Towards Automatic Recognition of Similar Chess Motifs

Miha Bizjak
University of Ljubljana
Faculty of Computer and Information Science
Ljubljana, Slovenia

Matej Guid
University of Ljubljana
Faculty of Computer and Information Science
Ljubljana, Slovenia

ABSTRACT

We present a novel method to find chess positions similar to a given query position from a collection of archived chess games. Our approach considers not only the static similarity due to the arrangement of the chess pieces, but also the dynamic similarity based on the recognition of chess motifs and dynamic, tactical aspects of position similarity. We use information retrieval techniques to enable efficient approximate searches, and implement textual encoding that captures the position, accessibility and connectivity between chess pieces, pawn structures, and moves that represent the solution to the problem. We have shown experimentally how important the inclusion of both static and dynamic features is for the successful detection of similar chess motifs. In another experiment the program was able to quickly traverse a large database of positions to identify similar chess tactical problems. A chess expert found the resulting program useful for automatically generating instructive examples for chess training.

KEYWORDS

problem solving, chess motifs, automatic similarity recognition

1 INTRODUCTION

A significant part of acquiring human skills is to identify our weaknesses and take measures to remedy them. In problem-solving domains such as chess, the analysis of past games is important for players trying to improve their game. Identifying their mistakes enables chess players to work on improving some aspects of their game. This is often done by training on similar problems. Finding relevant similar problems involves recognising both static patterns, i.e. finding similar chess positions, and dynamic patterns, i.e. finding similar move sequences that solve a problem. These static and dynamic patterns are often referred to as *chess motifs*. Learning and recognising chess motifs during the game is one of the main prerequisites for becoming a competent chess player [2].

Chess instructors often look for examples containing relevant chess motifs from real games to provide their students with useful teaching material. However, it is impossible for a human being to go through thousands or even millions of games and find problem positions with similar chess motifs and similar solutions to those overlooked by the students in their game. Finding contextually similar chess positions could also be used for annotating chess games [5] and in intelligent chess tutoring systems [10].

The goal of our research is to develop a method to automatically retrieve chess positions with similar chess motifs for a given query position from a collection of archived chess games.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Information Society 2020, 5–9 October, 2020, Ljubljana, Slovenia
© 2020 Copyright held by the owner/author(s).

1.1 Related Work

Existing chess search systems equipped with a query-by-example (QBE) [11] search interface are limited to searching only the exact matches in response to a given query position. To alleviate the problem of exact position searches, the Chess Query Language system (CQL) [1] allows the search for approximate matches of positions. However, it requires the user to define complex queries in the system-specific language. The search results can be sorted by any user-defined feature. In addition, the CQL works directly on game files and checks each game sequentially, making it inefficient for querying larger databases.

To overcome these problems, an approach has been proposed which is based on information retrieval for obtaining similar chess positions [4], constructing a textual representation for each board position and using information retrieval methods to calculate the similarity between these documents. Instead of constructing a query manually, the user specifies a chess position and a query encoding the characteristics of the position is automatically generated internally. Initially, a naive encoding was used, which only contains the positions of the individual pieces. The results have been improved by including additional information about the mobility of the individual pieces and the structural relationships between the pieces. Further work has been carried out to improve the quality of retrieval by implementing automatic recognition of pawn structures [7]. The additional information provided by the application of domain knowledge has proved useful, however, the positions are still only statically evaluated.

All existing approaches have a common shortcoming: they only allow the search for statically similar positions, while ignoring the dynamic factors, which are often far more important to obtain relevant search results.

2 DOMAIN DESCRIPTION

In this paper, we will focus on automatic retrieval of similar chess tactical problems from a large database of chess games. In chess, the term *tactic* is used to describe a sequence of moves that takes advantage of a certain position on the board and allows the player to gain material, a positional advantage, or even leads to a forced checkmate sequence.

