Toward Automated Knowledge Discovery in Case-Based Reasoning

Sherri Weitl-Harms
Creighton University
Omaha, NE 68178 U.S.A.
SherriWeitlHarms@creighton.edu

John D. Hastings
Dakota State University
Madison, SD 57042, U.S.A.
John.Hastings@dsu.edu

Jay H. Powell
Unaffiliated
jay.h.powell@gmail.com

Abstract

Automated Case Elicitation (ACE) enables case-based reasoning (CBR) systems to automatically acquire knowledge through real-time exploration and interaction with environments. CBR is an explainable AI methodology, where decisions are based on previous encounters. ACE combined with CBR continues learning as it is being deployed, and produces specific cases that can be reviewed by humans, unlike pre-trained large language models (LLMs) that learn by training offline on prior data. ACE and CBR may be useful methods to gather training data for use with generative AI, or to help them to adapt on the fly. This research explores ACE's potential by applying it to chess and conducting extensive experiments against Stockfish, the world's highest rated chess engine. An ACE agent was developed that combines random exploration with shallow alpha-beta search for novel game states. Results over 1000+ games showed the ACE player defeated Stockfish in nearly 10% of games—a notable achievement given Stockfish's extreme strength. Notably, the ACE agent required only 0.1 seconds per game compared an average of 8 minutes for Stockfish, while still gradually improving its win rate through accrued experience. Detailed analyses revealed how the relaxation of ACE's case matching criteria along with selective retention of useful cases enabled accumulation of strategic chess knowledge. The research provides valuable insights into ACE's proficiency for knowledge discovery in complex, adversarial domains. It lays groundwork for integrating ACE, an unsupervised CBR learner, with modern deep learning techniques like neural networks and large language models to combine the strengths of symbolic and subsymbolic AI. By demonstrating ACE's ability to extract strategic knowledge against world-class opponents, this work highlights its potential for impact across gaming, autonomous systems, and other complex problem-solving domains.

1 Introduction

LLMs have achieved impressive performance with minimal knowledge acquisition effort in many problem domains but typically depend on large pre-trained data sets to achieve it (Leake 2023). Conclusions are based on generalizations, require offline retraining on new information, and function as "black boxes" as their reasoning processes are opaque (Leake 2023). They do not capture many important types of information behaviors, such as retention of long-term memories and specific facts, and propose statistical assertions (Hammond and Leake 2023), which is a significant problem for the reliability, interpretability, transparency, and explainability of information provided (Leake 2023; Talaei Khoei, Ould Slimane, and Kaabouch 2023). Model performance must be balanced with comprehensibility to gain meaningful insights into the model's reasoning, enabling informed decisions and accountability (Talaei Khoei, Ould Slimane, and Kaabouch 2023). Additionally, attention to the model metrics is important, as the progressive improvements in deep learning models have significantly increased their number of parameters, latency, and resources required to train. (Menghani 2023; Sojka 2022).

CBR is a knowledge-based reasoning and learning methodology (Watson 1996) inspired by human cognition that adapts prior cases of prior experiences, to solve new problems. CBR 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 (Powell and Hastings 2006). The CBR process is naturally interpretable and explainable, can learn from few examples, and provides inertia-free lazy learning (Leake 2023). CBR is a general high-level process that defines a set of tasks, but for which the needed functionality can be implemented using various technologies, both neurally inspired and symbolic (Leake and Crandall 2020).

The main four steps in the CBR methodology are retrieval, reuse, revision and retention (Lopez de Mantaras, McSherry, and et al. 2005). The reasoning part of CBR follows from case adaptation and provides the ability to transform solutions to new contexts (Leake and Crandall 2020). CBR retrieves similar cases to be reused to solve a new problem. It uses case adaptation to revise existing cases to align with the new problem, and then it retains the new cases for future retrieval, reuse, and revision. Case-based reasoners are lazy learners, retaining raw cases (or cases with limited processing) to re-use them (Leake and Crandall 2020).

CBR systems can handle many problems that LLM do not do well, such as provide facts, capture and provide relevant information, make inferences and remember (Hammond and Leake 2023; Lopez de Mantaras, McSherry, and et al. 2005). Using CBR may provide opportunities to im-

prove LLMs to leverage the strengths of both (Hammond and Leake 2023). Recent research in this area focuses on the CBR methodology as the means to guide both LLMs and the interactions with them (Hammond and Leake 2023; Leake and Crandall 2020).

