An Alternative Approach for Playing Complex Games like Chess: Evaluating The Effects of Patterns by Falsification.

Jan Lemeire
ETRO Dept.
Vrije Universiteit Brussel, Belgium
jan.lemeire@vub.ac.be

Abstract

Computer algorithms for game playing rely on a state evaluation which is based on a set of features and patterns. Such evaluation can, however, never fully capture the full complexity of games such as chess, since the impact of a feature or a pattern on the game outcome heavily relies on the game's context. It is a well-known problem in pattern-based learning that too many too specialized patterns are needed to capture all possible situations. We hypothesize that a pattern should be regarded as an opportunity to attain a certain state during the continuation of the game, which we call the effect of a pattern. For correct game state evaluation, one should analyze whether the desired effects of the matched patterns can be reached. Patterns indicate opportunities to reach a more advantageous situation. Testing whether this is possible in the current context is performed through a well-directed game tree exploration. We hypothesize that this can be done more efficiently than traditional tree search. We argue that this approach comes closer to the human way of game playing. An implementation of this algorithm must, however, rely on a yet inexistent pattern engine.

1 Introduction

The current successful chess playing computer algorithms are based on a minimax tree exploration of all possible game continuations. The analysis is performed up to a certain point in the future. An evaluation function estimates the chance of winning at the leaves of the tree. Cognitive scientists, however, consider brute-force approaches not at all reflective of the way humans rationalize game playing. Humans make abundantly use of patterns, for evaluating game positions and for effectively pruning the search tree.

Explanation-based algorithms offer an alternative approach, which tries to mimic human game playing. In explanation-based learning (EBL), prior knowledge is used to analyze, or explain, how each observed training example satisfies the target concept [MKKC86]. This explanation is then used to distinguish the relevant features of the training examples from the irrelevant, so that examples can be generalized into patterns based on logical rather than statistical reasoning. But this approach suffers from the same problem as other algorithms based on pattern-based evaluation functions. In EBL, the explanations must give the sufficient and necessary conditions for a pattern to be successful. For a complex game like chess, patterns that have to capture all aspects of a game become too complex for being applicable.

We propose an approach combining pattern-based EBL and a tree exploration. It is based on a form of knowledge that is yet unexploited by computer algorithms: the impact of patterns on the game outcome is determined by a number of future game states, called the effects. But, in contrast with current approaches, the patterns should not contain all necessary conditions for achieving the effects. A correct evaluation of the impact of a pattern on the game outcome must check whether the effects of the pattern can be reached during the continuation of the game. Patterns denote opportunities, a well-directed tree search is used to confirm or falsify the opportunities.

The next section gives an overview of current chess algorithms. Their major shortcoming, an incorrect evaluation, is discussed in Section 3. Section 4 gives our hypothesis about the information that can be used for evaluation. Based on this new kind of knowledge, section 5 develops an alternative approach to game playing. Section 6 shows how this approach resembles the human way of chess playing. Finally, Section 7 investigates how the hypothesis could be verified.

2 Current Pattern-based Chess Algorithms

Besides the abundant game playing research in optimizing the brute-force minimax search much work is done on learning algorithms. They try to mimic human game playing. Evaluation functions are constructed based on patterns: the weights of all matched patterns are combined into a single evaluation function value. In the MORPH chess program [GL91] for example, the system is responsible for finding a useful set of features (patterns) for evaluating states and for determining their significance (in the form of a weight).

Another application of patterns is in decision-making [Mit97], patterns are used to map states to operators: a pattern defines the region in the state space for which an operator leads toward the optimal solution. Hoyle is a pattern-based program that learns to play two-person, perfect information, finite-board games [EGL96]. It learns patterns that guide the decision-making of a pattern-based agent. Based on experimental data, it seeks patterns that were persistently associated with wins, losses and draws, or that achieve a goal or subgoal.