Chess tactical problems are particularly important for the progress of chess players. Knowledge of tactical motifs helps them to quickly recognise the possible presence of a winning or drawing combination in a position. Chess players improve their tactical skills by solving tactical problems. A large number of games are decided by tactics, since a single mistake, which gives the opponent an opportunity for tactics can change the outcome of a game. To help players to discover tactical possibilities in games, many common patterns or *tactical motifs* have been defined in the chess literature [6]. Stoiljkovikj et al. developed a method for estimating the difficulty of chess tactical problems [9]. They introduced a concept of *meaningful search trees*, which can

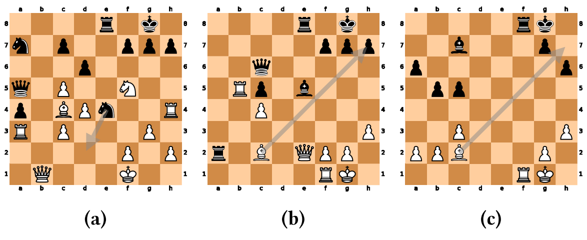


Figure 1: Tactical motifs.

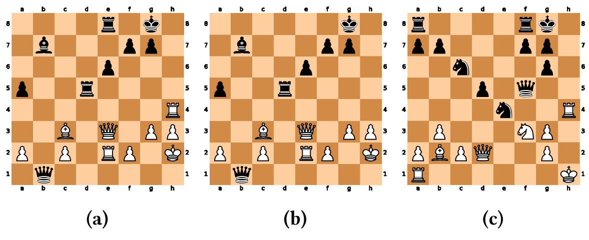


Figure 2: Static and dynamic similarity.

potentially be used either for motif recognition or as an additional feature for positional similarity ranking.

We use standard chess annotation. Chess games are stored using Portable Game Notation (PGN), chess positions are described with Forsyth-Edwards Notation (FEN), and chess moves are described with Standard algebraic notation (SAN) [3].

Figure 1 shows some of the more common motifs. In Figure 1a, Black performs a *double attack* on the white king and queen at the same time. White must move the king out of check, allowing Black to capture the queen. Figure 1b is an example of a *discovered attack*. By moving the bishop, White opens the queen's line of attack on the rook on a2. After Black responds to move out of the check, White can capture the black rook. The tactic in Figure 1c is called *deflection*. The black king protects the rook on f8. White gives a check with the bishop, forcing the black king to move away from the rook so that it can be captured.

To illustrate the difference between static and dynamic similarity using an example, we compare the query position in Figure 2a with the positions in Figure 2b and Figure 2c. The position in Figure 2b seems to be very similar to that in Figure 2a: only the white rook on h4 and the black rook on e8 have been removed. These two positions are statically similar. On the other hand, the position in Figure 2c seems to be quite different. However, if we compare the move sequences that represent solutions to these two tactical problems, we notice a great dynamic similarity. The solution in Figure 2a is 1. Rh8+ Kxh8 2. Qh6+ Kg8 3. Qxg7#. The solution in Figure 2c contains the same tactical motif as the solution mentioned above: the white rook is sacrificed on h8 and the black king must capture it, allowing the white queen to appear with check on h6 (note that it cannot be captured due to the activity of the white bishop along the long diagonal) and deliver checkmate on the next move. Note that such motif is not possible in the position shown in Figure 2b.

We are particularly interested in recognising the dynamic similarity, i.e. finding positions with similar motif(s) in the solution of the problem. However, we also want to take into account the static similarity, i.e. finding problems with similar initial position.

3 SIMILARITY COMPUTATION

To determine similarity between tactical problems we use an approach based on information retrieval. A set of features is computed from each problem's starting position and its solution move sequence. The features are then converted into textual terms, forming a document that represents the problem. A collection of documents is used to build an index, which can then be queried using the textual encoding of a new position to retrieve the most similar positions in the index. For the implementation of the system for indexing and retrieval of similar tactics we use the *Apache Lucene Core* library. Search results are ranked using the BM25 ranking function [8].