When CBR is used for guiding the design of deep learning systems, it can be implemented as part of the model itself (Leake and Crandall 2020). Adding a CBR process that reasons and learns from single cases could help the LLM learn from limited data (Leake and Crandall 2020). When adding CBR to the LLM, there are three possibilities for implementing adaptation (Leake and Crandall 2020): *add an explicit case retrieval phase for "pre-packaged" cases; generate cases by a reconstructive process; or explicitly retrieve cases with an added adaptation phase after solution generation, when no existing case matches*.

Automatic case elicitation (ACE) (Powell, Hauff, and Hastings 2005; Kommuri, Powell, and Hastings 2005; Powell, Hauff, and Hastings 2004) is an adaptation technique that can be used within a case-based reasoning (CBR) system. It uses reinforcement learning (Kaelbling, Littman, and Moore 1996; Sutton and Barto 1998; Shakya, Pillai, and Chakrabarty 2023), acquires knowledge automatically and without supervision through repeated real-time exploration and interaction with its environment. When ACE encounters a situation which is sufficiently distinct from previous experience, it applies a sequence of new actions at random until a change in the environment is observed. ACE does not utilize separate training and testing phases, but instead continually improves its performance through repeated exposure to its environment. Due to ACE's core philosophy of not requiring pre-coded domain knowledge (e.g. rules or cases) 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 (Powell and Hastings 2006). While it works without guidance from pre-coded domain knowledge, it uses domain knowledge to evaluate the success of a solution (Floyd and Esfandiari 2009). Cases created during a winning game gain positive reinforcement, whereas losing games have their cases reinforcement values decreased (Powell, Hauff, and Hastings 2004).

Automatically discovering knowledge for a chess playing agent is a challenging, non-trivial problem (Silver et al. 2018). For example, chess-playing LLMs can suggest board moves based on their existing training, and can be fine tuned to do even better by training offline on prior games. However, this training isn't done while games are being played as the generative models aren't adaptable and have only a small context "window" that isn't saved. ACE and CBR may be useful methods to gather training data for use with an LLM, or to help it to adapt on the fly. To explore this concept, this research performs an empirical study of an ACE-based chess agent connected through XBoard (GNU 2024) to play against Stockfish (Yang 2024), currently the strongest chess engine available to the public (with Elo ratings over 3400).

Section 2 provides a discussion of related work. Section 3 gives a brief description of the ACE algorithm. Section 4 sets forth an experimental evaluation in which various versions of ACE were evaluated against Stockfish. Section 5 details the results of our experimentation, which shows that the ACE player combined with alpha-beta search performs reasonably well against the best chess game, Stockfish. Section 6 provides future work and a conclusion.

2 Related Work

2.1 Chess Engines

According to (Maharaj, Polson, and Turk 2022), Stockfish and LCZero represent two competing paradigms in the race to build the best chess engine. The Stockfish chess engine (Yang 2024) is an open source program written in C++. Its search component is based on alpha-beta pruning (Slagle and Dixon 1969; Knuth and Moore 1975) with iterative deepening. For its evaluation function, Stockfish will go as deep as the framework allows it time to perform, and can look at over 70 million positions per second (Silver et al. 2018). When the time expires, it returns the best move it has found so far. Everything that it finds is stored in a hashtable, cached for potential future use. The next iteration of the evaluation function, it will start with the moves stored in the hashtable.

The LCZero project (LCZero 2024) was created via crowd computing to reproduce the Google DeepMind, non-public work of AlphaZero (Silver et al. 2018; Tomasev et al. 2020). These are general purpose reinforcement learning (RL) algorithms that can learn near-optimal strategies for any rule set from scratch, without any human supervision, by continually learning from its own experience, playing against itself many millions of times. LCZero has far surpassed the original strength of AlphaZero due to its additional training and improvements (Maharaj, Polson, and Turk 2022), and has won in competitions with Stockfish (chess.com 2024).

Several of the RL chess approaches were analyzed by (Hu 2023), which notes that the depths of RL are vast. Because these approaches are not naturally explainable, work has been done to attempt to gain insight into the chess concepts learned (McGrath et al. 2022). The analysis is possible because in chess, deep, expert human knowledge is explicitly available. Using these approaches in contexts without this would be much harder to analyze.

The work of (Maharaj, Polson, and Turk 2022) implies that Stockfish is currently the better tool for studying deep puzzles, and finds that the Stockfish engine has better search, but LCZero has better evaluation. Stockfish currently has Elo ratings over 3400.