In explanation-based learning (EBL) [MKKC86], a *pattern* denotes an advantageous situation. The learning module then tries to define the states that can lead to the pattern. These states are called *explanations* for the pattern. Consider the task of learning to recognize chess positions - the explanations - in which "one's queen will be lost within the next few moves" - the pattern [MT96]. In a particular example, the queen could be lost due to a fork, in which "the white knight is attacking both the black king and queen". A fork is, however, hard to define correctly. One has to capture all situations in which the pattern leads to a successful outcome. All counter-plans that are available to the opponent for saving both its threatened pieces have to be excluded. In its review of computer chess approaches, Fürnkranz also reports on the difficulty of formalizing common chess concepts such as the fork [Für01, p. 25]:

> "However, even simple patterns like a knight fork are non-trivial to formalize. The basic pattern for a knight fork is a knight threatening two pieces, thereby winning one of them. In the endgame, these might simply be two unprotected pawns (unless one of them protects the other). In the middle game, these are typically higher-valued pieces (protected or not). However, this definition might not work if the forking knight is attacked but not protected or even pinned. But then again, perhaps the attacking piece is pinned as well. Or the pinned king can give a discovered check..."

Minton [Min84] and Epstein [EGL96] highlight the same problem of learning too many too specialized rules with explanation-based learning. Even in simple games, such as tic-tac-toe, 45 concepts were learned with 52 exception clauses [FU91]. To tackle this problem, pattern-learning algorithms insert mechanisms to limit the number and complexity of the induced rules.

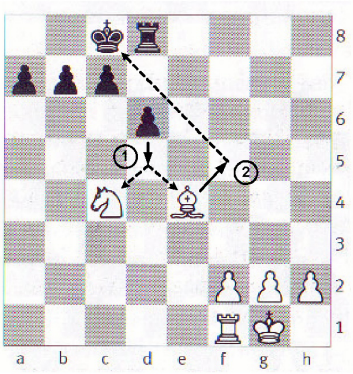

Figure 1: Fork situation, the move is with black

For complex games like chess, exact definitions of successful patterns must include an abundant number of exceptions, caused by the many possible configurations of the other pieces on the chessboard. All possibilities that are generated by the other pieces on the chessboard need to be described by as many exception clauses. We argue that in complex games such patterns become too complex if they want to capture all conditions that guarantee a favorable position. Explanation-based algorithms rely on explanations (eg. a successful fork) that inevitably lead to a pattern (eg. a rook capture). This is based on the following assumption:

$$state \perp\!\!\!\perp pattern \mid Explanations_{pattern}(state) \quad (1)$$

with $Explanations_{pattern}(state)$ the explanations of $pattern$ that apply for $state$. $A \perp\!\!\!\perp C \mid B$ stands for the independence of A and C given B. It says that A has no more information about C once B is known.

Our hypothesis is based on a problem with state evaluation. In this paper we develop an approach that overcomes this problem.

3 Evaluation fails

All game-playing algorithms rely in one way or another on an evaluation of game states. Either to measure the advantageousness of state or to select the most promising move The problem with complex games such as chess is that a correct evaluation cannot be cannot reduced to a linear (or non-linear) combination of features or patterns. Evaluation of pattern combinations heavily depends on game context. There will always be exceptions that contradict the evaluation.

Take the example of Fig. 1. It represents a position where black can make a fork with move 1. A *fork* is when a piece

threatens to capture two pieces, not all of which can be blocked. This is advantageous for black, since one of the two white pieces will be lost. On the other hand, white has an advantageous move to its disposal. He can make chess with the bishop on $f5$, by move 2 in Fig. 1. It is not immediately clear which player improves its situation. It depends on the specific combination of both patterns. If the white player can move a piece out of the fork position by making chess, black is obliged to protect his king and white will be able to retreat the other piece too. Analogous situations can even be more complex, possibly black can react on the chess by threatening another piece, so that he keeps a double threat. We argue that for a pattern indicating a good move, there are many patterns that could interfere and counter the advantage. In the fork example, it could be possible that even after the retreat of white pieces, black still got a positional advantage, because of having chased both pieces from their positions.