For each tactic, the input consists of a starting position in FEN format and a solution move sequence in algebraic notation. The solution can be provided with the position or calculated using a chess engine. Sections 3.1 and 3.2 describe the features and terms that are generated, and Figure 3 shows an example of a text encoding.

3.1 Static Features

The static part of the encoding includes information about the positions of pieces on the board, structural relationships between pieces and pawn structures present in the position.

The implementation is based on previous work on similar position retrieval [4] and pawn structure detection [7] and is intended to serve as a baseline on which we aim to improve by implementing encoding of dynamic features.

3.1.1 Piece positions and connectivity. The section describing piece positions and connectivity encoding consists of three parts:

- *naive encoding* - the positions of all the pieces on the board.
- *reachable squares* - all squares reachable by pieces on the board in one move, with decreasing weight based on distance from the original position, in format *{piece symbol and position}|{weight}*.
- *connectivity between the pieces* - the structural relationships between the pieces in the positions. For each piece it is recorded which other pieces it attacks, defends or attacks through another piece (*X-ray attack*). Attacks are encoded as *{attacking piece symbol}>{attacked piece symbol and position}*. For defense and X-ray attack terms, < and = separators are used instead.

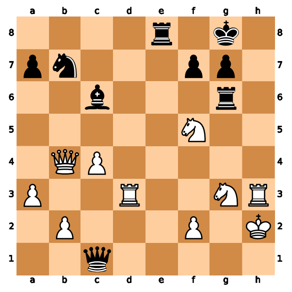
3.1.2 Pawn structures. For this section of the encoding, we use pawn structure detection algorithms [7] to detect the following pawn structures in the position and encode them into terms: isolated pawns (*I{pawn position}*), (protected) passed pawns (*F{pawn position}*), backward pawns, doubled pawns and pawn chains. Terms *P({number})* and *p({number})* are used to encode the number of pawn islands for white and black, respectively.

3.2 Dynamic Features

In the dynamic part of the encoding, we focus more on the solution of the tactical problem, trying to capture the motif behind it. We first encode some general characteristics of the solution, then add more specific terms describing the move sequence.

3.2.1 General dynamic features. In this part we encode some basic features of the solution move sequence that can help us determine similarity. We use a single term for each of the following features if it holds for the solution:

- *?px* - the player captures a piece in at least one of the moves



(a) Encoded position. Black to play, solution: 1... Qh1+ 2. Nxh1 Rg2#.

Feature set	Generated terms
static_positions	qc1 Pb2 Pf2 Kh2 Pa3 Rd3 Ng3 Rh3 Qb4 ...
	qa1|0.78 qb1|0.89 qd1|0.89 qe1|0.78 ...
	q>Pb2 q>Pc4 Q>nb7 N>pg7 r>Ng3
	P<Pa3 P<Ng3 K<Ng3 K<Rh3 P<Qb4 ...
	q=Pa3
static_pawns	If2 ia7 Fc4 P(2) p(2)
dynamic_general	?ox ?+ ?# ?S
dynamic_solution	!-q !-N !-r !-qN !-Nr !xq
	!Sq
	!#b !#r !#br
	!K>q !N>q !q>K !b>N !K>r !r>K !r>P

(b) Text encoding of each set of features for the above position.

Figure 3: Text encoding of a tactical position.

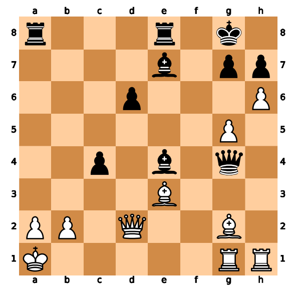
- *?ox* - the opponent captures a piece in at least one of the moves
- *?+* - the player gives a check at least once during the sequence
- *?=* - the player promotes a pawn in at least one of the moves
- *?S* - the player sacrifices one or more pieces
- *?#* - the solution ends with a checkmate
- *?1/2* - the solution ends in a draw

3.2.2 *Solution sequence features.* In this section we encode information about the solution move sequence. The encoding includes a term for each:

