Knowledge Discovery in Chess Databases: A Research Proposal

Johannes Fürnkranz
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Wien, Austria
E-mail: juffi@ai.univie.ac.at

Abstract

In this paper we argue that chess databases have a significant potential as a test-bed for techniques in the area of Knowledge Discovery in Databases (KDD). Conversely, we think that research in Artificial Intelligence has not yet come up with reasonable solutions for the knowledge representation and reasoning problems that are posed by knowledge-based computer chess programs, and consequently argue that KDD techniques could be useful for the advancement of various types of knowledge-based computer chess systems. Although we cannot present any concrete results, we hope to outline some fruitful directions for further research and exchange of ideas between the KDD and computer chess communities.

1 Introduction

Knowledge Discovery in Databases (KDD) or *Data Mining* is a rapidly growing research area which focuses on the discovery of useful and understandable pieces of knowledge from databases [12, 11]. On the other hand, the rapid increase in computing power of personal computers and the simultaneous fall of hardware prices had a considerable impact on the chess playing community. Most serious chess players use huge game databases, opening databases, and/or endgame databases that they can access and use for home preparation with various commercial programs. There is practically no upper limit to the size of these databases, and it is quite likely that with some effort useful knowledge can be extracted from them. However, state-of-the-art programs for analyzing these databases have only very restricted capabilities. What is needed is the support for high-level concepts that can be used for describing the knowledge contained in these databases, and allow high-level interactions like "How should I develop my pieces in this opening?", "What are promising plans in this pawn formation?", or "Show me a winning strategy for this type of endgame!". We believe that inductive learning techniques have the potential for supporting such tasks.

The purpose of this paper is to draw attention to research in *Knowledge Discovery in Chess Databases*. To that end, we will review previous work, discuss shortcomings, present directions for future work. In short, we will try to argue that chess is an interesting test-bed for ideas in the field of Knowledge Discovery in Databases.

2 Previous Work on KDD and Chess

There are a variety of approaches that employ machine learning techniques in the domain of chess [13]. Several of them can be viewed in the KDD framework. This section will briefly discuss previous work that aimed at discovering chess knowledge from databases.

The induction of chess concepts from databases is probably the task in computer chess that has been most intensively studied in machine learning research. Typically, positions from a certain chess endgame are preclassified into the categories won or not-won. Most inductive machine learning algorithms need a so-called *attribute-value representation* as an input. Each position in the training set has to be specified with exactly one value for a fixed set of predefined attributes. The learned concept description is limited to tests for certain values of the given attributes. If the positions are only represented with obvious attributes like the location of the pieces and the side to move, the programs are not capable of making useful generalizations. Additional attributes that encode potentially useful patterns like kings' opposition or the distance between certain pieces must be provided to the learning system.

The earliest work on learning from a chess database is reported in [26], where the inductive rule learning algorithm AQ [25] is applied to the KPK database described in [6]. The positions in the database were coded into 17 attributes describing important relations between the pieces. In this representation, rules with 80% predictive accuracy were learned from about 250 training examples.

Quinlan [40] describes several experiments for learning rules for the KRKN endgame. In [41], he used his decision tree learning algorithm ID3 to discover recognition rules for positions of the KRKN endgame that are `lost-in-2-ply` and `lost-in-3-ply` respectively. From less than 10% of the possible KRKN positions, ID3 was able to derive a tree that committed only 2 errors on a test set of 10,000 randomly chosen positions (these errors were later corrected in [49]). Quinlan noted that this achievement was only possible with a careful choice of the attributes that were used to represent the positions. Finding the right set of attributes for the `lost-in-2-ply` task required three weeks. Adapting this set to the slightly different task of learning `lost-in-3-ply` positions took almost two months. Thus for the `lost-in-4-ply` task, which he intended to tackle next, Quinlan experimented with methods for automating the discovery of new useful attributes. However, no results from this endeavor have been published.

A severe problem with this and similar experiments was that, although the derived decision trees were shown to be correct and faster in classification than extensive search algorithms, they were also incomprehensible to chess experts.

