Case-Based Evaluation in Computer Chess*

Yaakov Kerner

Department of Mathematics and Computer Science
Bar-Ilan University
52900 Ramat-Gan Israel
kerner@bimacs.cs.biu.ac.il

Abstract. Current computer-chess programs achieve outstanding results in chess playing. However, there is a deficiency of evaluative comments on chess positions. In this paper, we propose a case-based model that supplies a comprehensive positional analysis for any given position. This analysis contains evaluative comments for the most significant basic features found in the position and a general evaluation for the entire position. The analysis of the entire position is presented by an appropriate Multiple eXplanation Pattern (MXP), while the analysis of each chosen feature is presented by a suitable eXplanation Pattern (XP). The proposed analysis can improve weak and intermediate players' play in general and their understanding, evaluating and planning abilities in particular. This model is part of an intelligent educational chess system which is under development. At present, our model deals only with a static evaluation of chess positions; addition of searching and playing modules remains for future work.

Keywords: case-based reasoning , computer chess, explanation patterns

1. Introduction

Case-Based Reasoning (CBR) in computers has been successfully employed in several domains like law (Ashley & Rissland, 1987) and medicine (Koton, 1988). Another potentially exciting CBR domain is game playing in general and chess in particular, since human players use extensive knowledge in their playing. However, little research has been done on CBR in game playing in general and in chess in particular. In CBR research on non-chess playing programs, we find treatment in the games of eight-puzzle (Bradtke & Lehnert, 1988) and othello (Callan et al., 1991).

Current game-playing programs use various types of brute-force search, relying on a heuristic evaluation function to evaluate positions. However, most game-playing programs do not make the evaluation process explicit. That is, there is a deficiency of evaluative comments concerning given positions. General evaluative comments

* This research is in partial fulfillment of the requirements towards the degree of Ph.D. by the author under the supervision of Dr. Uri J. Schild at Bar-Ilan University. It was supported in part by the "Ben-Gurion Fund for the Encouragement of Research" Histadrut - The General Federation of Labour in Israel.

concerning given positions have been supplied by several systems (e.g., Michie, 1981; Berliner & Ackley, 1982; Epstein, 1989; Pell, 1994). Nevertheless, these systems do not supply any detailed evaluative comments about the internal content of the given positions. Moreover, these systems are not case-based systems. We believe that using CBR can contribute to the task of giving detailed evaluative comments about the internal content of the given positions.

Our goal is development of an intelligent educational chess system. In this paper, we propose a case-based model that supplies an analysis for any chess position (except for illegal and mate positions). Our model is not currently useful to strong players since it does not use any search. Instead, we analyze the given position by analyzing the most significant basic features found in the position. The model should be helpful to weak and intermediate players wishing to improve their play. The proposed analysis is mainly directed at teaching chess evaluation and planning.

This paper is organized as follows: Section 2 gives background describing knowledge-based position evaluation and evaluative comments. In Section 3 we present our basic chess patterns. Section 4 describes suitable data structures for evaluation of chess positions. In Section 5 we propose several algorithms for evaluating chess positions. Section 6 presents a short example for evaluation of a position based on one of our case-based algorithms. Section 7 summarizes the research and suggests future directions. In the Appendix we give a glossary of the main chess concepts mentioned in the paper.

2. Background

2.1. Position Evaluation in Chess

The quality of the player's evaluation capability is one of the most important factors in determining his strength as a player. Evaluation of a position allows a chess player to work out plans and to decide which specific variations to calculate. Evaluation combines many different factors, each factor with its own weight, depending upon the factor's relative importance (Samuel, 1967).

A psychological study by de Groot (1965) shows that chess players base their evaluations of chess positions on patterns (typical positions) gained through experience. The player's patterns guide him either in deciding which move to play or rather, which strategy to choose in a given position. Simon (1974) estimates that a master has an estimated repertoire of between 25,000 and 100,000 patterns.

2.2. Case-Based Position Evaluation in Computer Chess

Chess playing programs do not involve case-based evaluation in the sense of retrieving a similar position and adapting its evaluation to the position at hand. These programs use hash tables. Hash tables are optimization tables that store positions that have already been evaluated in previous searches (Zobrist, 1970). However, hash tables are used only to prevent recalculations of the same positions and different

kinds of symmetric positions, such as: white and black symmetric, vertical symmetric, horizontal symmetric and diagonal symmetric. These tables cannot pertain to analogous positions.

A pseudo-CBR evaluation process is proposed by Levinson and Snyder (1991). Their system, called Morph, splits the given position into several chosen patterns that have been already evaluated and computes the evaluation value of the entire position based on the values of the chosen patterns. However, Morph is restricted to lower level tactical or piece position patterns with only limited ability to abstract from these. MorphII (Levinson, 1994) has addressed these concerns, by abstracting new and wider patterns from the rules of the domain and additional patterns from those. However, MorphII's patterns may not really coincide with the way human would classify the position and thus only have limited explanatory use.