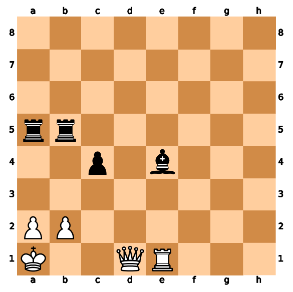
- type of piece moved: *!-{piece symbol}*
- type of piece captured: *!x{piece symbol}*
- attack between pieces that occurs during the solution: *!{attacking piece symbol}>{attacked piece symbol}*
- type of piece sacrificed: *!S{piece symbol}*
- (if the final position is a checkmate) type of piece involved in checkmate: *!#{piece symbol}*

We count a piece as involved in checkmate if it is attacking either the king directly or any of the squares where the king could move from the current position (ignoring checks).

To include information about the order of moves and captures we also include a term for each two consecutive moves and captures in the solution. We also include a term for each pair of pieces involved in checkmate to capture more specific combinations of pieces.



(a) Base problem. Black to play, solution: 1... Rxa2+ 2. Kxa2 Ra8+ 3. Ba7 Rxa7+ 4. Qa5 Rxa5#.



(b) Simplified problem. Black to play, solution: 1... Rxa2+ 2. Kxa2 Ra5+ 3. Qa4 Rxa4#.

Figure 4: A pair of tactical problems from the data set.

4 EXPERIMENTAL RESULTS

To evaluate the effectiveness of our methods, we used a number of problems that we have collected from the Chess Tactics Art (CT-ART 6.0) training course[1]. Many puzzles in this course consist of pairs of positions: one is taken from a real game, another represents a simplified version where the same tactical motif usually appears on a smaller 5×5 board. This fact allowed us to obtain a set of position pairs that were considered similar by human experts. We manually checked the puzzles and verified the similarity between the solutions of the individual problem pairs. A total of 400 pairs were collected for the test data set.

An example of such a pair is shown in Figure 4. The solution to both problems is to sacrifice the rook on the a-file to expose the king, resulting in checkmate with the other rook and the bishop on e4. The solution in the simplified problem contains the same motif, but there are much fewer pieces, so the solution is generally easier for the students to find.

4.1 Evaluation of Similarity Detection

We tested the effectiveness of our methods using the set of 400 pairs of problems described in the previous section. We first built an index using the simplified version of the problem from each pair, then performed a query on the index with each of the regular problems. For each query we recorded the rank of the matching position in the results and calculated how often the matching position appeared as the top result or within the first N results.

We tested the search accuracy using the following feature subsets: each feature group on its own, all static features, all dynamic features and all features combined. All runs used the default BM25 parameters $k_1 = 1.2$ and $b = 0.75$ and all included feature sets were weighted equally. The results are presented in Table 1.

Using either only static or dynamic features did not yield the best results. The results were significantly improved when both static and dynamic features were combined. This shows that each set of features covers a different aspect of a tactic, both of which need to be considered when determining similarity.

4.2 Similar Position Retrieval

In the second experiment, we selected 10 contextually different chess tactical problems and then automatically retrieved 5 most similar positions for each of them from a large database of 278,840

[1]https://chesskingtraining.com/ct-art

Feature set used	Accuracy		
	top-1	top-5	top-10
static_positions	0.234	0.378	0.428
static_pawns	0.033	0.083	0.126
dynamic_general	0.008	0.038	0.071
dynamic_solution	0.421	0.657	0.761
all static features	0.252	0.370	0.433
all dynamic features	0.418	0.652	0.761
all features, equal weights	**0.481**	**0.736**	**0.814**

Table 1: Success rates for different configurations.

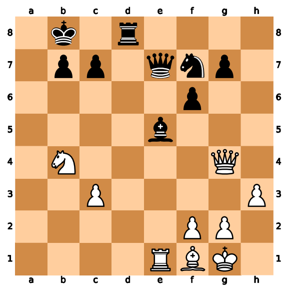
tactical problems constructed from the lichess.org game database. Building the index took about 14 minutes (it only needs to be done once), and retrieval was fast: only about 4 seconds.