2.2 Related CBR Applications

Several papers have described the application of CBR to chess including Flinter and Keane (1995) and Sinclair (1998) who each use CBR for chess play by automatically generating case libraries from sets of pre-existing grandmaster games. Cases in ACE as reported in (Powell, Hauff, and Hastings 2005; Powell and Hastings 2006) differ significantly from these approaches in that cases are not derived from a set of grandmaster games, but instead originate from

actual interaction with the environment and include a snapshot of the environment with no compiled features.

Results in Powell et al. (2005) and Neto and Julia (2018) suggest 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. Initial testing in the domain of chess (Kommuri, Powell, and Hastings 2005) revealed that experience alone, without the ability to adapt for differences between new and previous cases, is insufficient in complex domains. To decrease the complexity of the state space, Powell and Hastings (2006) introduced an alpha-beta search (ABR) algorithm into the ACE framework. Their results suggested 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 and that relaxations to the case matching component in order to encourage exploration can lead to additional improvements (Powell and Hastings 2006). Moral et al. 2020 investigated adaptations of ACE on the Truco card game, and found that the alternative case learning strategies were able to play better when compared to agents based on a case base initially collected with human players.

A distant, but related approach is seen in (Samuel 1959; 1967), which describes the use of rote-learning and lookahead in the game of checkers. The approach requires that the game must have at least one intermediate goal, as opposed to ACE which has no such requirement, and instead utilizes only the final success rating of the interaction with its environment. In addition, ACE uses a process of exploration to augment and improve its knowledge.

Leake (2023) discusses the opportunities for combining CBR with neural networks to leverage both paradigms. Hoffman and Bergman (2022) used CBR to improve the optimization of hyperparameters in machine learning models. The case base is then used to retrieve hyperparameter vectors given a query vector and to make decisions whether to proceed with this query or abort and sample another vector.

CBR researchers have explored many machine learning techniques for acquiring case adaptation knowledge (Ye, Leake, and Crandall 2022). A CBR approach that like ACE, does not use human-generated cases, is CBGen (Borck and Boddy 2017), a genetic algorithm-based approach. ROAD (Leake and Ye 2019), uses heuristics to guide multi-step adaptations, with each adaptation chosen in the context of adaptations applied previously. An approach to handling situations when no case matches are available for expansion is the Expansion-Contraction Compression (ECC) approach (Leake and Schack 2018), which precedes compression with adaptation-based exploration of previously unseen parts of the problem space to create "ghost cases" and exploits them to broaden the range of cases available for competence-based deletion. An approach to handling the case adaptation process in game playing was described in (Miranda, Sanchez-Ruiz, and Peinado 2019), where control was given to a human player when the CBR game-playing bot reached game states that were not well represented by cases in its case base, then the CBR bot regained control when the game states were known again. Their results, applied to Pac-Man, showed that by using interactive learning the amount of human intervention decreases rapidly, the case base needed to achieve reasonable imitation is considerable smaller than that used in a non-interactive approach and the resulting agent outperforms other agents using non-interactive CBR. Finally, Bach et al. (2014) presented a clustering-based method for capturing cases in time series sensor data. In this approach, the domain expert validates the cases suggested by the system.

3 Automatic Case Elicitation

This research aims to determine the value of experience, in the form of cases exploration using CBR and ACE within the domain of chess, when playing against a top opponent. The reader is referred to Powell et al. (2005) for an indepth look at ACE. Briefly, ACE can be described as a learning technique in which a CBR system 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). 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. 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. It will fail horribly early on while it learns, but gains competence with experience.

The primary reasoning module, *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 (A) suggested by the *Decision* function and the case library until a change in the environment is observed.

The function *Decision* recurses through the set of cases which match the current situation (in decreasing order by rating) and returns the action recommended by the first case whose rating is above a randomly chosen value. If the set of matching cases is exhausted, *Decision* chooses an action (one which might eventually be found to be entirely ineffective or illegal) at random.

Upon the completion of the interaction, *Evaluate* is called to update the ratings of each applied case. 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. *Ace* closes with a call to *Store* to commit the changes to the case library.

4 Experimental Methodology

This research focused on playing the best implementation of ACE, or MACE (Modified ACE) version 2 from (Powell and Hastings 2006) against the Stockfish (Yang 2024) chess engine. Stockfish version 14.1 was selected due to ease of installation through the Linux package manager. The pack-

age documentation suggests that the latest version (16) does not offer significantly improved capabilities.