To sum up, Morph and MorphII do not supply any detailed evaluative comments about the given position. Moreover, they do not use CBR in the sense of retrieving a similar position and adapting its evaluative analysis to the position at hand.

2.3. Evaluative Comments

Chess experts have established a set of qualitative evaluation measures. Each chess position can be evaluated by one of these measures. Table 1 presents a few qualitative measures, their equivalent quantitative measures and their meanings. These measures are based on the common relative values of the queen, rook, bishop, knight and pawn which are 9, 5, 3, 3 and 1 points, respectively (Shannon, 1950).

Table 1. A few qualitative measures (LMs) in chess, their equivalent quantitative measures (NMs) and their meanings. The LMs are intervals of NMs, which are numbers based roughly on positional evaluations.

Qualitative measures	Quantitative measures	Meaning
+−	3 < NM	White is winning
±	1 < NM <= 3	White has a big advantage
±	0 < NM <= 1	White has a small advantage
=	NM = 0	The game is even
∓	-1 =< NM < 0	Black has a small advantage
∓	-3 =< NM < -1	Black has a big advantage
−+	NM < -3	Black is winning

Little research has been done concerning the task of giving detailed evaluative comments for game positions. Most game-playing programs do not make the evaluation process explicit, but rather give only one evaluative score. A chess student is not always capable of understanding why the specific chess program he is working with, evaluated the position the way it did. In order to increase his evaluating ability effectively, he needs to receive explanatory evaluative comments. However, there are

programs that can supply a more detailed explanation concerning an evaluated position. A few such programs are presented below.

A theory of evaluative comments has been proposed by Michie (1981). In addition to the construction of the classical minmax game tree, Michie has developed a model of fallible play. His theory assigns to each position two values: "game-theoretic value" and "expected utility." Based on combinations of these values, his theory supplies short comments on chess positions, e.g., "Black has a theoretical win but is likely to lose." However, his theory does not supply any comments about the internal content of the evaluated positions, e.g., the pawn-structure. Moreover, this theory does not suggest any plans for the continuation of the game.

Another explanation mechanism has been constructed by Berliner and Ackley (1982). Their system, called QBKG, can produce critical analyses of possible moves for a backgammon position using a hierarchical knowledge tree. It gives only two kinds of comments. The first is a general evaluation of the discussed position. The second is an answer to the question: "Why did you make that move, as opposed to this move?" In addition, Berliner and Ackley admit that their system is only able to produce comments on about 70% of positions presented to it.

In the last years, a few additional programs that can explain their positions, have appeared (e.g., HOYLE (Epstein, 1989); METAGAMER (Pell, 1994)). HOYLE and METAGAMER view a feature as an advisor that encapsulates a piece of advice about why some aspect of the position may be favorable or unfavorable to one of the players. Using these advisors, these programs can comment generally on positions.

To sum up, little research has been done concerning case-based detailed evaluations of game positions in general and case-based detailed evaluations of chess positions in particular. Michie, Berliner and Ackley, Epstein and Pell contribute to the task of giving general evaluative comments concerning game positions. Nevertheless, their models and, to the best of our knowledge, other existing models do not supply any detailed evaluative comments concerning the internal content of the given positions. In this paper, we propose an initial framework for a detailed case-based evaluation model for computer chess programs.

3. Basic Chess Patterns

A basic chess pattern is defined as a certain minimal configuration of a small number of pieces and squares which describes only one salient chess feature. In order to discover as many different basic chess patterns as possible, we, with the help of strong chess masters, have constructed a hierarchical tree structure that includes most basic positional features concerning evaluation of chess positions.

This tree is a hierarchical classification of most common chess features at different levels of abstraction. At the root of the tree, we have the concept of "static evaluation." Each leaf (a node at the last level) in this tree represents a unique basic pattern (e.g., "one isolated pawn in the endgame stage"). Each basic pattern has two suitable explanation patterns (one for White and one for Black) that contain several important comments concerning the discussed pattern.

Most of the concepts were collected from a variety of relevant chess books (Nimzowitsch, 1930; Fine, 1952; Pachman, 1973; Pachman, 1976; Kotov, 1978). The highest level concepts of this tree are shown in Fig. 1. Figures 2 and 3 illustrate the sub-trees describing the pawn and king concepts, respectively. A few important concepts mentioned in these trees are defined in a chess glossary in the Appendix.