Shapiro [46] tried to alleviate this problem by decomposing it into a hierarchy of smaller sub-problems that could be tackled independently. A set of rules was induced for each of the sub-problems which together yielded a more understandable result. This process of *structured induction* has been employed to learn correct classification procedures for the KPK and the KPa7KR endgames [44]. An endgame expert helped to structure the search space and to design the relevant attributes. The rules for the KPa7KR endgames were generated without using a database as an oracle. The rules were interactively refined by the expert by specifying new training examples and suggesting new attributes if the available attributes were not able to discriminate between some of the positions. This rule-debugging process was aided by a self-commenting facility that displayed traces of the classification rules in plain English [45]. A similar semi-autonomous process for refining the attribute set was used in [50] to generate decision trees for the KQKQ endgame.

However, the problem of decomposing the search space into easily manageable subproblems again is a task that requires extensive collaboration with a human expert. Thus there have been several attempts to automate this process. Paterson [38] tried to automatically structure the KPK endgame using a clustering algorithm. The results have been negative, as the found hierarchy had no meaning to human experts. Muggleton [30] has applied DUCE to the KPa7KR task studied by [44]. DUCE is a machine learning algorithm that is able to autonomously suggest high-level concepts to the user. It looks for common patterns in the rule base (which initially consists of a set of rules each describing one example board position) and tries to reduce its by replacing the found patterns with new concepts. In machine learning the autonomous introduction of new concepts during the learning phase is commonly known as *constructive induction* [24]. Using constructive induction DUCE reduces the role of the chess expert to a mere evaluator of the suggested concepts instead of an inventor of new concepts. DUCE structured the KPa7KR task into a hierarchy of 13 concepts defined by a total of 553 rules. Shapiro's solution, however, consisted of 9 concepts with only 225 rules. Nevertheless DUCE's solution was found to be meaningful for a chess expert.

Bain [2, 3] tried to learn rules that predict the number of plies to a win with optimal play on both sides in the KRK endgame. This series of experiments differs from previous work in two aspects: First, the problem was not represented as a single relational table, but instead an *inductive logic programming* algorithm was used that could incorporate relational background knowledge such as checking whether two pieces are on the same file or row and checking the distances between two pieces. The other difference was that these experiments did not aim at learning predictive rules from a subsample of the database, but aimed at compressing the database by learning rules that could reproduce the entire database, but use only a fraction of the disk space that would be required for storing the entire database.

In [15] we have tried to use the same database (with slightly more complex background knowledge) for learning a playing strategy from the database. For this purpose, we generated 100 games of a player using an optimal strategy (rook

side) vs. a player that plays randomly. From all positions and their associated optimal moves and bad mistakes (dropping the rook or stale-mating), we had the ILP system ICL [8] learn predicates that check whether a given move is optimal or a bad mistake. These predicates were then used in an artificial player that generates all legal moves, rules out all moves that were deemed bad mistakes, and randomly plays one of the remaining moves that was judged to be an optimal move. The results obtained so far were quite interesting: the induced player was able to beat a random opponent quite consistently, but failed to do so within 50 moves against an optimal opponent. The learned rules were rather complicated, because the provided background knowledge was at a fairly low level of abstraction (distances between pieces). All in all, the results had a striking similarity with results obtained in the area of *behavioral cloning* [43].

The approaches discussed so far induced concepts from simple endgame databases and suitable background knowledge. Databases of middle-game positions or entire games have so far only been used for the tuning of evaluation functions. These techniques are usually concerned with the adjustment of numerical weights and cannot be considered to be at the core of KDD research. The most notable approach is the work of [17], where statistical techniques and limited look-ahead were combined to tune the weights of their world championship program DEEP THOUGHT to yield optimal performance in a database of grandmaster moves. However, with the success of Tesauro's backgammon player, research in this area shifted gradually from tuning on databases to tuning by self-play via *temporal-difference learning* [47].

3 Why Chess for KDD?

Many of the datasets that have been extensively studied in inductive symbolic machine learning have become standard benchmark problems. For example, the KRK, KRKN and KRKPa7 datasets are part of the UCI collection of machine learning databases (along with datasets of other games, like Othello, Abalone, Connect-4, etc., and many "real-world" datasets). However, many researchers still consider applications of machine learning algorithms to game-playing domains as fairly trivial. While this may be true for some of the datasets in the UCI repository, we think that this is certainly not true for game playing domains in general. In fact, our opinion is that chess might yield a first-class test-bed for many ideas that have developed in KDD, as we attempt to illustrate in this section.