MACE begins with an empty case base containing no chess knowledge. To handle novel game states not in its case base, MACE employs the traditional ABR search to a shallow depth of three. These learned experiences build the case base.

When facing a board state, MACE selects a move by iterating through a set of all matching cases ordered by rating (with ratings between 0.4-1.0). For each case, a random number between 0.0-1.0 is generated, which if lower than the case rating, the case is selected, otherwise the next case is considered. If no case is selected, i.e., the system has encountered a novel situation, MACE invokes ABR search to generate a move. By using ABR to recommend potentially good moves in unfamiliar positions, MACE balances focused search with randomized exploration. As the case base grows through experience, MACE can increasingly rely on matched cases and decreasingly on shallow ABR search for move decisions.

The systems played each other through XBoard (a graphical user interface for chess) (GNU 2024) using a modified version of code distributed with TIELT (Aha and Molineaux 2004). Figure 1 shows an XBoard screenshot of the MACE agent competing against Stockfish during a match. MACE is shown in black and completely eliminated all but the Stockfish king. MACE lost the first 56 games before able to accomplish this. MACE consumed only three seconds of overall play time, and Stockfish used 7:38 minutes. However, MACE was unable to efficiently close out the game and played to a draw due to exceeding the number of moves allowed per game. Through experimentation, this was found to be a common issue. It was discovered that the board evaluation was subtracting 10k points for MACE losing a king, but wasn't adding 10k points for taking a king. Although MACE might eventually make a winning move, the number of moves was naturally higher due to this lack of focus. This was addressed, so the underlying alpha-beta search acted more reliably in novel end game situations. The improved version of MACE was then used in the experimentation described below.

The experiments were designed to determine:

1. the value of experience in the domain of chess through a combination of ACE and alpha-beta search,
2. verify that ACE playing from scratch is a valid approach against the de facto top computer chess player,
3. understand the limitations as well as parameters in implementing ACE in this manner, and
4. the effect of learning from an opponent in a complex game environment such as chess.

For experimentation, the following approaches were used to play against Stockfish:

1. **ABMACE:** An alpha-beta MACE with no case library.
2. **ExactMACE:** An alpha-beta MACE, with an added case library, where only cases that were part of a winning game were given a rating of 1 and saved. If a case was part of a losing game, it was dropped. Thus, cases that led to failures were not saved. This algorithm is referred to as 'Exact' as the strings/boards must match exactly to be reused.
3. **ExactMACEDrop:** An exact alpha-beta MACE, which preserves bad cases, so the system would not use the case again. Cases with a rating below 0.1 are typically cases (bad choices) toward the end of the game. It is important for the system to"remember" truly bad cases as shown in (Boulmaiz, Reignier, and Ploix 2023). However, cases with ratings between 0.1 and 0.5 were dropped from the library, to save room and lower the number of cases that need to be evaluated, as the search space is so large.
4. **RelaxedMACE:** Similar to ExactMACE, it had a case library where cases that were part of a winning game were given a rating of 1 and saved, and cases that were part of a losing game were dropped. However, this version also used a relaxed matching criterian of < 3 meaning that boards that are different by fewer than 3 pieces were considered a match.
5. **RelaxedMACEDrop:** A relaxed MACE algorithm with preserved bad cases, so the system would not use the case again, but still dropping cases with relatively low ratings. Several variations were tried, using ratings between a range of values, such as dropping cases with values between 0.1 and 0.5, to save room and lower the number of cases that need to be evaluated. The relaxed matching used in these versions also ranged from three to seven. The six versions of the RelaxedMACEDrop used were:
   - Relaxed match < 7, drop cases $(0.1 <= x < 0.4)$

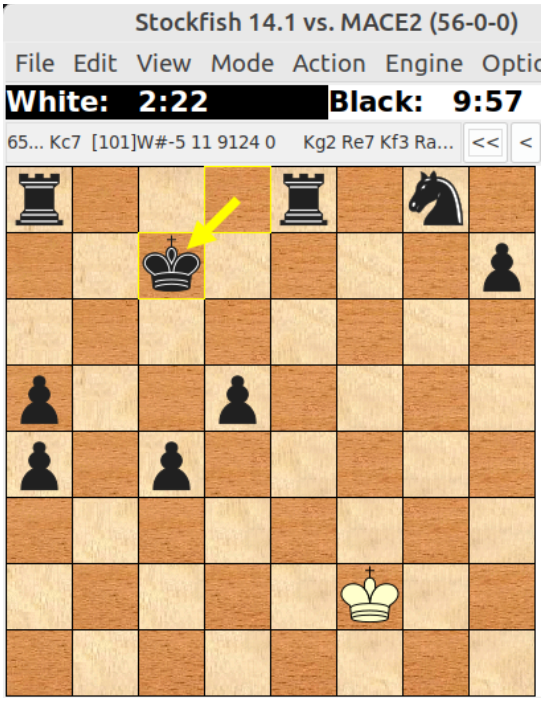

Figure 1: Screenshot of a Chess Match Between MACE2 and Stockfish.