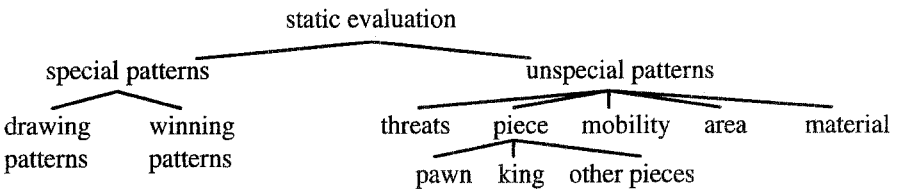

Fig. 1. The evaluation tree

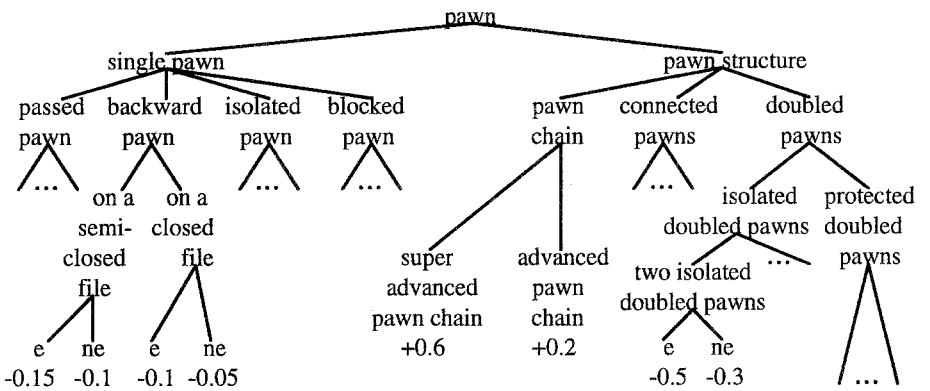

Fig. 2. The evaluation sub-tree for the pawn concept

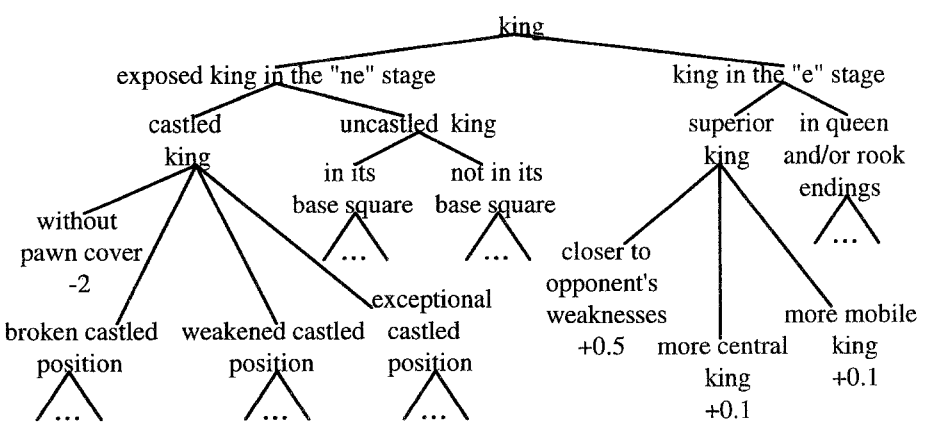

Fig. 3. The evaluation sub-tree for the king concept

Each leaf which represents a unique basic chess pattern has its own evaluative value. For example, the pattern "two isolated doubled pawns in the endgame stage" has an evaluative value of -0.5. It is important to mention that the evaluative value given to each basic pattern is only a general estimate that represents its value in the majority of the positions that include this pattern.

In addition, often when evaluating a chess pattern, we consider the stage in the game in which the pattern takes place. This distinction is important since the same pattern can be evaluated differently in the "**e**" and "**ne**" stages (the **e**ndgame stage and **n**ot the **e**ndgame stage respectively). For example, the pattern "isolated pawn" becomes a greater weakness in the "**e**" stage, since the importance of a pawn becomes greater.

This tree is used primarily for two main tasks: (1) searching the tree to find all basic patterns included in a given position and (2) determining pattern similarity in the adaptation process in our model (details in Section 5). The structure of our evaluation tree is similar in some ways to the E-MOP, the memory structure introduced by Kolodner (1981). Our chess concepts resemble Kolodner's generalized information items and our basic chess patterns can be viewed as her "events."

At present, the tree contains 613 nodes where 403 of them are leaves (i.e., basic patterns). In the next section we describe suitable data structures for evaluating any given chess positions.

4. Data Structures for Evaluation of Chess Positions

4.1. XPs

XPs are explanation patterns (Schank, 1986). Kass (1990, p. 9) regards XPs as "*variablized explanations that are stored in memory, and can be instantiated to explain new cases.*" The XP, according to Schank (1986, p. 39), contains the following slots: (1) a fact to be explained, (2) a belief about the fact, (3) a purpose implied by the fact, (4) a plan to achieve purpose, and (5) an action to take.

The XP structure has been applied to criminal sentencing (Schild & Kerner, 1993). We find this structure also appropriate for the domain of evaluation of chess positions. While the judicial XP describes a specific viewpoint of a judge concerning a sentence, the chess XP describes a specific viewpoint of a chess position from either White's point of view or Black's point of view.

In the case of chess, since we are concerned with the evaluation of the given position, we use an evaluation slot instead of an action slot. The evaluation slot contains two kinds of evaluative values: a quantitative **m**easure (NM) and a qualitative **m**easure (LM) (which are demonstrated in **T**able 1).