3.1 Efficiency

An important requirement for knowledge discovery techniques is that they scale up to large amounts of data. With the decline of hardware prices more and more data can be stored at low cost and it is the task of KDD algorithms to discover significant regularities in these huge databases. Thus an often heard question that is raised for new KDD techniques is how they scale up to databases with

significant sizes. Usually these techniques are evaluated on databases from the UCI repository of machine learning databases, where only a few datasets have more than 10,000 examples.

Chess databases, on the other hand, are available in all sizes. Typical game databases consist of several hundreds of thousands games, each consisting of, say, 30 positions on average, not counting variations and comments. Every commercial chess playing program has access to huge opening databases. All endgames with up to 5 pieces are available on three CD-Roms and certain types of 6-piece endgames are on the way. These endgames, however, already pose serious challenges to commonly available storage media [48]. A simple 3-piece endgame database already has about 64^3 entries (including some illegal positions), which are usually stored in a highly compressed format (for example by using board symmetries).

3.2 Background Knowledge

Most KDD techniques assume a data representation in the so-called *attribute-value* format. Encoding chess positions in this format would result in databases that have one entry per training position, where each entry encodes several important predicates that are true or not true in this particular training position. This requires that each concept from the background knowledge has to be represented as a new attribute of the dataset, which makes it tedious to provide additional background knowledge that might help to focus the learner on interesting concepts. Most available chess databases, like the KRKN and the KRKPa7 datasets from the UCI repository of machine learning databases, conform to this format.

However, research in the field of *Inductive Logic Programming (ILP)* [31, 7] has lead to the development of algorithms that are able to make use of background knowledge in full first-order horn-clause logic. Roughly speaking, these algorithms are concerned with the induction of PROLOG programs. The ability to use background knowledge in the form of PROLOG clauses allows systems such as PAL [28] to formulate rules with complex predicates in the background knowledge. The rules may even employ a limited look-ahead by including conditions like `make_move(Side,Piece,From,To,Pos,NewPos)` and looking for discriminating patterns in the new position that results from the specified move. We believe that due to its ability to incorporate relational background knowledge, inductive logic programming is a very promising line of research for knowledge discovery in chess domains, because chess concepts usually depend on spatial relations between pieces on the board and/or temporal relations between moves in a plan. This type of knowledge can be elegantly coded into a relational first-order logic representation.

3.3 Data Selection

An important step in the KDD process [11], that is widely neglected in KDD research, is the step of data selection, i.e., the creation of a training set from

the raw data, which is suitable for the selected data mining technique. While this step is in general domain-dependent, there might be some general principles that can be inferred from case studies. Interesting questions that have to be dealt with in the domain of chess are, e.g., "What constitutes a training examples, an entire game, the sequence of moves, the sequence of positions, or single positions?", "How to encode chess positions into feature vectors of fixed length, which are needed for most KDD algorithms?", or "How to deal with the temporal sequence that is inherent within move sequences in a chess game?".

Consider for example the case of learning the best pawn move in a certain pawn structure. One could try with a simple classifier learning approach. An important question, however, is what positions should be used for training the classifier? A simple approach would be to select all games with the pawn structure in question and use those positions as training examples, in which a pawn move is made thus destroying the structure. The different pawn moves that have occurred in the various games could be used as the dependent variable that has to be predicted by the classifier. However, it is quite likely that this approach would over-generalize in the sense that pawn moves would also be suggested for many positions where it would be premature. This could be prevented by adding all positions in which non-pawn moves have been tried as additional negative examples that constrain the learned concept. This approach, however, will treat all positions with a certain pawn structure as equal and will neglect the temporal sequence of positions in a game, which might contain important information. A systematic investigation of the various options in this or a similar task is certainly a promising endeavor.

3.4 Irrelevant Features

A common problem in KDD is that the data sets usually contain many irrelevant features. This problem is typically solved with techniques for feature subset selection [5, 18], i.e., techniques that filter out those attributes of the training set that appear to be irrelevant for the learning problem at hand. It is obvious, that in chess positions the exact location of several pieces is not always relevant, which means that chess databases might be a nice test-bed for such techniques. In particular, equivalents of feature subset selection for inductive logic programming algorithms, a research area which has not yet received the attention that it deserves [14], could be nicely studied in relational classification tasks in the domain of chess.

