Anthropomorphic Chess Evaluation Via Qualitative Analysis

Luke Salamone

Northwestern University McCormick School of Engineering, Evanston, Illinois
Submitted 8 June 2021

Introduction

Chess is a two-player, perfect information, zero-sum game. Chess-playing computer programs, otherwise known as chess engines, have existed since at least the late 1940s (UoMCompsci, 2012). Because it was said to require the perfect combination of planning, strategy, psychology, and calculation, chess was once thought to be an activity directly correlated with intelligence, and that a truly intelligent computer should be able to defeat humans. Although chess engines have long since surpassed the capabilities of human grandmasters, several opportunities still exist for an engine that reasons in a more human-like way.

This paper contains two related proposals. The first is a chess engine tournament, unique in the type of engine which will be permitted to enter and likely to succeed. Importantly, the vast majority of engines currently holding the highest performance ratings will likely not be effective.

The second proposal is the outlines of a chess engine that is likely to be successful in this tournament, taking advantage of the highly qualitative nature of chess position evaluation. Although it is unlikely to perform as strongly against top-performing engines, there are several distinct advantages of such an engine which are outlined in **§3A**. In short, there is likely to be a great deal of educational value as well as financial incentive driving the construction of highly successful qualitative chess engines.

§1: Background

The general algorithm for performing search in zero-sum games like chess is known as minimax. While the details of minimax can be found outside of this paper, the algorithm generally functions by attempting to find the optimal move under the assumption that an opponent will also play the optimal move. Minimax is essentially a depth-first search of the game tree, with leaf nodes assigned a value from a static evaluation function.

Minimax is often augmented with “alpha-beta pruning” to reduce the number of positions which will be evaluated. This effectively cuts the computational complexity exponent in half by removing from consideration those branches which cannot affect the final result (Russell & Norvig, 2010, 167).

There are several challenges to the minimax function which require attention. The primary and most fundamental challenge to minimax is the sheer size of the game tree, containing an immense number of possible positions overall. This makes an exhaustive search impossible. Consequently, many of the nodes evaluated during minimax will not be terminal (end of game) nodes, and will need to be assessed using heuristics. Because most nodes are evaluated using the static evaluation function, the overall performance of minimax is highly dependent on the performance of the static evaluation function.

Other challenges in minimax evaluation exist as well. In particular, a phenomenon dubbed the "Horizon Effect" is a peculiar failure mode of minimax searches. The Horizon Effect, first described by Berliner (1975, I). His illustration of the problem (his Figure 1.3) is reproduced in **Figure 1**. Algorithms that do not account for the Horizon Effect will try to "push" bad outcomes of their search beyond their search horizon, instead opting to make hopeless moves which only serve to delay the inevitable. Notably, because human players are not limited by search depth, they are less susceptible. The Horizon Effect and its implications will be discussed later on.

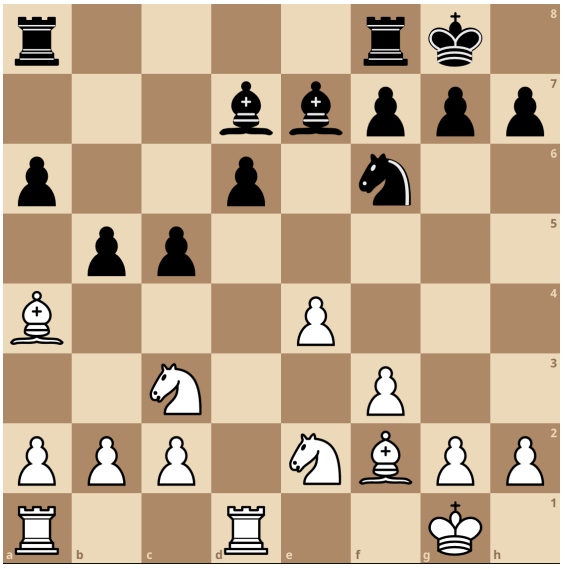

Figure 1. White to move. Here, white's bishop on a4 is doomed, attacked by black's pawn on b5. White could delay the inevitable by moving his bishop to b3, but then black simply seals the bishop's fate with pawn to c4. In that position, white does not have time to save his bishop, and it will be captured no matter what on the next move by the pawn on c4. Due to the Horizon Effect, at a limited depth white will not recognize this and will play hopeless moves like pawn from e4 to e5, temporarily attacking the knight but easily parried by capturing with the pawn on d6. This phenomenon is deemed the "Horizon Effect" because by pushing negative outcomes beyond the "horizon" of calculation depth, the engine is able to trick itself into believing that the problem doesn't exist. A true case of "see no evil".