In our model, each basic pattern in the evaluation tree has two *general XPs* (one for White and one for Black). Six different examples of XPs are presented in **F**igures 6 and 7. For the sake of convenience, we use some abbreviations: W for White, B for Black, K for king and Q for queen. Without loss of generality, our examples will be evaluated from White's viewpoint, assuming that it is White's turn to move.

In summary, the XP-structure seems to be a convenient data-structure for describing and explaining a specific chess pattern of a given position. However each chess position usually includes more than one important pattern. Thus, the XP-structure does not suffice to explain an entire chess position.

4.2. MXPs

The MXP (Multiple eXplanation Pattern) structure (Schild & Kerner, 1993) was first introduced to assist judges in deciding which sentence to hand down in a new case. The MXP is a detailed graphical explanation to a given case and its outcome. In general, the MXP is defined as a collection of XPs and an outcome slot. Each XP represents a unique important viewpoint concerning the given case and carries its weight in its evaluation slot to the outcome slot of the entire case. In our model, each important chess viewpoint regarding the given position is explained and evaluated by a suitable XP, and the general evaluation for the entire position is represented in the outcome slot.

In order not to overload the user with too much information, we stipulate that the chess MXP is composed of the three most important XPs (those with the highest absolute evaluation values) suitable to the discussed position. At present, our evaluation function is a summation over the evaluation values of these XPs. We use this simple rough function to enable the user to understand how the system reached its general evaluation. Nevertheless, the summation is meaningful since each XP included in the MXP describes only one unique independent basic pattern. That is, the quantitative measure of the "general evaluation" slot is a summation over its XPs' quantitative values. Its qualitative measure is dependent on its quantitative measure according to Table 1.

Figures 4 and 5 present two chess positions. Figures 6 and 7 describe the MXPs that analyze these positions, respectively. The most important chess concepts mentioned in these MXPs are defined in a glossary in the Appendix.

