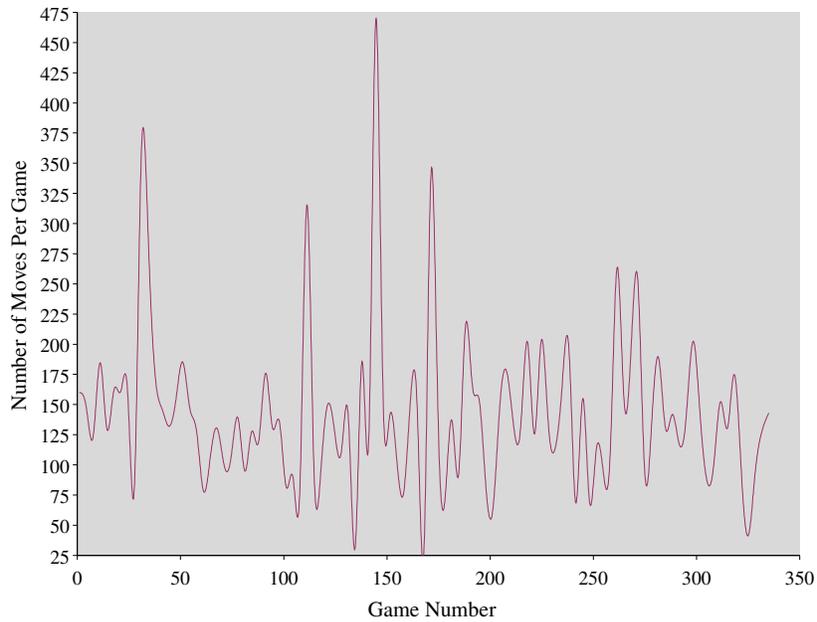
Fig. 2. Number of Moves Per Game for CHEBR-chess A vs. CHEBR-chess B

that the number of moves did not decrease over time, indicating that the approach had difficulty learning within the alloted number of games. It appears that the spikes in the number of moves per game decreased after approximately 175 games. However, given the chaotic nature of the curve, one would suspect that a considerable number of additional training would be necessary for the system to perform adequately, if at all.

Figure 3 illustrates the percentage of games ending in checkmate over the same 335 games. The frequency of checkmates peaked at 22% after 37 games, then dropped to 14% before starting a slow climb. Note that feedback to the players indicating the success of their interaction did not differ for games ending in resignation by one player versus those games ending in checkmate, and as such, players were in no way motivated to find shorter solutions.

As a potential solution to the many long games, we placed move limits on the games to "force" the players to play for shorter games (hopefully checkmate). We attempted a variety of limits including 50, 100 and 150 moves. Once the move limit was reached within a game, the game was ended and the players were informed of the failure. We tried two alternate approaches to having players handle such failures by either:

1. simply disregarding the game and not modifying the ratings of the applied cases, or
2. marking the game as a failure and lowering the ratings of each applied case.