§ 1A: Early Chess Engines

An exhaustive history of chess engines is beyond the scope of this document, but chess engines have generally coevolved with the computers they run on. Because early engines had far less computing power to lean on than modern computers, many of the early pioneers were quite enthusiastic about "selective search" strategies.

In any given position, there are many possible moves. By some estimates, chess has an average branching factor of around 30, with other estimates

putting the number around 40[1]. In any case, it is possible to dramatically reduce this number by employing “selective search”, which means excluding nodes from recursive tree search altogether. Reducing the number of possible moves that will be explored in any given position promises dramatic computational speedups, allowing the tree to be searched deeper at the expense of width. Given the slow speeds (by contemporary standards) of early computers, the allure of such a technique should be clear.

An early attempt to narrow the search tree was proposed by Hans Berliner, who in 1975 devised CAPS-II which utilized a tree search algorithm with five total “reference levels” including ALPHA and BETA (IV-3). His paper also was cognizant of the work of Chase and Simon from two years prior, recognizing the need for a bottom-up board representation (II-1). Unfortunately, his board representation was largely geometrical and included little in the way of qualitative relationships between pieces (II-6). Nevertheless, the resulting program was able to solve tactical puzzles in a quite impressive manner (V-10).

Another example of the use of such a narrowed search tree is PARADISE (PAttern Recognition Applied to DIrecting SEarch). This system used a knowledge base containing 200 “production rules” to match board patterns and determine the best move at any time. An example production rule is shown in **Figure 2**.

[1] It isn’t easy to find a concrete value for the branching factor of chess. One source claims, without citation, that the branching factor may be up to 40 (Winston, 1992, 117). However, a more recent statistical analysis of 2.5 million chess games put the real number closer to 31 (Barnes, 2019). This branching factor depends on the stage of the game; middle games have far more available moves than end games. Because not all games are the same length, shorter games will tend to have higher average branching factors than longer ones. Barnes also includes a quite interesting graphic which readers are encouraged to view at the citation URL.

```
((DMP1)
  (NEVER
    (EXISTS (SQ)
      (PATTERN MOBIL DMP1 SQ)))
  (NEVER
    (EXISTS (P1)
    (PATTERN ENPRISP1 DMP1)))
  (ACTION ATTACK
    ((OTHER-COLOR DMP1)
      (LOCATION DMP1))
    (THREAT
    (WIN DMP1))
  (LIKELY 0)))
```

Figure 2. A sample production rule from the PARADISE knowledge base. This production rule detects and acts upon the opponent’s trapped pieces. A trapped piece is identified as a non-pawn defensive piece which cannot move to another square without being captured and is not already attacked. Finally, the production rule describes the threat of this action (winning the piece) and how likely it is to succeed.

Because the tree search was narrower, the system was able to search to higher depths. Still, because of hardware limitations of the time, PARADISE was extremely slow, generating only 109 nodes in 20 minutes of search time (Wilkins, 1980, 167).

PARADISE executes static analysis on a position using its many production rules with the ultimate goal of creating and executing a plan. Matched production rules post concepts to the database in addition to information about the intentions of the concept and its likelihood of success (Wilkins, 1980, 172-173).

The program is then able to use this information to form a plan, which is a set of actions for the side to move, along with corresponding plans for each defensive alternative. Because there may be many potential alternatives at each move, this plan is necessarily non-linear, containing many branches instead.

```
(((WN N5)
  (((BN N4) (SAFEMOVE WR Q7)
    (((BK NIL) (SAFECAPTURE WR BR))
      ((ANYBUT BK) (SAFECAPTURE WR BK))))
    ((BN N4)
      (CHECKMOVEWR Q7)
      (BK NIL)
      (SAFECAPTURE WR BQ))))
  ((THREAT (PLUS
    (EXCHVAL WN N5)
    (FORK WR BK BR)))
  (LIKELY 0))
  ((THREAT (PLUS
    (EXCHVAL WN N5)
    (EXCHWR BQ)))
  (LIKELY 0)))
```

Figure 3. White to move, PARADISE has produced a plan which involves checking the black king by moving the knight to g5, then checking the black king by moving the rook to d7. Depending on black's next move white will then try to either capture the queen on d4 or the rook on d7.

An example of such a plan is reproduced in **Figure 3** above (Wilkins, 1980, 174-175)[2].

Searching only selective lines is more difficult to implement than full-width search. Further, selective searches may miss important continuations of a position, causing the computer to select an incorrect move (Frey & Atkin, 1979, 126-128). It was for this reason that Berliner himself, an original proponent for the application of strict logical rules in chess, decided to seek brute-force search methods instead (McClain, 2017)[3].