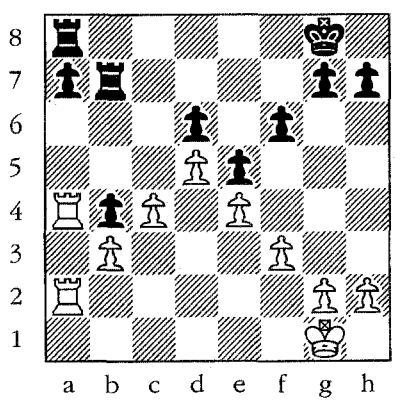

Fig. 4. Chess position 1

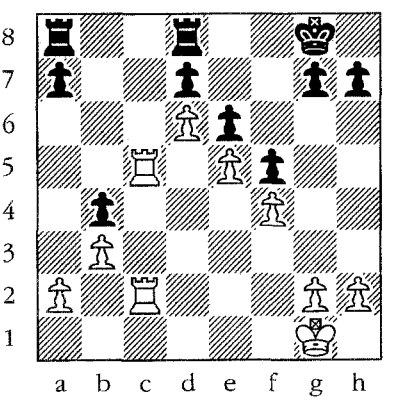

Fig. 5. Chess position 2

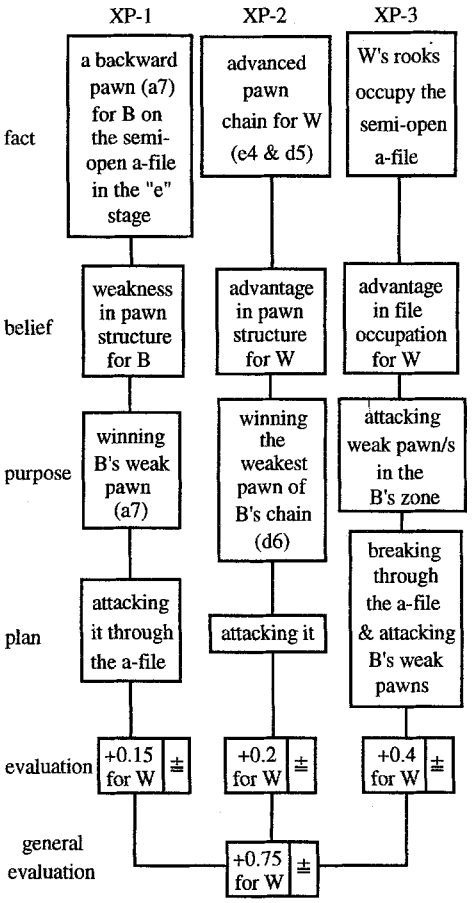

Fig. 6. MXP for position 1

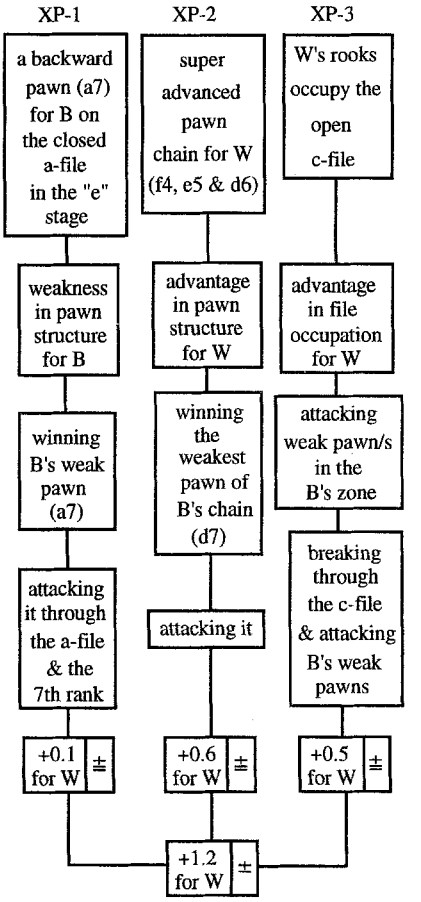

Fig. 7. MXP for position 2

Our MXP structure fits the way Steinitz (the first formal world chess champion between 1886-1894) taught players. "*Steinitz taught players most of all to split the position into its elements. Naturally they do not play the same role in a given position, they do not have the same importance. Once, he has worked out the relationship of the elements to each other, the player moves on the process of synthesis which is known in chess as the general assessment*" (Kotov 1978, p.24).

We believe that the MXP structure provides a better framework for explaining a given position than those given by other systems, because we supply comments on the internal content of the position, and our comments are more detailed. Moreover, the application of the MXP structure for evaluating chess positions shows that the MXP structure is an appropriate knowledge structure not only for sentencing criminals. These findings lead us to believe that the MXP should be examined as a suitable tool for other CBR domains where there is need to evaluate or to solve complex problems.

5. Algorithms for Evaluation of Chess Positions

5.1. A Simple Evaluation Algorithm

We aim at evaluating any given chess position by constructing a suitable MXP for it. We search our evaluation tree in order to find all basic patterns included in the position. We choose only the most important basic patterns (i.e., the patterns with the highest absolute evaluation values). We retrieve the stored XPs of these patterns and combine them into a new MXP. The retrieved XPs are analogical to the snippets (portions of cases) used in other CBR domains by Kolodner (1988), Redmond (1990) and Branting (1991).

A description of this algorithm, *Algorithm-1*, is given here. Given a New Chess Position (NCP):

(1) Find the evaluation values of all basic patterns (features) included in the NCP.
(2) For the most important patterns retrieve their XPs.
(3) Combine all these adapted XPs into a MXP.
(4) Compute the general evaluation of the MXP of the NCP using a simple summation over the evaluation values of the retrieved XPs.

This algorithm has been used primarily in the establishment of the original data base of positions and their MXPs. We use this data base for evaluating new given chess positions in our case-based evaluation algorithms described in the next subsection. At present, we have 12 representative positions. These positions have been slightly adapted from positions taken from different relevant chess books (e.g., Pachman, 1976; Kotov, 1978; Shereshevsky, 1985; Averbakh, 1986).

5.2. Case-Based Evaluation Algorithms

Case-based algorithms, more creative than Algorithm-1, can be proposed. These algorithms use the data base of positions and their MXPs. Controlled learning of new positions with their MXPs enlarges the extent of this data base and improves the explanation ability of the evaluation algorithms. General descriptions of two algorithms are given below.

Algorithm-2, given below, uses only the MXP most suitable to the NCP. Given a NCP, the algorithm is as follows:

Retrieval of suitable MXPs and Selection of the best MXP

(1) Find the evaluation values of all basic patterns included in the NCP.
(2) Choose the most important patterns according to their absolute evaluation values.

(3) Retrieve all positions that their MXPs include at least one XP whose fact-slot is either one of the patterns found in step (2) or a "brother" of one of them (according to the evaluation tree). In case of failure, jump to step (8).
(4) For each retrieved position compute its similarity measure relative to the NCP.
(5) Choose the most suitable MXP (i.e., with the highest similarity measure).

Adaptation and System Evaluation

(6) Keep exactly matched XP.
(7) Adapt suitable XPs of the chosen MXP to the NCP using the evaluation tree and suitable general XPs.
(8) Explain other important facts of the NCP by general XPs.

Construction of the solution, Real World Evaluation and Storage

(9) Combine all exactly found and adapted XPs into a new MXP.
(10) Compute the general evaluation of the new MXP by summing the evaluation values of its XPs.
(11) Test the proposed MXP by a chess expert and make optional improvements by hand where needed.
(12) If the proposed MXP is found appropriate for acquisition then store it according to the fact-slots of its XPs.

Algorithm-3 can use several MXPs suitable to the NCP. The difference between it and Algorithm-2 is step (8). In Algorithm-3 step (8) is as follows: For all important facts of the NCP not found in the facts of the MXP nor adaptable to the XPs of the MXP, select the next MXP suitable to the NCP, and return to step (6).

In the next sub-sections of Section 5 we shall give a detailed description of the most important CBR stages of *Algorithm-2*.