Figure 5 shows a query position and the first two of the five most similar retrieved positions. This example illustrates how similarity ranking works and how the static and dynamic features contribute to the similarity scores of the results. The query position is an example of a discovered attack motif. With 1... Bh2+, Black sacrifices the bishop to later capture the rook on e1 with the queen. The first result shows the same motif with an almost identical move sequence. The main difference is that the key pieces are on the d-file and not on the e-file. The second result is another case of a discovered attack. In this example it is not a bishop but a knight sacrificed with a check to the white king. It is the static similarity (the arrangement and position of the pieces in the initial position) that contributes most to the great overall similarity of this tactical problem, although a certain dynamic similarity was also detected.

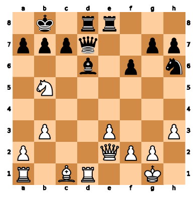
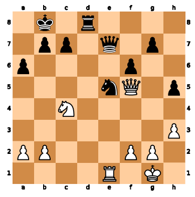
The resulting most similar positions were shown to a chess expert. The expert was asked to comment on the reasons for the similarity of the resulting problems with the original query positions, taking into account both static and dynamic aspects. The expert was able to explain the similarity in 48 out of 50 problems. Overall, the expert praised the program's ability to detect dynamic similarity of positions, even if the initial positions differ significantly.

5 CONCLUSIONS

We introduced a novel method for retrieving similar chess positions, which takes into account not only static similarity due to the arrangement of the chess pieces, but also dynamic similarity based on the recognition of chess motifs and dynamic, tactical aspects of position similarity. The merits of the method were put to the test in two experiments. The first experiment emphasized the importance of including both static and dynamic features for the successful detection of similar chess motifs. In the second experiment, the program was able to quickly traverse a large database of positions to identify similar chess tactical problems. A chess expert was able to explain the similarity in the vast majority of the retrieved problems and praised the program's ability to detect dynamic similarity of positions even if the initial positions differ significantly. The resulting program can be useful for the automatic generation of instructive examples for chess training.



(a) Query position. Black to play, solution: 1... Bh2+ 2. Kxh2 Qxe1.

Position	Solution	Similarity score	
	1... Bh2+ 2. Kxh2 Qxd1	static dynamic **total**	38.95 45.04 **83.99**
	1... Nf3+ 2. Qxf3 Qxe1+	static dynamic **total**	64.62 12.32 **76.94**

(b) Retrieval results.

Figure 5: Example of retrieval results.

REFERENCES

[1] G Costeff. 2004. The Chess Query Language: CQL. *ICGA Journal*, 27, 4, 217–225.

[2] Mark Dvoretsky and Artur Yusupov. 2006. *Secrets of Chess Training*. Edition Olms.

[3] International Chess Federation (FIDE). 2020. The FIDE Handbook. https://handbook.fide.com/. (2020).

[4] Debasis Ganguly, Johannes Leveling, and Gareth JF Jones. 2014. Retrieval of similar chess positions. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 687–696.

[5] Matej Guid, Martin Možina, Jana Krivec, Aleksander Sadikov, and Ivan Bratko. 2008. Learning positional features for annotating chess games: A case study. In *International Conference on Computers and Games*. Springer, 192–204.

[6] Chess Informant. 2014. *Encyclopedia of Chess Combinations, 5th Edition*. Chess Informant.

[7] Matic Plut. 2018. Recognition of positional motifs in chess positions. Diploma thesis. University of Ljubljana.

[8] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109, 109.

[9] Simon Stoiljkovikj, Ivan Bratko, and Matej Guid. 2015. A computational model for estimating the difficulty of chess problems. In *The Annual Third Conference on Advances in Cognitive Systems*.

[10] Beverly Park Woolf. 2010. *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann.

[11] Moshé M Zloof. 1975. Query-by-example: the invocation and definition of tables and forms. In *Proceedings of the 1st International Conference on Very Large Data Bases*, 1–24.