3.5 Noise

Another issue that is usually of considerable importance is noise in the data. *Noise* is a commonly used term for all sorts of errors and inconsistencies in the data. It may be the result of misclassifications, unavailable or erroneous information, contradicting examples, etc. The effect of noise are usually overly concept descriptions, because many learning algorithms have severe problems in discriminating between contradictory evidence that results from noise and can

thus be ignored and important exceptions to the current theory that have to be incorporated into the final theory. This problem is known as *overfitting*.

Phenomena of this kind also appear quite frequently in chess databases. In particular with respect to evaluation of a certain position or move, there is usually contradicting evidence from different commentators or from the various outcomes of games that continued from the given position. Imagine for example, the problem of learning the best move or sequence of moves in a certain middle-game pawn formation. There might be two contradicting plans that occur pre-dominantly, along with a variety of other variations that have been infrequently tried. Can we conclude that the one of the two frequently played plans that results in the higher proportion of wins is the better variation? If the evidence is backed up with a significant number of games, it certainly constitutes an interesting finding. What if a line has only been played twice, but has lead to a win in both cases? Would it be worth a try? Or is there the danger that there is a refutation that was not known to both players? A general problem with game databases is that they usually only contain the moves that have been played by two reasonably strong opponents, which means that certain variations that are well-known from theory and important to know will never or only rarely occur in such a database. This missing information also constitutes a severe problem for knowledge discovery techniques. In general, we think it is very hard to determine from the evidence in a game database whether a certain move that deviates from the pre-dominant plan in a certain position is a bad mistake or a significant innovation, a problem that is very similar to the problem of noise.

3.6 Unsupervised Learning

The study of algorithms for the inductive concept learning problem has a long tradition in Machine Learning and is probably the subfield that has received the most attention. This tradition has carried over to KDD, but there are a a variety of other approaches with different aims. For example, the discovery of association rules [1] or general dependencies [23, 39] might find interesting applications in chess databases for discovering typical piece-patterns, such as "In many cases when white castles queen-sides, he will sooner or later play h4.". Other techniques are able to discover temporal patterns [37] or interesting deviations from the norm [22]. For an excellent collection of papers on various KDD techniques consult [10]. There are also chess-specific KDD tasks that deserve a deeper investigation, such as the discovery of playing strategies from endgame databases, which we briefly discuss in section 4.2.

4 Why KDD for Chess?

A major point of motivation for KDD is that the discovered knowledge not only has to be interpretable and understandable, but also that it is novel, of interest for the user and maybe even has a commercial value. This section tries to illustrate why KDD approaches might be of interest to the chess community. In

particular, we will attempt to show that the potential of these techniques goes far beyond the induction of simple endgame classifier that have been predominant in the research so far (section 2).

4.1 Developing a High-level Chess Language

The first step that is needed for knowledge discovery in chess databases is the development of a suitable vocabulary of chess concepts in which the discovered knowledge can be formulated. Quinlan's work on the KRKN database has nicely illustrated the need for an appropriate vocabulary for learning (see section 2). However, such a knowledge representation formalism for chess concepts could also contribute significantly to computer chess in general. This has already been recognized in [52], where an advice-taking chess program is described. The aim of this project was the development of an abstract programming language that would allow a chess master to "advice" a playing program in terms of this language. Many formalisms have subsequently been developed in the same spirit [4, 16], most of them limited to certain endgames (see [27] for a bibliography).

The characteristics such a representation formalisms has to incorporate are that it has to be sufficiently expressive for formulating abstract strategic concepts, that it has to be extensible and can be easily understood by a user, and that it can be efficiently implemented. The last point is particularly important for KDD purposes, because there must be efficient ways for evaluating potential tests in discovered rules in order to allow efficient knowledge discovery in large-scale databases. Donninger [9] shows a promising step into the right direction by providing a very efficient interpreter of an extensible language for expressing certain characteristics of a board position. However, the expressiveness of the language is currently limited to propositional logic, a trade-off that had to be made because of efficiency considerations and the ability to provide a graphical interface that also allows untrained users to formulate rules.