- Relaxed match < 3, drop cases $(0.1 <= x < 0.4)$
- Relaxed match < 5, drop cases $(0.1 <= x < 0.4)$
- Relaxed match < 3, drop cases $(0.1 <= x < 0.5)$
- Relaxed match < 5, drop cases $(0.1 <= x < 0.5)$
- Relaxed match < 3, drop cases $(0.2 <= x < 0.5)$

Besides ABMACE and ExactMACE, the other algorithms rated cases using a modified distance measure from (Powell and Hastings 2006). The philosophy follows that of a self-driving car - in the event of a crash, the most recent actions were the most likely culprits (i.e., incorrect decisions leading to a negative outcome). Ratings for cases start with a 0.5 rating. A final winning move receives a 1.0 rating with a final losing move receiving a 0.0 rating. Intermediate moves are updated proportionally (toward 1.0 for a winning game or toward 0.0 for a losing game) based on their distance from the end of the game. Thus, moves made toward the beginning of the game, which are very distant from the end of the game are updated by only a small amount.

5 Results and Discussion

Figure 2 shows a comparison of the average winning percentages for each experimental approach when playing against Stockfish for 1000 games. The best performing approach was RelaxedMACEDrop with relaxed matching < 3, and drop cases $(0.1 <= x < 0.5)$. This variation approached a 10% win ratio with its success rate still trending upward somewhat after 1000 games. This level of performance was repeatable in that running the experiment multiple times produced similar results. Future exploration is needed to determine how many games are required to achieve the best performance for this variation.

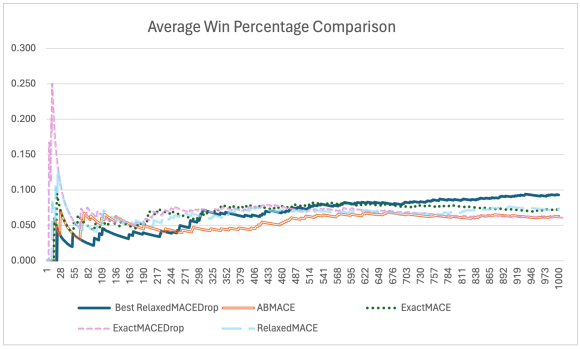

Figure 2: Averaged Win Percentages of various MACE algorithms playing against Stockfish in a series of 1000 chess matches.

Figure 2 perhaps reveals some potential areas for improvement if the early gains could be preserved and built upon for some of the versions. Or perhaps the early jumps in win percentages simply show that early performance can be somewhat erratic. In general, the steep upward slopes show where MACE is doing well, indicating that it is figuring out a successful strategy, followed by a drop in performance, perhaps by not properly leveraging or remembering what it has already learned.

In addition to the top performing version of RelaxedMACEDrop, the variations listed in the prior section were explored to determine the impact of parameter adjustments. As shown in Table 1, these variations produced different success rates suggesting that fine tuning these parameters could lead to further improvements.

	Algorithm	Avg Win Pct
1	ABMACE	0.0630
2	ExactMACE	0.0730
3	ExactMACEDrop, dropped cases $(0.1 <= x < 0.5)$	0.0610
4	RelaxedMACE, match < 3	0.0735
5	RelacedMACEDrop match < 7, drop cases $(0.1 <= x < 0.4)$	0.0620
6	RelacedMACEDrop match < 3, drop cases $(0.1 <= x < 0.4)$	0.0700
7	RelacedMACEDrop match < 5, drop cases $(0.1 <= x < 0.4)$	0.0765
8	**RelacedMACEDrop match** < 3, **drop cases** $(0.1 <= x < 0.5)$	**0.0930**
9	RelacedMACEDrop match < 5, drop cases $(0.1 <= x < 0.5)$	0.0705
10	RelacedMACEDrop match < 3, drop cases $(0.2 <= x < 0.5)$	0.0791

Table 1: Average Win Percentage of the various reasoning approaches evaluated in the experiments.