5.2.1. Retrieval of suitable MXPs and Selection of the best MXP

Given a NCP, we retrieve all MXPs that include at least one XP whose fact-slot is either one of the patterns found in step (2) or a "brother" of one of them (according to the evaluation tree). The MXP with the highest similarity measure (*sm*) to the NCP is chosen for the next stage of our CBR algorithm. Our similarity function has the following form: $sm = a*ifs + b*ps$ where *sm* is the computed similarity measure, *a* and *b* are specific constants, *ifs* is the important features' similarity, and *ps* is the position's similarity. The *ifs* is similar to the *contrast measure* of Tversky (1977) and the *ps* is similar to the *nearby measure* of Botvinnik (1984). Intuitively, the *ifs* and *ps* can be regarded as a semantic similarity and a structural similarity, respectively.

The *ifs* function is the computed similarity measure between the important facts (basic patterns) found in the NCP and the retrieved MXP. It is defined as follows: Let S1 be the set of all facts found both in the NCP and in the retrieved MXP. Let S2 be the set of all facts of the NCP for which we found near-neighbors patterns to them in the retrieved MXP (according to the evaluation tree). Let S3 be the set of all facts

found in the NCP but without near-neighbor facts in the retrieved MXP. Let S4 be the set of all facts found in the retrieved MXP but without near-neighbor facts in the NCP. Then, $ifs = \alpha * \sum_{i \in S1} W_i + \beta * \sum_{i \in S2} (W_i * d_i) - \gamma * \sum_{i \in S3} W_i - \delta * \sum_{i \in S4} W_i$ where α, β, γ and δ are specific constants, w (weight) is the evaluation value of every discussed fact, whether it is an important fact found in the NCP or it is a fact-slot of a XP of the discussed MXP, and d a near-neighbor factor that measures the distance between each pair of facts.

The *ps* function measures the similarity between two positions: the NCP and the position related to the retrieved MXP. It is defined by: $ps = c*ips + d*wms + e*bms$ where c and d are specific constants, *ips* is the identical pieces' similarity, *wms* is White's material similarity, and *bms* is Black's material similarity. *ips* is defined as the number of the exact pieces found on the same squares of the two positions divided by the number of pieces found in the NCP. The *wms* and *bms* are defined as follows:

$wms = 1 - abs((wpm(NCP) - wpm(RP))/ wpm(NCP))$ *and*

$bms = 1 - abs((bpm(NCP) - bpm(RP))/ bpm(NCP))$

where *abs* is the absolute function, *wpm* is the calculated material value of White's pieces (except the king) according to Table 1, *bpm* is the same function for Black's pieces, *NCP* is the new chess position, and *RP* is the retrieved position.

5.2.2. Adaptation and System Evaluation

After choosing the most suitable MXP we adapt its XPs in order to construct a MXP for the NCP. In step (6), for the facts found both in the NCP and in the retrieved MXP, we take exactly the XPs of these facts from the retrieved MXP.

In step (7), for the facts of the retrieved MXP found as near-neighbors (according to the evaluation tree) using suitable general XPs, we process a learning process on each XP of the MXP (call each in its turn XP-1) in order to adapt XP-1 to its matching fact in the NCP. To validate the proposed adaptation we use some chess tests (e.g., a limited search). These tests are partly a simulation of the proposed adaptation and serve as the system evaluation to its own solution.

In step (8), for all important facts of the NCP that are neither found in the facts of the chosen MXP nor could be adapted reasonably to the XPs of the chosen MXP, using chess tests, we adapt suitable general XPs.

5.2.3. Construction of the solution, Real World Evaluation and Storage

A MXP for the NCP is proposed after combining all exactly found and adapted XPs and computing the general evaluation slot of the MXP, using a summation over the evaluation values of the XPs of the new MXP. A chess expert will either approve or disapprove of this MXP. In case of disapproval, at present, potential improvements are inserted by hand. In case of approval, a potential learning process is executed.

We have constructed a learning mechanism that is able to enlarge our data base of MXPs. A new MXP will be added to the data base of MXPs only if at least one

adapted XP is learned (step 7 or 8). Such a MXP is inserted in the flat data base of MXPs. The indexes that enable any kind of access (insertion or retrieval) to a MXP in this data base are the fact-slots of the XPs of the MXP.

6. A Short Example

In this section, we illustrate a use of Algorithm-2. Assuming the NCP is position 2 (**Fig.** 5), we retrieve the MXP presented in **Fig.** 6 (which is the MXP of position 1 that is presented in **Fig.** 4) as the best MXP for explaining position 2.

The adaptation process constructs the MXP presented in **Fig.** 7 as an explanation for the NCP. Due to the lack of space, we will only explain the construction of the XP relates to Black's backward pawn (a7) on the closed a-file in the "e" stage (i.e., XP-1 in **Fig.** 7) from White's viewpoint.