4.2 Contributions to Chess Theory

Although chess theory is fairly well developed, there are many aspects of the game that still need to be explored. As an example of the potential of KDD approaches consider Ken Thompson's impressive work on five-men endgame databases, which is now publicly available on three CD-ROMs. His work has shown both, the enormous magnitude of the task and its importance for chess. Many of the analyzed endgames have provided new insights into chess theory. For example, it is now known that certain endgames cannot be be won within the 50 moves that are allowed by the rules of chess, or that other endgames which were believed to be draws in general positions can in fact be won (although, against best defence, usually not within 50 moves). As an example consider the KBBKN endgame, which was considered to be a draw for a long time, and was shown to be a win in at most 66 moves for all but some trivial cases [42].

On the other hand, many of these endgame databases are not thoroughly understood by human experts. The most famous example are the attempts

of grandmasters to defeat a perfect KQKR database within 50 moves or the attempt of an endgame specialist to defeat a perfect database in the "almost undocumented and very difficult" KBBKN endgame [42]. GM John Nunn's effort to manually extract some of the knowledge that is implicitly contained in these databases resulted in a series of widely acknowledged endgame books [32, 34, 35], but Nunn readily admitted that he does not yet understand all aspects of the databases he analyzed [33]. It would be rewarding to develop algorithms for automatically discovering playing strategies for such endgames (see [29] and [15] for some preliminary work). Such strategies could be both, easily implemented into a computer program as well as enrich the state-of-the-art of endgame theory. While for the former case an optimal strategy[1] would be desirable, for the latter case a suboptimal winning strategy is often preferable, if it is simple and understandable. This is a particularly hard problem, because it is non-trivial to decide which suboptimal moves contribute to some global progress and which suboptimal moves do not improve the position. An attempt to learn a simple playing strategy for the KRK endgame might easily end up with the simple strategy "always move your rook away from the enemy king" which will always result in a won position (at least for the next 49 moves), but clearly make no progress towards the goal of mating the opponent's king. Other tasks that could be automatized with KDD approaches include the discovery of opening theory mistakes, the automatic detection of particularly promising or unpromising line-ups or middle-game plans in certain types of openings, and many more.

4.3 Educational Chess Programs

Another obvious point, where chess knowledge would be of considerable importance, and probably the point with the highest commercial potential is the use of high-level chess knowledge in educational chess programs. For example, imagine a program that analyzes a certain position or an entire game on an abstract strategic level, tries to understand your opponent's and your own plans, and provides suggestions on different ways to proceed. Some commercial programs already provide such capabilities, but at a very preliminary level that usually is only able to detect tactical, but not strategic mistakes. The ICCA has recognized the potential of such programs, and has created the *The Best Annotation Award* which will be awarded annually for the best computer-generated annotation of a chess game.[2] Some preliminary work on using case-based reasoning for a strategic analysis of a given chess position can be found in [20, 21]. An automatic or semi-automatic facility for enriching the basic vocabulary of patterns and plans would be highly desirable for the development of such a system, or may even form an important component of such a program.

[1] In the following, an *optimal* playing strategy refers to one that will lead to mate in the minimum number of moves.

[2] See *ICCA Journal* 15(4):235–236, 1992.

4.4 Increasing Playing Strength

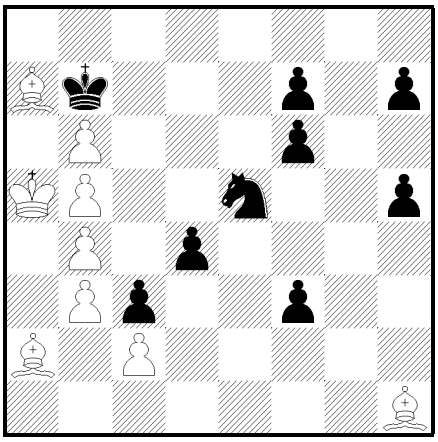
It should be obvious that incorporating additional knowledge into computer chess programs can lead to significant increases in playing strength. However, the motivation to investigate such approaches has significantly declined with the somewhat unexpected success of brute-force programs. Some authors have investigated approaches to incorporate strategic long-term knowledge into conventional chess programs [19, 36, 9], but the investigation of knowledge-based chess programs has practically come to a stand-still. However, we think that there are many situations where the myopia of brute-force chess programs surfaces.