Previous work largely focused on creating the most powerful chess engine on the hardware which was available. With the singular goal of defeating human players, sacrificing explainability for the sake of computational expediency was a logical and understandable tradeoff to make. In the present day, however, there is no longer a question of whether humans or computers are superior chess players. It seems quite clear that the time has come to revisit some of the tradeoffs made in the past.

§1B: Modern Chess Engines

Stockfish Classic is currently the #1 ranked chess engine in the world in the blitz chess category (Chess.com, 2020). Stockfish is an open-source chess engine which performs a "full width" search on the game tree. Leaf nodes are evaluated using either a classical evaluation function, or more recently, a neural network evaluator called NNUE. The classical evaluation function uses a set of around 30 factors weighted empirically using a dedicated testing framework called Fishtest. Altogether, the project has around 200 contributors (Stockfish, 2021). Optimized for speed, Stockfish typically evaluates around 5 million nodes per second on a typical 4-core computer (Chessify, 2021).

[2] An interesting sidebar about this position is that based upon Stockfish analysis, this position results in inescapable checkmate for black. That is, even with optimal play, black will be checkmated in 19 or fewer moves. The winning sequence begins with white moving his queen to e5, exploiting the pin on the f6 pawn from the f1 rook.

[3] In some literature this brute-force full-width calculation strategy is known as a "Shannon type-A" algorithm (Frey & Atkin, 1979, 126; Slate & Atkin, 1983, 113). This is in contrast with a Shannon type-B algorithm, which uses a pruned evaluation tree.

Neural networks may also be used more directly. Leela Chess Zero, also known as Lc0 is another open-source chess engine, modeled after DeepMind's AlphaZero chess engine. Lc0 uses Predictor + Upper Confidence Bound tree search (PUCT) to search its game tree. New nodes are evaluated by iteratively choosing moves from a probability distribution until an unexplored node has been reached, at which point the node value will be estimated by a neural network and propagated back up the tree. PUCT is very similar to Monte Carlo Tree Search, but with game "rollouts" replaced by neural network evaluations (Leela Chess Zero, 2020).

One other notable engine is Maia Chess. Maia is notable in that its objective is to model human behavior rather than to perform optimally. This is interesting because Maia is in many cases trained to perform suboptimally. The other notable feature about Maia is the unusual manner in which moves are selected: instead of performing any type of tree search at all, the engine simply returns the inference from a single neural network using the board position as input (McIlroy-Young et al., 2020, 6).

§2: Qualitative Analysis

Qualitatively, there are many aspects to a chess game that may be captured. Rather than enumerating them here, it seems far more valuable to use this space making note of the way that a human grandmaster analyzes a position. It will become quite apparent that at the highest levels, the qualitative aspects of position analysis dominate over quantitative aspects (i.e. the number and value of each piece).

In the selected lecture, Grandmaster Varuzhan Akobian details a game he played previously. At a key moment of the game, Akobian sacrificed his rook for a key pawn in the center of the board. The resulting position is reproduced for convenience in **Figure 4**.

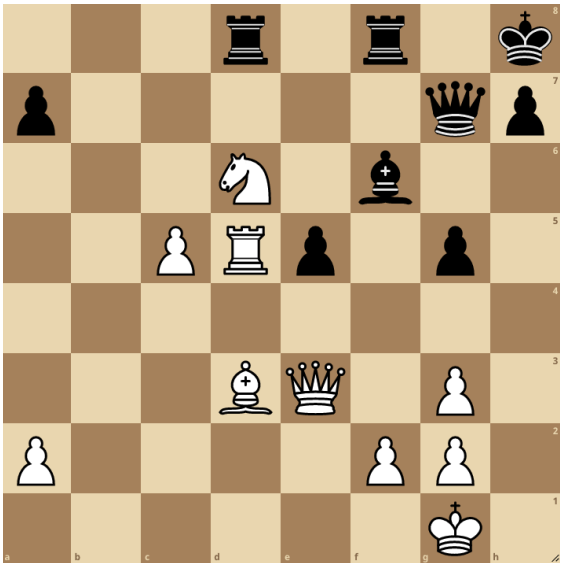

Figure 4. Quantitative analysis would posit that black is winning due to his extra rook for white's knight and pawn. However, qualitative analysis provides a more complete picture of black's predicament.