As shown by the example, a quasi-unlimited number of counter moves, generated by the context in which the pattern appears, exist that can neutralize the effects. Assume that multiple patterns co-occur. The total evaluation depends on the interaction of the patterns: the influence of one pattern on the other determined by the context (or instantiation) in which they appear. If the total evaluation cannot be expressed by a function that simply combines the influences of each pattern, we call them *interfering patterns*. Fig. 2 shows the possible situations for two patterns, p_1 and p_2, where P_1 and P_2 represent the states in which the patterns occur respectively. Each pattern defines a region of the state space. Assume that no other patterns are present. For example, P_1 can be a fork and P_2 the possibility of the opponent to give check. The context in which both patterns appear determine whether the fork threat can be countered by the check. A pattern can be classified as beneficial if the other player has no counter moves, as in $P_1 \setminus P_2$ or $P_2 \setminus P_1$. On the other hand, the evaluation of the intersection $P_1 \cap P_2$ depends on the specific context of the patterns. In one context p_1 can neutralize p_2, while in another the advantages of p_2 win over those of p_1. Evaluation by a weighted sum of the advantages of both patterns is clearly inadequate. The intersection can be divided in subsets denoting each interaction mode of both patterns. For correct evaluation, all subsets should be identified and described. The number of combinations increases in an exponential way with an increasing number of interfering patterns. We will propose an alternative way of evaluating these combinatorial situations correctly.

For adequate evaluation functions the features or patterns on which they are based must contain all information on the outcome of the game. This condition can be written as:

$$state \perp\!\!\!\perp outcome \mid Patterns(state) \qquad (2)$$

where $Patterns(state)$ stands for the patterns that apply

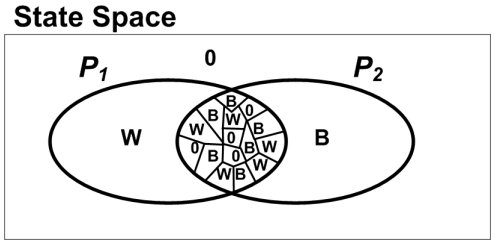

Figure 2: Evaluation of two interfering patterns P_1 and P_2, with W advantageous for white, B for black and 0 for a balanced position

for $state$. For any evaluation function based on a set of patterns or features, Eq. 2 must hold. No matter what kind of function is used, such as neural networks by NeuroChess [Thr95], the optimal function relies on the informative quality of the features. Many research is done on finding the optimal evaluation function, such as reinforcement learning or temporal-difference learning [Sut88]. Most applications use a linear function of features and try to find a set of weights that correctly classify game states.

In contrast, if only a few pieces are left on the chess board, called an *endgame*, a set of relatively simple rules can determine who will inevitably win the game.

4 Hypothesis

Our analysis is based on the observation that the outcome of a game is determined by the exact interaction of the patterns and heavily depends on the context of the game state. Trying to describe all the interactions leads, by the complexity of the game, to an enormous amount of rules or patterns. We hypothesize that the influence of a pattern on the game outcome depends on the achievement of certain states during the continuation of the game. We call these states the *effects* of the pattern. We define a *pattern* as representing a set of resembling situations that can be described by a generic definition. The elements of the set share some attributes. So, a pattern can be evaluated by their effects:

$$pattern \perp\!\!\!\perp outcome \mid Effects(pattern) \qquad (3)$$

with $Effects(pattern)$ indicating whether the effects of the pattern become true during the continuation of the game. The influence of a pattern on the game outcome is completely described by its positive and negative effects, The difference with the explanation-based approach is that we do not expect the game always to reach the effect in the presence of the pattern. Note the difference in terminology. What Explanation-Based Learning (EBL) calls an 'explanation' corresponds to our 'pattern' and what they call 'pattern' is for us an 'effect'.

The game can be analyzed by the set of existing patterns and whether their effects can be achieved. This is expressed

by the following independence:

$$state \perp\!\!\!\perp outcome \mid Effects(Patterns(state)) \quad (4)$$

This equation also states that whether a state leads to an effect depends on the set of patterns that apply for that state. Note that the evaluation of the effects can depend on the specific context too. By attacking a piece for example, the piece is forced to move to a safer position. The effectiveness of the attack depends on which position is more beneficial for the opponent. In such cases, the effects of the effect should be evaluated in a similar way - by evaluating the interaction with other patterns.