Neither of these solutions proved satisfactory. Clearly, ignoring long games by not committing them to memory was ineffective in shortening games as players had no in-
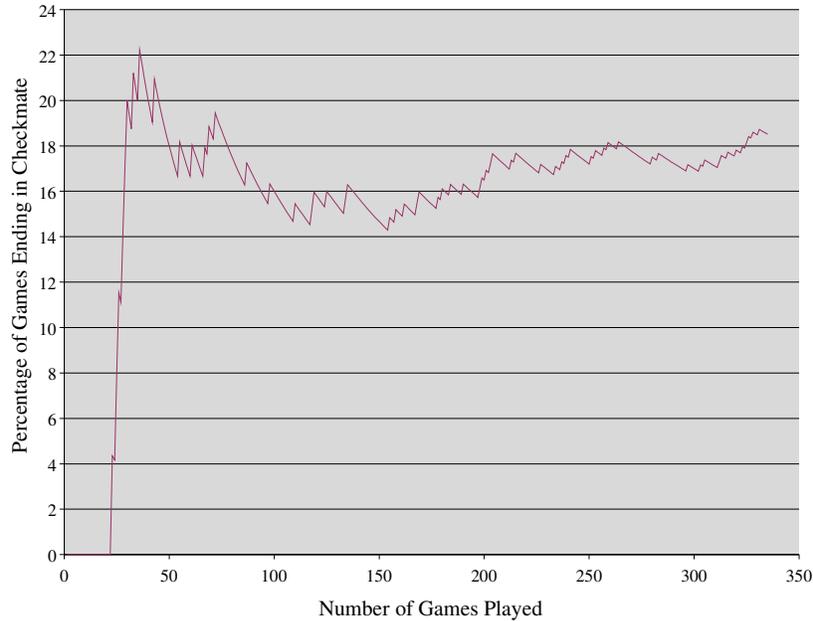
Fig. 3. Percentage of Games Ending in Checkmate for CHEBR-chess A vs. CHEBR-chess B

centive to play for a short match or any remembered experience about what constituted a long game. Marking long games as a failure was also ineffective because it dropped the ratings of frequently used early moves that had little relationship with the length of the game. The ratings' decrease for early moves was significant for the 50 game threshold and caused the players to effectively quit using those moves. It is unclear how tightly the ratings of early actions should be related to failures late in the game; it is possible that either early actions play a big part in leading the player down a path for success or failure (like an early fork in the road), or that the effect of actions on the final outcome is inversely proportional to the distance from the final state (and that minor "mistakes" made early in the action sequence can be corrected later).

Although the case representation (i.e., observation, action, and global success rating) used in ACE proved relatively effective in the world of checkers, initial applications to chess illustrated some weaknesses. The biggest problem appears to be that the abstract nature (with the global rating) of the cases does not support the type of introspection which could support the creation of case adaptation rules. Without adaptation (and the ability to retrieve similar but different cases), we chose to retrieve only exact matches, which required our system to explore and encounter a larger variety of situations in order to succeed. This approach applied to chess performed inadequately as the complexity of chess (10^{43} [5] board positions for chess compared to 10^{20} [6] for checkers) makes it impossible to explore a sufficient portion of the state space (especially board states later in the game which are encountered infrequently). Even if similar

matching cases were allowed (without adaptation), cases are stored without regard to the context of their use, and would likely be applied incorrectly.

4 Future Work

Given the weaknesses of the predictive approach to ACE when applied chess, we plan to investigate a case-based planning (CBP) approach. CBP was one of the earliest CBR research areas [7, 8], and has a rich history [9] with applications to a variety of problem domains. CBP is a process whereby a system stores problem solutions (as plans) in memory, and constructs new plans in order to achieve one or more goals for a new situation by retrieving and adapting similar, previously remembered plans. The focus in CBP is retrieval and adaptation, rather than the refinement of a plan from scratch. Viewed simply, plans (i.e., cases) are a sequence of actions which, when applied to a situation, affect some outcome (the goal). New plans result in either success or failure. Failed plans can be contributed to the case base "as is", or can be repaired given the additional (external or future) information necessary to determine the cause of the failure.

In the CBP approach to ACE, knowledge in the form of cases will again be acquired from scratch through real-time interaction with the system's environment. However, cases will not abstractly represent the history of an action in the context of a specific observation, but will rather represent a complete problem solving episode as:

1. a chronological sequence of discrete observations $(O_1,...,O_n)$ of the dynamic environment from the initial observation (O_1) to the final observation (O_n) with each observation connected by any valid action taken by the system, and
2. the success of the plan.

As in the past, during interaction with its dynamic environment, the ACE system will constantly monitor the environment for changes. When changes occur, ACE will attempt to interact with the environment by retrieving a set of subcases sufficiently similar to the current environment.