Key takeaways from these results are:

1. The addition of rudimentary cases (approach 2) beyond ABR search (approach 1) leads to an increase in performance. This suggests that *memory in the form of cases is beneficial*.
2. In comparing approaches 4 and 8 (with case matching fixed at < 3), the addition of a limited case dropping mechanism can lead to a further increase in performance in certain configurations. This suggests that it might be *beneficial to forget prior experience in some situations*. However, in comparing approaches 2 and 3, or approaches 4 and 6, it appears that narrowing the case library in certain configurations can cause performance to drop, so an improper range can have the opposite effect.
3. In comparing approaches 3, 8 and 9 (with the drop range fixed at $(0.1 <= x < 0.5)$), it appears that relaxing the case matching somewhat can provide an increase in performance, as long as it's not increased too much. This suggests that some *creativity or exploration in game play is beneficial*, but relaxing the case matching too much can cause actions which are less applicable to be applied.

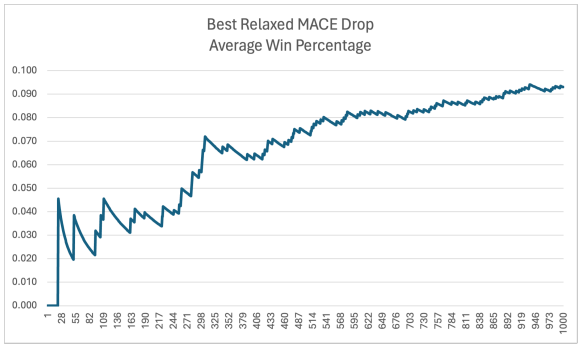

Figure 3: Best MACE algorithm playing against Stockfish in a series of 1000 chess matches.

It was observed that MACE will go through spurts with a much higher success rate, followed by losing a number of games. This behavior can be somewhat seen in Figure 3 and more clearly in Figure 4. For the best configuration, success was sometimes seen very early where it might win 3 games in the first 10, which is impressive. Further research needs to be conducted to determine how MACE can best leverage prior learned strategies and build on momentum, rather than losing it, and having a drop in performance. Adjustments to the previously discussed parameters are likely to help, but perhaps additional mechanisms need to be explored.

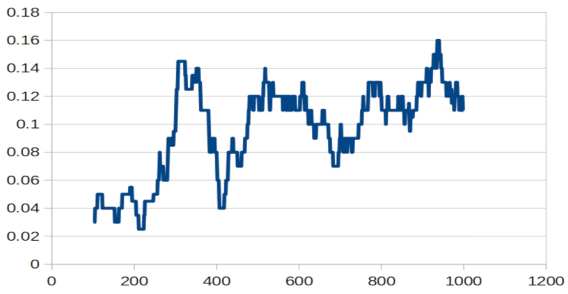

Figure 4: Win percentage of the best MACE algorithm playing against Stockfish over the last 100 chess matches.

Given a longer period of exploration and experience, MACE was able to successfully learn how to retrieve and apply non-exactly matching cases. The fact that MACE demonstrated its top performance in longer matches is a key result and suggests that the matching component can be relaxed 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.

MACE is incredibly fast and has a case library which can be inspected (which neural nets do not have). Stockfish is very slow. MACE uses only about 0.1 seconds per game as compared with Stockfish would often takes 8 minutes for a game, meaning that MACE consumes 99% less time than Stockfish.

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 against a top-rated opponent. Second, it appears that keeping both positive and negative cases assists MACE in its exploration and ultimately improves its effectiveness given sufficient interaction with its environment.

6 Future Work and Conclusion

Several experimental adjustments are planned. This includes saving the "forgotten" dropped cases in a separate file, and then seeing if MACE comes back to them again. Early cases might have an unfair rating while MACE is learning. Another statistic to capture in the future is how many times it actually uses a case from the case library. Given the complexity of the state space, it is likely low, particularly later in a game. Initial exploration found that the first few moves were often reused.

Rating is learned by experience - over time it is learned which moves were successful in the past, and successful moves are reused. Is MACE learning on its own the strength of this method or is it discovering already known expert insights into the quality of a move? A review of the case library and the ratings of moves for comparison with known insights by chess experts, is needed.