Patterns denote opportunities of the form $pattern \rightarrow effect$, where the statement should be read as 'in absence of interfering patterns, the pattern leads to the effect'. A pattern *interferes* with another pattern, if, in the context of a particular game state, the pattern affects the achievement of the effect of the other:

$$effect(pattern_1^{state}) \not\perp\!\!\!\perp pattern_2^{state} \quad (5)$$

where $pattern_i^{state}$ stands for the specific configuration of the pattern in the context of $state$. Take the fork, the pattern instantiation denotes the particular piece that threatens two specific pieces of the opponent and on which locations this occurs.

Hypothesis 1 *For complex, but non-chaotic games, the evaluation of a game state can be based on a set of generic patterns and their effects, as expressed by equations 3 and 4.*

This hypothesis applies for the game of chess, while othello is a chaotic game for which the evaluation of a state can not rely on a set of generic patterns [Gin98]. In Othello it is very difficult to foresee the effects of a move. Moves that look very similar can result in total different positions, differences that cannot be explained by generic patterns.

The fork is a combinatorial, tactical pattern. Our hypothesis also applies for long-term strategic patterns. The sacrifice of a piece is advantageous if it give the player the initiative which can lead to a decisive attack. The opponent will try to neutralize the threats. A weak king's defense is bad if the opponent can take advantage of it. He could for example try to get a succesful attack. Or he can try to exploit the threat of attacking the king to get to a more advantageous position on another part of the board.

Note that a pattern can have side effects. The move that leads to a fork can be of great influence on the rest of the game even if it did not result in the capture of a piece. However, such situations depend on random coincidences or on other patterns that are implied by the move. Secondly, a pattern incorporates other patterns having their own specific effects, like a 'fork' also is a 'threat' or an 'attack' pattern. These accidental and other effects have nothing to do with the fork pattern as such.

5 Game playing

We will now propose a way of game playing based on the hypothesis of the previous section. Eq. 4 states that the effects of patterns suffice to evaluate a game state. The effects describe future situations. Patterns that appear in the current state denote which effects can possibly be attained. However, to correctly identify whether the attempt to achieve an effect is realistic the combination of all interfering patterns should be evaluated by analyzing the continuation of the game. This corresponds to a game tree exploration that is only expanded for relevant moves, moves that influence the $pattern \rightarrow effect$ plan. They are identified by the interfering patterns. The possible effects of the patterns define exactly to what point the game tree should be explored. This is expressed by Eq. 3.

Take the game tree of Fig. 3. Assume that the white player considers playing move a by which he arrives at a position in which pattern 1 is true. He hopes of achieving one of the advantageous effects of the pattern. The black player sees two possible counter moves. If he chooses for move c, however, white can collect the benefits of pattern 1 with move e. This is not possible if black chooses for move d. White can then play f or g, but in both cases black neutralizes the threat of pattern 1 with moves h and j respectively. Both moves bring the game in a state in which the positive effects of the pattern cannot be attained anymore.

The feasibility of this approach relies on a second hypothesis. We hypothesize that the moves *interfering* with the pattern can be identified so that only those have to be explored. Other moves can be classified as being irrelevant; they do not approximate white to the achievement or falsification of pattern 1's effect. The game tree can be pruned effectively.

Hypothesis 2 *We have an algorithm that can differentiate between interfering and non-interfering patterns.*

Our approach comes close to *lazy explanation-based learning* [Tad89]. Lazy EBL does not try to learn explanations that include all conditions to guarantee that the pattern will be achieved. It learns over-general, overly optimistic plans, which are generated by simply not checking whether the moves that led to the achievement of a goal were forced. If one of these optimistic plans predicts that its application should achieve a certain goal, but it is not achieved in the current game, the opponent's refutation is generalized to a counter-plan that is indexed with the original, optimistic plan. The next time this plan is invoked, the program is able to foresee this refutation.

Our approach also resembles those used in complex scheduling and planning problems [Min88], such as PRODIGY [VCP+95]. They identify goals, decompose them into subgoals, seek plans for attaining the goals, etc.