His analysis starts at 25:15 in the video (Saint Louis Chess Club, 2013):

> I would like you to spend a minute or two just to give me the evaluation of this position. It may not seem that clear because I'm down the exchange[4]. I have a knight and a pawn for a rook. Rook is valued 5, knight and a pawn is 4. It may seem like I'm down a pawn here. But what do you think is the proper evaluation of this position?
>
> ...Basically white is very active. There are a few other things we can mention about white's position, that it's very strong. What else is very strong? White's king is very safe, he cannot

[4] Novice chess players are taught that chess pieces have quantitative values, which may come into consideration when exchanging one piece for another. These values are measured in terms of pawns. Knights and bishops are generally understood to be worth 3 pawns, rooks are worth 5, and queens are worth 9.

> attack me. But how about the black king? Do you think the black king is very safe? [No.] For example, I could put my queen here [the e4 square] then I have a battery! Remember when we have a queen and a bishop on the same diagonal we call that a battery. And suddenly if I can deflect this queen [black queen on the g7 square] I will just go queen takes pawn, checkmate!
>
> His dark square bishop is basically trapped behind his own pawns so it's ineffective. . . . My bishop is very active. . . . And one more thing that you can mention. Passed pawn, exactly! And it's a very strong passed pawn because with a knight on d6 very quickly [the pawn] will turn into [a queen].
>
> . . . How much advantage does white have here? Big advantage, slight advantage, maybe winning? . . . We're not going to use Houdini [a chess engine], Houdini will probably say black is slightly worse. But in practical play, I would be very comfortable to play this against anybody, and pretty comfortable I can win this position for white.

Note that quantitative analysis is almost entirely absent from GM Akobian's evaluation. Towards the beginning he mentions that he has sacrificed his rook for a knight and a pawn, and consequently is at a material deficit. However, he quickly discards this shallow evaluation, going so far as to label his subsequent qualitative evaluation as the "proper" evaluation.

GM Akobian goes on to mention several other qualitative features of the position which are difficult to assign quantitative value to. Firstly, the activity of his pieces means that it is much easier to play the position as white because his pieces are on better squares, including some deep in black's half of the board. The lack of activity is mentioned later on, noting that black's bishop is essentially trapped behind his own pieces.

King safety is another difficult thing to quantify. In the given position, it is difficult to find a way that black can even check the white king. Moving the d8 rook to b1 will take 2 moves, and even then the b1 square is guarded by the bishop on d3. So the white king is indeed quite safe. In contrast black king is quite vulnerable, guarded mainly by the black queen, who is herself vulnerable to deflection[5] or direct attack.

GM Akobian emphasizes the weakness of black's king by sketching out a simple game-winning checkmate plan: arrange the bishop and queen in a battery which attacks the h7 pawn, deflect the black queen, and deliver checkmate with the queen by taking the h7 pawn. Although it is not immediately clear how to implement the plan, this type of simple plan creates a well-defined long-term "threat" that black must contend with.

Another threat he mentions is encompassed by white's passed pawn[6] on c5. This pawn may become a queen, which would become an insurmountable advantage for white. Therefore, this threat is another long-term vulnerability for black.

Finally, note that GM Akobian does not assign a quantitative value to the board position, but rather a "very comfortable to win" assessment. Very little of

---

[5] Deflection is a chess tactic which involves "distracting" an opponent's piece which plays an important defensive role. For example, a piece which is defending two pieces simultaneously may be deflected by capturing one of the defended pieces.

[6] A passed pawn is a pawn which cannot be stopped or attacked by an opponent's pawns. This occurs when there are no opponent pawns in the "file" (vertical column) of the pawn, as well as the file to the left and the right, if applicable. For example, a pawn in the C file is a passed pawn if there are no opponent pawns in the B, C or D files. A pawn in the H file is passed if there are no opponent pawns in the G or H files.

this analysis involves quantities, but rather qualitative situations which must be dealt with. Consequently, it seems that qualitative reasoning is an ideal tool which a chess engine might use.

The nature of expert-level perception experienced by GM Akobian was studied directly in a 1973 paper by Chase and Simon. Participants of three different levels of chess ability (a master-level player, an experienced class A player, and a novice) were asked to complete two chess-related cognitive tasks. The first was a perception task, requiring him to reproduce a chess position on an adjacent board as quickly as possible, with the model board in plain view. The second task was a memory task, requiring participants to reproduce a position from memory after viewing it for only 5 seconds (page 58).

Importantly, the perceptual study attended to chess players' tendencies to "chunk" the board position as they reproduced positions, tending to remember groups of interrelated pieces. These pieces tended to have relationships which the authors characterized in five ways: a piece attacks another, a piece defends another, two pieces are adjacent, two pieces are the same color, two pieces are the same type (page 68).