As an extreme example, consider the problem shown in figure 1. It is quite safe to assume that this problem will not be solved by a computer program based on brute force search in the near future. We would even suppose that it is quite uncertain to assume that no program would correctly play this position. A brief experiment that we have conducted with FRITZ4 has demonstrated that FRITZ will play Bb1 to prevent black from playing Pd3, bring its king to a1, Ba2, king to e1, Bb1, and finally move the king to f2 in order to capture the pawn on f3. It does not realize that Pf6-f5 will draw after this capture and the exchange that follows it because of black's queening threats, which prevent all further activities of white.

However, despite the long solution sequence, the problem can be solved quite nicely by human beings. A typical path to a solution might proceed as follows: Because of the above drawing chance for black, white's only hope lies in queening one of its pawns on the b-file. To achieve this, white has to conquer square a6. However, he has no moves that put black into zugzwang, because black can answer all white king moves with king moves b7-a8-b7 or b7-c8-b7. However, a typical maneuver in such position is the so-called *triangle maneuver*, where one king is able to use a 3-cycle to return to its original square, while the other king is only able to make a 2-cycle. White can therefore try to move its king to e1, playing e1-f2-f1-e1, which would gain one move. When the white king returns to a5, we will have exactly the same position, but with black to move. As he cannot its king because of white's threat Ka6, he has to move pawn. Then the entire sequence is repeated 11 times until black has no more pawn moves and has to answer 254. Ka5 with Kc8 thus allowing Ka6, followed by a mate in 15.

While this position certainly is quite unlikely to occur in an actual game, it nicely illustrates the difference between a knowledge-based approach and a search-based approach to chess. It is quite obvious that the use of knowledge can be used to narrow down the search considerably. In fact, more than half of the moves need not be calculated at all, if one has found the 23-move maneuver for forcing black to move a pawn, which has to be repeated 11 times. This general pattern of repeating a certain move sequence to force the opponent to a weakening move will occur quite frequently in endgames. Likewise, the type of position discussed above, where FRITZ4 apparently tries its luck with capturing the black pawn on f3 only to reach a drawn endgame, is not uncommon in pawn endgames and is a nice example how explicit knowledge may considerably

Nenad Petrović, 1969



Mate in 270 moves

Figure 1: A hard problem for search-based chess programs

increase playing strength.

4.5 Tournament Preparation

Another possible application for KDD approaches would be to provide a sophisticated aid for unearthing characteristics of the style of individual players and for studying their weaknesses and strengths. Some commercial programs already support tools of this sort by, e.g., allowing to derive statistics on the success of a player with different openings or plotting statistics on the frequency of with which he moves a certain piece to a certain square. KDD techniques like the discovery of association rules or the detection of interesting deviations from the norm could provide much more powerful tools to that end. This type of application would lay its emphasis on the integration of suitable KDD techniques into a chess database interface instead of merely utilizing previously discovered knowledge.

5 Conclusion

In this work, we have studied the potential of chess as a test-bed for ideas in the wide field of Knowledge Discovery in Databases, and, conversely, to investigate the potential of KDD techniques for (computer) chess. To that end, we have started with an overview of previous work in that area, and have then outlined some ideas how important problems in KDD research can be studied in interesting learning problems in the domain of chess. Thereafter we have

sketched some ideas how the computer chess community and the chess community in general might profit from sophisticated applications of KDD techniques to chess databases. However, by no means we would like to give the impression that the application of KDD techniques to chess is a trivial task. It is our firm belief that such investigations will provide valuable results, but will require significant further research. The main goal of this paper was to motivate and encourage research projects in this area.

Acknowledgements

The author acknowledges support from a grant of the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)*. Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry of Science and Transport. Several aspects of this work have benefited from discussions with F.-G. Winkler, Luc De Raedt, and Chrilly Donninger. Thanks.

References

[1] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, 1995.

[2] Michael Bain. *Learning Logical Exceptions in Chess*. PhD thesis, Department of Statistics and Modelling Science, University of Strathclyde, Scotland, 1994.

[3] Michael Bain and Ashwin Srinivasan. Inductive logic programming with large-scale unstructured data. In K. Furukawa, D. Michie, and S. Muggleton, editors, *Machine Intelligence 14*, pages 233–267. Oxford University Press, 1995.

[4] Ivan Bratko and Donald Michie. A representation of pattern-knowledge in chess endgames. In M.R.B. Clarke, editor, *Advances in Computer Chess 2*, pages 31–54. Edinburgh University Press, 1980.