Currently, matching in MACE is a simple string comparison. Future exploration is needed to see if MACE could be improved through a more sophisticated matching approach, e.g., training using a neural net or other technique. Also, if the matching is relaxed, should an adaptation component be trained to better leverage prior experience? For example, if a new board matches a prior somewhat different case from the case library because it somehow would have a similar inherent strategy or choice, what action needs to be taken in the new situation? Would it take a couple of actions to get it on the "winning" path of the prior solution? Another weakness of the case library is that the cases are not linked in any way. Future work is needed to determine if those should be tied together in some way. Conceptually, it might be viewed as an N-dimensional web. So, for a specific case, how can it get there, and where might it go?

With the MACE algorithm being able to adapt in real time, it is possible to leverage it to train an LLM, which are normally trained offline. This could be done by ether training from scratch or fine tuning them with a low-rank adapter, which will require an added step for the training of an LLM. Perhaps MACE could work hand in hand in real time to help with explanations of why certain moves are considered good? A weakness of many AIs is that they are not explainable. With CBR and ACE, experts can look at the case library and see how the knowledge is used. Future work is needed to train an ACE system on an LLM, which would require figuring out how to interface with the LLM and also "training" the LLM with a prompt that tells it to focus on playing chess at a very high level and compare the performance of such a system with the prior ACE attempts.

This research introduced the concept of using ACE/CBR to shape the deep learning pipeline. It advocates for using this approach as an added step in the learning/training process as an adaptation phase after solution generation. As a first step into studying this approach, this research applied ACE/CBR in chess, playing against Stockfish, the de facto strongest chess engine available to the public. An ACE agent was developed that combines random exploration with shallow alpha-beta search for novel game states. Results over 1000+ games showed the ACE player defeated Stockfish in nearly 10% of games—a notable achievement given Stockfish's extreme strength, while consuming 99% less time than Stockfish. Results indicate the success of an unsupervised CBR learner can lead to superior performance in chess. Results also indicate that ACE and CBR combined with generative AIs may lead to improvements in solutions.

References

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.

Bach, K.; Gundersen, O. E.; Knappskog, C.; and Ozturk, P. 2014. Automatic case capturing for problematic drilling situations. In Lamontagne, Lucand Plaza, E., ed., *Case-Based Reasoning Research and Development*, 48–62. Cham: Springer International Publishing.

Borck, H., and Boddy, M. 2017. Automated case generation using a genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, 187–188. New York, NY, USA: Association for Computing Machinery.

Boulmaiz, F.; Reignier, P.; and Ploix, S. 2023. Learning from successes and failures: An exploration of a case-based reasoning technique. In Fujita, H.; Wang, Y.; Xiao, Y.; and Moonis, A., eds., *Advances and Trends in Artificial Intelligence. Theory and Applications*, 74–87. Springer Nature Switzerland.

chess.com. 2024. chess.com. `https://www.chess.com/terms/alphazero-chess-engine`, Accessed 2024-01-04.

Flinter, S., and Keane, M. T. 1995. On the automatic generation of case libraries by chunking chess games. In *Proceedings of the First International Conference on Case Based Reasoning (ICCBR-95), LNAI 1010*, 421–430. Springer Verlag.

Floyd, M. W., and Esfandiari, B. 2009. An active approach to automatic case generation. In McGinty, L., and Wilson, D. C., eds., *Case-Based Reasoning Research and Development*, 150–164. Berlin, Heidelberg: Springer Berlin Heidelberg.

GNU. 2024. Xboard. `https://www.gnu.org/software/xboard`, Accessed: 2024-01-04.

Hammond, K., and Leake, D. 2023. Large language models need symbolic ai. In *Proceedings of the 17th International Workshop on Neural-Symbolic Reasoning and Learning, CEUR Workshop (NeSy 2023)*, 204–209.

Hoffmann, M., and Bergmann, R. 2022. Improving automated hyperparameter optimization with case-based reasoning. In Keane, M. T., and Wiratunga, N., eds., *Case-Based Reasoning Research and Development*, 273–288. Springer International Publishing.

Hu, M. 2023. Planning with a model: Alphazero. In *The Art of Reinforcement Learning: Fundamentals, Mathematics, and Implementations with Python*. Berkeley, CA: Apress. 245–280.

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.

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.

LCZero. 2024. Lczero. `https://lczero.org`, Accessed: 2024-01-08.

Leake, D., and Crandall, D. 2020. On bringing case-based reasoning methodology to deep learning. In Watson, I., and Weber, R., eds., *Case-Based Reasoning Research and Development*, 343–348. Springer International Publishing.