We plan to add adaptation rules [10, 11] to ACE in an attempt to account for any differences between the current environment and those in the matching subcases. These adaptation rules should allow the ACE approach to more adequately handle novel situations.

ACE will iterate through each of the similar subcases, applying the adapted action associated with a subcase at random with the probability of selection related to a rating of the potential effectiveness of the subcase (as determined by the history of the subcase, the distance of the subcase from the end of the interaction, and the past success of the adaptation rules). As in the past, ACE will ensure that the probability of selecting a previous action will be less than 1.0. This mechanism allows for the possibility of not selecting a subcase in order to encourage occasional exploration of the domain. When a subcase action is not selected, a new action will be constructed.

At the end of each interaction (e.g., a game in the domain of chess), the ACE planning approach will contribute the new case to memory and analyze its cases in order to automatically tune and construct adaptation rules. We have begun implementating this approach and hope to apply it to chess in the near future.

5 Conclusion

Because automatic case elicitation (ACE) can flexibly acquire knowledge automatically without precoded domain knowledge, it was easily applied to chess after having been initially applied to checkers. We found that the current approach to ACE with its reliance on experience and exploration, while quite capable in the domain of checkers, did not perform adequately in the exponentially more complex world of chess. Our results suggest that experience alone, without the ability to adapt for case differences between new and prior cases, is insufficient in more complex domains. We also feel that our abstract case structure might limit the flexibility needed to succeed in highly complex domains. As an alternative, we propose a planning approach to automatic case elicitation, an approach that will more easily support the creation of adaptation rules and produce better performance in complex environments.

6 Acknowledgements

References

1. Powell, J.H., Hauff, B.M., Hastings, J.D.: Utilizing case-based reasoning and automatic case elicitation to develop a self-taught knowledgeable agent. In Fu, D., Orkin, J., eds.: Challenges in Game Artificial Intelligence: Papers from the AAAI Workshop (Technical Report WS-04-04), AAAI Press (2004) 77–81
2. Powell, J.H., Hauff, B.M., Hastings, J.D.: Evaluating the effectiveness of exploration and accumulated experience in automatic case elicitation. In: Proceedings of the Sixth International Conference on Case-Based Reasoning (ICCBR-05), LNAI 2080, Chicago, Illinois, Springer (2005)
3. Sinclair, D.: Using example-based reasoning for selective move generation in two player adversarial games. In: Proceedings of the Fourth European Workshop on Case-Based Reasoning (EWCBR-98), LNAI 1488, Springer-Verlag (1998) 126–135
4. Aha, D.W., Molineaux, M.: Integrating learning in interactive gaming simulators. In Fu, D., Orkin, J., eds.: Challenges in Game Artificial Intelligence: Papers from the AAAI Workshop (Technical Report WS-04-04), AAAI Press (2004) 49–53
5. Shannon, C.E.: Programming a computer for playing chess. Philosophical Magazine **41** (1950) 256–275
6. Lake, R., Schaeffer, J., Lu, P.: Solving large retrograde analysis problems using a network of workstations. In: Advances in Computer Chess VII, Maastricht, Netherlands (1994) 135–162
7. Hammond, K.J.: Case-based planning: Viewing planning as a memory task. Academic Press, Boston (1989)
8. Hammond, K.J.: Explaining and repairing plans that fail. Artificial Intelligence **45** (1990) 173–228
9. Spalazzi, L.: A survey on case-based planning. Artificial Intelligence Review **16** (2001) 3–36
10. Hanney, K.: Learning adaptation rules from cases. Master's thesis, Computer Science Department, Trinity College Dublin (1996)

11. Hanney, K., Keane, M.T.: The adaptation knowledge bottleneck: How to ease it by learning from cases. In Leake, D., Plaza, E., eds.: Proceedings of the Second International Conference on Case Based Reasoning (ICCBR-95), LNAI 1266, Springer Verlag (1997) 359–370