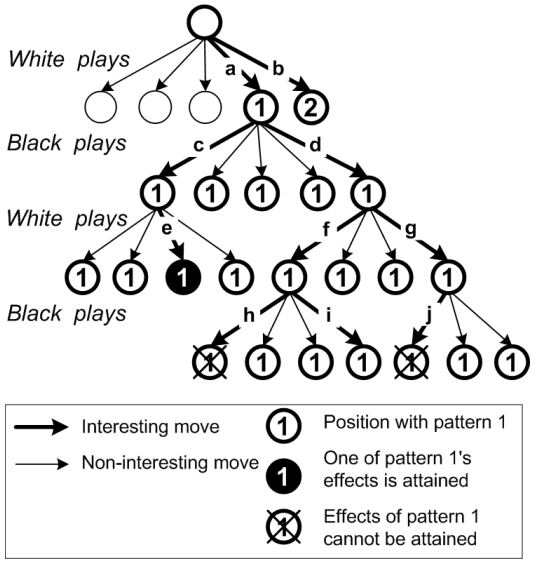

Figure 3: Game tree exploration by looking at patterns and their possible effects

The difference is that we check through falsification which goals (what we called the effects) are feasible in the context of the current state.

6 Observations on human-like game playing.

Our approach, which combines pattern-based evaluation and a well-focused tree search, gives a strategy for game playing that confirms observations on the way humans play chess.

Psychological studies have shown that the differences in playing strengths between chess experts and novices are not so much due to differences in the ability to calculate long move sequences, but to which moves they start to calculate. Humans can effectively prune the game tree and avoid waisting time on fruitless lines of investigation. For this pre-selection of moves chess players make use of patterns [Für01]. Cowley and Byrne showed that chess experts rely on falsification [CB04]. There are two main ways people can test the truth of their hypotheses. One can either seek *confirmation*: evidence that is consistent with a hypothesis, or *falsification*: evidence that is inconsistent with a hypothesis. The results of the research show that chess masters were readily able to falsify their plans. They generated move sequences that falsified their plans more readily than novice players, who tended to confirm their plans.

Another major difference with computer chess is the ability to analyse games and explain decisions. Humans can easily answer the *why* questions. We can explain the impact of a move, why a player won or why a pattern was successful. We can exactly pinpoint the moves that were decisive, determine whether the chosen tactic was successful or not. This is more information than any current chess program can provide. We postulated that all relevant aspects of a game state can be captured by a set of generic patterns and their interference. This allows accurate evaluations, which only get wrong when relevant patterns were overlooked. Computer chess algorithms act in the opposite way. They consequently generate all possible moves, but their evaluations are only estimations. By the effect of patterns we know exactly what to look for in the future, while computer algorithms rely on guesses.

Finally, it's well-known that humans have difficulties formally defining the knowledge they use. Our approach can explain this. A pattern only denotes an opportunity. A precise description of the states in which it is successful is not necessary, a well-directed tree search is used to confirm or falsify the hypothesis. Note that counter moves may be added to the pattern definitions as exception clauses. But this is not really necessary, because an analysis, based on interfering patterns, will falsify ineffective patterns.

7 Hypothesis verification and implementation.

4 ways can be explored to verify the hypotheses, but all have them are difficult to perform.

7.1 Cognitive Study

The most important evidence for the hypotheses is the way we, humans, play games, as demonstrated by the study of Cowley and Byrne [CB04]. Using results from cognitive science as evidence relies on the premise that *if the human brain is capable of calculating something, an algorithm exists for it*. It is, however, difficult to measure the algorithm the brain is using to perform a certain task.

7.2 Hypothesis 1 validation

We designed some experiments to test the first hypothesis, as described by Eq. 3. Experiments were performed with equality outcome positions by using the CAPA engine [Ben], which is well-written code and easy modifiable in order to design the experiments. We added a random element, the engine chooses between the moves that are rated the highest and therefore considered almost equivalent. We tried to study the impact of the fork pattern on the outcome of the game. After each move, the fork pattern is identified and once a fork is detected, during the continuation of the game, it is tested whether a piece could be captured..

We could, however, not get a decisive conclusion yet. It was difficult to design random experiments and to only measure the impact of 1 pattern.

7.3 Theoretical proof

If the hypotheses are true, there must exist a theoretical proof for it. To my intuition, this must rely on the exact definition of the pattern that *caused* an advantageous situation.