[5] Rich Caruana and Dayne Freitag. Greedy attribute selection. In W.W. Cohen and H. Hirsh, editors, *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 28–36, New Brunswick, NJ, 1994. Morgan Kaufmann.

[6] M. R. B. Clarke. A quantitative study of king and pawn against king. In M.R.B. Clarke, editor, *Advances in Computer Chess*, pages 108–118. Edinburgh University Press, 1977.

[7] Luc De Raedt, editor. *Advances in Inductive Logic Programming*, volume 32 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 1995.

[8] Luc De Raedt and Wim Van Laer. Inductive constraint logic. In *Proceedings of the 5th Workshop on Algorithmic Learning Theory (ALT-95)*. Springer-Verlag, 1995.

[9] Chrilly Donninger. CHE: A graphical language for expressing chess knowledge. *ICCA Journal*, 19(4):234–241, 1996.

[10] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, 1995.

[11] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37–54, Fall 1996.

[12] William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. Knowledge discovery in databases: An overview. *AI Magazine*, 13(3):57–70, Fall 1992.

[13] Johannes Fürnkranz. Machine learning in computer chess: The next generation. *ICCA Journal*, 19(3):147–160, September 1996.

[14] Johannes Fürnkranz. Dimensionality reduction in ILP: A call to arms. In L. De Raedt and S. Muggleton, editors, *Proceedings of the IJCAI-97 Workshop on Frontiers of Inductive Logic Programming*, pages 81–86, Nagoya, Japan, 1997.

[15] Johannes Fürnkranz and Luc De Raedt. Learning playing strategies from chess endgame databases: An ILP approach. Unpublished, Leuven, 1997.

[16] Micheal George and Jonathan Schaeffer. Chunking for experience. *ICCA Journal*, 13(3):123–132, 1990.

[17] Feng-hsiung Hsu, Thomas S. Anantharaman, Murray S. Campbell, and Andreas Nowatzyk. A grandmaster chess machine. *Scientific American*, 263(4):44–50, October 1990.

[18] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In W.W. Cohen and H. Hirsh, editors, *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 121–129, New Brunswick, NJ, 1994. Morgan Kaufmann.

[19] Hermann Kaindl. Positional long-range planning in computer chess. In M. R. B. Clarke, editor, *Advances in Computer Chess 3*, pages 145–167. Pergamon Press, 1982.

[20] Yaakov Kerner. Case-based evaluation in computer chess. In M. Keane, J.P. Haton, and M. Manago, editors, *Topics in Case-Based Reasoning (EWCBR-94)*, Lecture Notes in Artificial Intelligence, Berlin, 1994. Springer-Verlag.

[21] Yaakov Kerner. Learning strategies for explanation patterns: Basic game patterns with application to chess. In M. Veloso and A. Aamodt, editors, *Proceedings of the 1st International Conference on Case-Based Reasoning (ICCBR-95)*, volume 1010 of *Lecture Notes in Artificial Intelligence*, pages 491–500, Berlin, 1995. Springer-Verlag.

[22] Willi Klösgen. Efficient discovery of interesting statements in databases. *Journal of Intelligent Information Systems*, 4(1):53–69, 1995.

[23] Heikki Mannila and K.-J. Räihä. Algorithms for inferring functional dependencies from relations. *Data & Knowledge Engineering*, 12:83–99, 1994.

[24] Christopher J. Matheus. A constructive induction framework. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 474–475, 1989.

[25] Ryszard S. Michalski. On the quasi-minimal solution of the covering problem. In *Proceedings of the 5th International Symposium on Information Processing (FCIP-69)*, volume A3 (Switching Circuits), pages 125–128, Bled, Yugoslavia, 1969.

[26] Ryszard S. Michalski and P. Negri. An experiment on inductive learning in chess endgames. In Elcock and D. Michie, editors, *Machine Intelligence 8*, pages 175–192. Edinburgh University Press, 1977.

[27] Donald Michie and Ivan Bratko. Comments to 'chunking for experience'. *ICCA Journal*, 18(1):18, March 1991.

[28] Eduardo Morales. PAL: A pattern-based first-order inductive system. *Machine Learning*, 26(2-3):227–252, 1997. Special Issue on Inductive Logic Programming.