Leake, D., and Schack, B. 2018. Exploration vs. exploitation in case-base maintenance: Leveraging competence-based deletion with ghost cases. In Cox, M. T.; Funk, P.; and Begum, S., eds., *Case-Based Reasoning Research and Development*, 202–218. Cham: Springer International Publishing.

Leake, D., and Ye, X. 2019. On combining case adaptation rules. In Bach, K., and Marling, C., eds., *Case-Based Reasoning Research and Development*, 204–218. Springer International Publishing.

Leake, D. 2023. Bridging ai paradigms with cases and networks. In *Comput. Sci. Math. Forum*, 71.

Lopez de Mantaras, R.; McSherry, D.; and et al. 2005. Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review* 20(3):215–240.

Maharaj, S.; Polson, N.; and Turk, A. 2022. Chess ai: Competing paradigms for machine intelligence. *Entropy* 24(4).

McGrath, T.; Kapishnikov, A.; Tomašev, N.; Pearce, A.; Wattenberg, M.; Hassabis, D.; Kim, B.; Paquet, U.; and Kramnik, V. 2022. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences* 119(47).

Menghani, G. 2023. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Comput. Surv.* 55(12).

Miranda, M.; Sanchez-Ruiz, A. A.; and Peinado, F. 2019. Towards human-like bots using online interactive case-based reasoning. In Bach, K., and Marling, C., eds., *Case-Based Reasoning Research and Development*, 314–328. Springer International Publishing.

Moral, R. C. B.; Paulus, G. B.; Assunção, J. V. C.; and Silva, L. A. L. 2020. Investigating case learning techniques for agents to play the card game of truco. In *2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, 107–116.

Neto, H. C., and Julia, R. M. S. 2018. Ace-rl-checkers: decision-making adaptability through integration of automatic case elicitation, reinforcement learning, and sequential pattern mining. *Knowledge and Information Systems* 57(3).

Powell, J. H., and Hastings, J. D. 2006. An empirical evaluation of automated knowledge discovery in a complex domain. In *Workshop on Heuristic Search, Memory*

Based Heuristics and their Applications: Twenty-First National Conference on Artificial Intelligence (AAAI–2006), 65–70. Technical Report WS-06-08.

Powell, J. H.; Hauff, B. M.; and Hastings, J. D. 2004. Utilizing case-based reasoning and automatic case elicitation to develop a self-taught knowledgeable agent. In Fu, D., and Orkin, J., eds., *Challenges in Game Artificial Intelligence: Papers from the AAAI Workshop (Technical Report WS-04-04)*, 77–81. AAAI Press.

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., *Proceedings of the Sixth International Conference on Case-Based Reasoning (ICCBR-05), LNAI 3620*, 397–407. Chicago, Illinois: Springer.

Samuel, A. L. 1959. Some studies in machine learning using the game of checkers. *IBM Journal on Reseach and Developement* 3:211–229.

Samuel, A. L. 1967. Some studies in machine learning using the game of checkers, ii – recent progress. *IBM Journal on Reseach and Developement* 11:601–617.

Shakya, A. K.; Pillai, G.; and Chakrabarty, S. 2023. Reinforcement learning algorithms: A brief survey. *Expert Systems with Applications* 231.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T.; Simonyan, K.; and Hassabis, D. 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* 362(6419):1140–1144.

Sinclair, D. 1998. 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*, 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.

Sojka, M. 2022. Performance comparison between selected chess engines. *Journal of Computer Sciences Institute* 24:228–235.

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

Talaei Khoei, T.; Ould Slimane, H.; and Kaabouch, N. 2023. Deep learning: systematic review, models, challenges, and research directions. *Neural Computing and Applications* 35(31).

Tomasev, N.; Paquet, U.; Hassabis, D.; and Kramnik, V. 2020. Assessing game balance with alphazero: Exploring alternative rule sets in chess. *CoRR* abs/2009.04374.

Watson, I. 1996. Case-based reasoning is a methodology not a technology. *Knowl. Based Systems* 12:303–308.

Yang, D. 2024. Stockfish. `https://stockfishchess.org`, Accessed: 2024-01-04.

Ye, X.; Leake, D.; and Crandall, D. 2022. Case adaptation with neural networks: Capabilities and limitations. In Keane, M. T., and Wiratunga, N., eds., *Case-Based Reasoning Research and Development*, 143–158. Springer International Publishing.