7.4 Implementation

The implementation of this algorithm must be based on an advanced pattern detection engine. It should not only be able to define and recognize patterns, but also to identify which patterns interfere with each other. No such powerful engine currently exists. Certainly not if we compare it to the pattern engine humans have. We can learn and identify patterns very fast and easily, and we also see in a glance which patterns affect each other.

A fork pattern is relatively easy to implement, but how to define 'pressure', a 'weak defense' etcetera. Moreover, patterns heavily relate to each other; a fork pattern is related to two threat patterns. How this should be taken into consideration is not clear.

Finally, it is totally unclear how to reason with patterns. Take the following sentence: "White attacks two black pieces with a fork, one of the pieces can make chess. White thus has to move its king and black can bring his second piece into safety."

It must be noted that an implementation might be possible for a less complex game.

8 Conclusions

This paper developed an hypothesis about a new form of generic knowledge that can be used for playing complex games like chess. The knowledge is in the form of patterns, where a pattern is a generic description - in terms of configurations of interaction between squares and pieces - that defines a set of states sharing some features. But evaluations cannot rely on these patterns alone. The influence on the outcome of the game depends on the exact interaction of the patterns in the context of the game state. Trying to describe all these interactions will lead, by the complexity of the game, to an enormous amount of rules or patterns. Current pattern-based algorithms suffer from this complexity. We hypothesize that the influence of patterns on the game outcome can be evaluated by the achievement of their effects. The game playing strategy that follows from this hypothesis combines an advanced pattern detection engine with a well-focused tree search. Absence of such a pattern engine, however, makes an implementation currently not feasible. We showed that the proposed strategy strongly resembles the human way of chess playing.

References

[Ben] Enrico Benedetti. The capa chess engine. `http://capa.sourceforge.net`.

[CB04] M. Cowley and R. M. J. Byrne. Chess masters hypothesis testing. In *Proceedings of the 26th Annual Conference of the Cognitive Science Society. Mahwah, NJ*, pages 250–255, 2004.

[EGL96] Susan L. Epstein, Jack J. Gelfand, and Joanna Lesniak. Pattern-based learning and spatially-oriented concept formation in a multi-agent, decision-making expert. *Computational Intelligence*, 12(1):199–221, 1996.

[FU91] Tom Fawcett and Paul E. Utgoff. A hybrid method for feature generation. In *Machine Learning: Proc. of the Eighth Int. Workshop*, pages 137–141. Morgan Kaufmann, 1991.

[Für01] Johannes Fürnkranz. Machine learning in games: A survey. In J. Fürnkranz and M. Kubat, editors, *Machines that Learn to Play Games*, pages 11–59. Nova Science Publishers, Huntington, NY, 2001.

[Gin98] Matthew L. Ginsberg. Computers, games and the real world. in special issue: Exploring intelligence. *Scientific American*, 1998.

[GL91] Jeffrey Gould and Robert A. Levinson. Method integration for experience-based learning. Technical Report UCSC-CRL-91-27, UCSC, Santa Cruz, CA, August 1991.

[Min84] Steven Minton. Constraint-based generalization: Learning game-playing plans from single examples. In *Proceedings of the 2nd National Conference on Artificial Intelligence, Austin, TX*, pages 251–254, 1984.

[Min88] Steven Minton. *Learning Effective Search Control Knowledge: An Explanation-Based Approach*. Kluwer, Boston, MA, 1988, 1988.

[Mit97] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[MKKC86] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1:47–80, 1986.

[MT96] Tom M. Mitchell and Sebastian B. Thrun. Learning analytically and inductively. In David M. Steier and Tom M. Mitchell, editors, *Mind Matters: A Tribute to Allen Newell*, pages 85–110. Lawrence Erlbaum Associates, Mahwah, New Jersey, 1996.

[Sut88] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[Tad89] Prasad Tadepalli. Lazy explanation-based learning: A solution to the intractable theory problem. In *IJCAI*, pages 694–700. Morgan Kaufmann, 1989.

[Thr95] S. Thrun. Learning to play the game of chess. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems (NIPS) 7*, Cambridge, MA, 1995. MIT Press.

[VCP+95] M. Veloso, J. Carbonell, A. Pérez, D. Borrajo, E. Fink, and J. Blythe. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1):81–120, 1995.