[29] Stephen Muggleton. Inductive acquisition of chess strategies. In J. E. Hayes, D. Michie, and J. Richards, editors, *Machine Intelligence 11*, chapter 17, pages 375–387. Clarendon Press, 1988.

[30] Stephen Muggleton. *Inductive Acquisition of Expert Knowledge*. Turing Institute Press. Addison-Wesley, 1990.

[31] Stephen H. Muggleton, editor. *Inductive Logic Programming*. Academic Press Ltd., London, 1992.

[32] John Nunn. *Secrets of Rook Endings*. Batsford, 1992.

[33] John Nunn. Extracting information from endgame databases. In H. J. van den Herik, I. S. Herschberg, and J. W. H. M. Uiterwijk, editors, *Advances in Computer Chess 7*, pages 19–34. University of Limburg, 1994.

[34] John Nunn. *Secrets of Pawnless Endings*. Batsford, 1994.

[35] John Nunn. *Secrets of Minor-Piece Endings*. Batsford, 1995.

[36] Andreas L. Opdahl and Björnar Tessem. Long-term planning in computer chess. In H. J. van den Herik, I. S. Herschberg, and J. W. H. M. Uiterwijk, editors, *Advances in Computer Chess 7*. University of Limburg, 1994.

[37] B. Padmanabhan and A. Tuzhilin. Pattern discovery in temporal databases: A temporal logic approach. In E. Simoudis, J. Han, and U.M. Fayyad, editors, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 351–354, Portland, OR, 1996. AAAI Press.

[38] A. Paterson. An attempt to use CLUSTER to synthesise humanly intelligible subproblems for the KPK chess endgame. Technical Report UIUCDCS-R-83-1156, University of Illinois, Urbana, IL, 1983.

[39] Bernhard Pfahringer and Stefan Kramer. Compression-based evaluation of partial determinations. In U.M. Fayyad and R. Uthurusamy, editors, *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pages 234–239, Montreal, Canada, 1995. AAAI Press.

[40] J. Ross Quinlan. Discovering rules by induction from large collections of examples. In D. Michie, editor, *Expert Systems in the Micro Electronic Age*, pages 168–201. Edinburgh University Press, 1979.

[41] J. Ross Quinlan. Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 463–482. Tioga, Palo Alto, 1983.

[42] A. J. Roycroft. Expert against oracle. In J. E. Hayes, D. Michie, and J. Richards, editors, *Machine Intelligence 11*, pages 347–373. Oxford University Press, Oxford, UK, 1988.

[43] Claude Sammut. Automatic construction of reactive control systems using symbolic machine learning. *Knowledge Engineering Review*, 11(1):27–42, 1996.

[44] Alen D. Shapiro. *Structured Induction in Expert Systems*. Turing Institute Press. Addison-Wesley, 1987.

[45] Alen D. Shapiro and Donald Michie. A self commenting facility for inductively synthesized endgame expertise. In D. F. Beal, editor, *Advances in Computer Chess 4*, pages 147–165. Pergamon, Oxford, 1986.

[46] Alen D. Shapiro and Tim Niblett. Automatic induction of classification rules for a chess endgame. In M. R. B. Clarke, editor, *Advances in Computer Chess 3*, pages 73–92. Pergamon, Oxford, 1982.

[47] Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, March 1995.

[48] Ken Thompson. 6-piece endgames. *ICCA Journal*, 19(4):215–226, 1996.

[49] T. F. Verhoef and J. H. Wesselius. Two-ply KRKN: Safely overtaking Quinlan. *ICCA Journal*, 10(4):181–190, 1987.

[50] Jean-Christophe Weill. How hard is the correct coding of an easy endgame. In H. J. van den Herik, I. S. Herschberg, and J. W. H. M. Uiterwijk, editors, *Advances in Computer Chess 7*, pages 163–176. University of Limburg, 1994.

[51] Franz-Günter Winkler and Johannes Fürnkranz. On effort in AI research: A description along two dimensions. In Robert Morris, editor, *Deep Blue Versus Kasparov: The Significance for Artificial Intelligence: Papers from the 1997 AAAI Workshop*, pages 56–62, Providence, RI, 1997. AAAI Press. Technical Report WS-97-04.

[52] Albert L. Zobrist and Frederic R. Carlson. An advice-taking chess computer. *Scientific American*, pages 93–105, June 1973.