The fact slot of XP-1 of the NCP relates to a closed file while the fact slot of XP-1 of the retrieved MXP relates to a semi-open file. These facts are close neighbors in the evaluation tree. Therefore, we choose XP-1 of the retrieved MXP for the adaptation process. In addition, we retrieve a suitable general XP according to the discussed fact of the NCP. Using these two retrieved XPs, we construct XP-1 of the new MXP.

In the fact slot we write exactly the discussed fact of the NCP. Since the belief and purpose slots are the same in both retrieved XPs, we take them as they are. The plan slots in the two retrieved XPs are different. The plan slot of the general XP proposes to attack the weak pawn through its rank (i.e., the 7th rank). The plan slot of the XP-1 of the retrieved MXP proposes to attack the weak pawn through its file (i.e., the a-file). Utilizing an elaboration strategy, we refine the plan slot of these two retrieved XPs and construct a new plan slot, which is to attack the weak pawn through both the a-file and the 7th rank.

By using simple limited searching, we ensure that the new plan can be theoretically made on the board. We find a way to switch the White rook on c5 to the a-file (a5) and to switch the White rook on c2 to the 7th rank (c7). For the evaluation slot we take the evaluation value of the general XP since it relates to the same discussed fact.

Due to the approval of this MXP by our chess expert and to the construction of a new more complex plan slot, we store this MXP in the data base of MXPs according to the facts-slots of its XPs.

To sum up, we think that this controlled learning process will improve the evaluation ability of our algorithms because of the accumulation of new MXPs. These algorithms become more adequate by deriving more creative and better evaluations than those supplied with fewer MXPs.

7. Summary and Future Work

We have made a contribution to CBR research by extending its range in the game-playing domain in general and in computer chess in particular. We have developed a

case-based model which supplies a comprehensive positional evaluation for any chess position. This model seems to supply better explanations for chess positions than other existing computer game-playing programs. In addition, our model includes a learning mechanism that enables it to supply more adequate evaluations.

We think that this model, in principle, can be generalized for evaluating any game position for any board game. The three highest levels of our evaluation tree are appropriate for game playing in general. While the king and pawn sub-trees are unique for chess-like games, all other sub-trees (e.g., threats and material) fit in general to all board games.

In computer chess, profound understanding has been shown to be inefficient without deep searching. Therefore, to strengthen our model, there is a need to add searching capability. Case-based planning is another important issue which we have to deal with more deeply. The plans we retrieve for each XP of the MXP for the discussed position and adapt to fit the discussed position, may be combined into one complete plan using game tree search. Finally, addition of playing modules will enable our system to, besides play chess, learn in the real world and therefore to improve its evaluating, explaining and planning capabilities.

Acknowledgments

Thanks to Ofer Bruk (an international master) and Allan Savage (a FIDE master). These two strong players and experienced trainers played a crucial role in the development of the evaluation tree. Thanks also to Sean Engelson, Robert Levinson, Avraham Norin, Barney Pell, Shlomit Zeiger, Sara Zimin and the reviewers for helpful comments on earlier drafts of this paper. Finally, special thanks to my talented students Alon Geri and Danny Moore for implementing a big part of the system described in this paper as their graduation project.

References

Ashley, K. D. & Rissland, E. L. (1987). Compare and Contrast, A Test of Expertise. In *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 273-278). Los Altos: Morgan Kaufmann.

Averbakh, Y. (ed.) (1986). *Comprehensive Chess Endings* (a five-volume set). Translated by Neat, K. P. Oxford: Pergamon Press.

Berliner, H. J. & Ackley, D. H. (1982). The QBKG System: Generating Explanations from a Non-Discrete Knowledge Representation. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 213-216). AAAI Press.

Botvinnik, M. M. (1984). *Computers in Chess: Solving Inexact Search Problems.* Translated by Brown A. A. New York: Springer-Verlag.

Bradtke, S. & Lehnert, W. G. (1988). Some Experiments With Case-Based Search. In *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 133-138). San Mateo: Morgan Kaufmann.

Branting, L. K. (1991). Reasoning with Portions of Precedents. In *Proceedings of the Third International Conference on AI and Law* (pp. 145-154). New York: ACM Press.

Callan, J. P., Fawcett, T. E. & Rissland, E. L. (1991). Adaptive Case-Based Reasoning. In *Proceedings of a Workshop on CBR* (pp. 179-190). San Mateo: Morgan Kaufman.

de Groot, A. D. (1965). *Thought and Choice*. Mouton, The Hague.

Epstein, S. (1989). The Intelligent Novice - Learning to Play Better. In D. N. L. Levy & D. F. Beal (Eds.), *Heuristic Programming in Artificial Intelligence - The First Computer Olympiad*. Ellis Horwood.

Fine, R. (1952). *The Middle Game in Chess*. New York: David McKay Company.

Kass, A. M. (1990). *Developing Creative Hypotheses by Adapting Explanations*. Technical Report #6, p. 9. Institute for the Learning Sciences, Northwestern University, U.S.A.

Kolodner, J. L. (1981). Organization and Retrieval in a Conceptual Memory for Events or Con54, Where are You? In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence* (pp. 227-233). Los Altos: William Kaufmann.

Kolodner, J. L. (1988). Retrieving Events from a Case Memory: A Parallel Implementation. In *Proceedings of a Workshop on CBR* (pp. 233-249). San Mateo: Morgan Kaufman.

Koton, P. (1988). Reasoning about Evidence in Causal Explanations. In *Proceedings of a Workshop on CBR* (pp. 260-270). San Mateo: Morgan Kaufman.

Kotov, A. (1978). *Play Like a Grandmaster*. Translated by Cafferty, B. London: B. T. Batsford Ltd.

Levinson, R. & Snyder, R. (1991). Adaptive Pattern-Oriented Chess. In *Proceedings of the Ninth National Conference on Artificial Intelligence* (pp. 601-606). Menlo Park: AAAI Press/The MIT Press.

Levinson, R. (1994). Morph II: A Universal Agent: Progress Report and Proposal. Technical Report UCSC-CRL-94-22, University of California Santa Cruz.

Michie, D. (1981). A Theory of Evaluative Comments in Chess with a Note on Minimaxing. *The Computer Journal, Vol. 24, No. 3*, 278-286.

Nimzowitsch, A. (1930). *My System - A Chess Treatise*. New York: Harcourt, Brace and Company.

Pachman, L. (1973). *Attack and Defence in Modern Chess Tactics*. Translated by Clarke P. H. London: Routledge and Kegan Paul LTD.

Pachman, L. (1976). *Complete Chess Strategy* (Volumes 1 & 2). Translated by Littlewood J. London: B. T. Batsford Ltd.

Pell, B. (1994). A Strategic Metagame Player for General Chess-Like Game. In *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 1378-1385). Seattle: AAAI Press/The MIT Press.

Redmond, M. (1990). Distributed Cases for Case-Based Reasoning: Facilitating Use of Multiple Cases. In *Proceedings of the Eight National Conference on Artificial Intelligence* (pp. 304-309). Menlo Park: AAAI Press/The MIT Press.

Samuel, A. L. (1967). Some Studies in Machine Learning Using the Game of Checkers II- Recent Progress, *IBM Journal of Research and Development, Vol. 11, No. 6*, 601-617.

Schank, R.C. (ed.) (1986). *Explanation Patterns: Understanding Mechanically and Creatively*. Hillsdale: Lawrence Erlbaum.

Schild, U. J. & Kerner, Y. (1993). Multiple Explanation Patterns. In *Proceedings of the First European Workshop on Case-Based Reasoning*, Vol. II (pp. 379-384). Kaiserslautern: Germany. Extended paper in Wess, S.; Althoff, K-D.; and Richter, M. M. (Eds.), *Topics in Case-Based Reasoning - EWCBR'93*, Lecture Notes in Artificial Intelligence 837 (pp. 353-364). Berlin: Springer-Verlag, 1994.

Shannon, C. E. (1950). Programming a Computer for Playing Chess. *Philosophical Magazine, Vol. 41(7)*, 256-277.

Shereshevsky M. I., (1985). *Endgame Strategy*. Translated by Neat, K. P. Oxford: Pergamon Press.

Simon, H. A. (1974). How Big is a Chunk? *Science No. 183*, 482-488.

Tversky, A. (1977). Features of Similarity, *Psychological Review, 84, 4*, 327-352.

Zobrist, A. L. (1970). *Technical Report #88*. Computer Science Department, University of Wisconsin, Madison, WI.

Appendix

Chess Glossary

Advanced Pawn chain: White/Black's head of a series of pawns in a pawn chain is in the 5-th/4-th rank, respectively.

Backward pawn: A pawn that has been left behind by neighboring pawns of its own color and can no longer be supported by them.

Blocked pawn: A pawn that is blocked by an opponent's piece which is not a pawn.

Castled king: A king that has made a castle.

Center: Squares e4, d4, e5 and d5.

Closed file: A file with pawns of both colors.

Doubled pawns: At least two pawns of the same color on the same file.

Endgame: The last and deciding stage of the chess game. In this stage the position becomes simplified and usually contains a relatively small number of pieces.

Exposed king: A king without good defence, mainly without a good pawn cover.

Isolated doubled pawns: At least two pawns of the same color on the same file that are not protected by any neighboring pawns of their own color.

Isolated pawn: A pawn that has no neighboring pawns of its own color.

Mobility: Number of potential single moves in the current position.

Open file: A file without pawns.

Passed pawn: A pawn that has no opponent's pawns which can prevent it from queening.

Pawn chain: Two consecutive series of pawns abutting on one another in consecutive diagonals.

Protected doubled pawns: At least two pawns of the same color on the same file in which at least one of them is protected by a neighboring pawn of its own color.

Semi-closed file: A file with pawn/s of only one's own color.

Semi-open file: A file with pawn/s of only the opponent's color.

Super-Advanced Pawn chain: White/Black's head of a series of pawns in a pawn chain is in the 6-th/3-th rank, respectively.