The results of the study found that "the C, S, and null relations are low because subjects are placing pieces which usually have multiple relations. Thus, from the within-glance relations, it appears that subjects are noticing the pawn structure, clusters of pieces of the same color, and attack and defense relations over small spatial distances." (page 68). In other words, it seems likely that human players are quite attentive to certain types of qualitative relationships between pieces.

§3: A Qualitative Chess Engine

It is unlikely that a qualitative chess engine will be able to entirely do away with the basic structural algorithm involved in chess calculations, i.e. minimax. We would like our qualitative engine to calculate in a way most similar to humans, and thus will require some level of ply depth to the calculations. However, a qualitative engine will have a much stronger sense of the "flow" of the game, and will thus explore fewer branches. Rather than considering each position as discrete, a qualitative engine should note how each move guides the evolution of the chess board position.

It is important to note that a qualitative chess engine may not be the most computationally efficient, a factor which was the primary motivation during the period of time when top chess engines needed to be run on supercomputers and every ounce of performance needed to be squeezed out of the machine. A qualitative engine should instead favor explainability over performance whenever possible. Specifically, it would be most ideal for an engine to produce an explanation of which moves were considered and why a particular move was chosen.

More modern qualitative research can improve upon Wilkins' knowledge-based PARADISE approach. It is important to recognize that his knowledge base is quite similar in nature to the FAC component of the retrieval model presented in (Forbus et al., 1995), but does not take advantage of the performance speedups presented there. Because of the high number of positional examples available online (Lichess, 2021), there is a huge opportunity for a performant analogical retrieval system at present. The MAC/FAC retrieval system could pay huge performance dividends in retrieval if applied to this problem.

Specifically, the Lichess database referenced above contains 1,737,489 chess "puzzles" as of the time of

writing. A chess puzzle is simply a chess board position in which players are encouraged to find the best move. Each puzzle relates to one or more chess “themes” (e.g. “mate in 1”, “pin”, “discovered attack”, etc.), analogous to Wilkins’ concepts outlined above. Each puzzle also includes the best move to make in the position. Some research will need to be done to derive meaning from this best move, relating it by analogy to the current position being evaluated by the engine.

Qualitative spatial calculi may also be used to construct more psychologically plausible models of chess positions than simply noting which pieces occupy which squares, seeking to emulate the models suggested by Chase and Simon. Chess pieces have intricate relationships which can be captured, and which change whenever a piece moves to another square. Importantly, however, not all relationships are affected by the movement of a single chess piece, suggesting that performance gains may be realized by recomputing only those relationships which have changed.

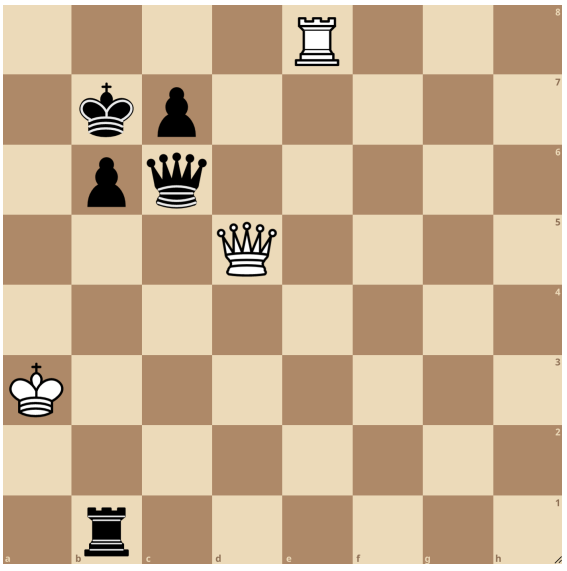

Figure 5. White to move. From this image, many basic piece relationships are apparent with only 8 pieces left on the board. Importantly, the black king is defending the black queen. The black queen is also being attacked by the white queen, and is attacking the white queen. White’s queen is undefended, a state sometimes referred to as hanging.

These relationships reveal the opportunity for a tactic. White can simultaneously move his rook to attack black’s king (danger levels) and take advantage of the defensive connection between the black king and queen with the move rook to b8. This forces black to move his king, removing the defense of his queen. In this position, black’s queen can be freely captured by white’s queen. Due to the mobility advantages of a queen over a rook, this is a favorable move sequence for white.

It is likely that low-level piece relationships may give rise to higher level relationships and tactics. For example, the concept of “capturing the defender” arises from the concept of attacking a piece A which defends B, which works when pieces A and B are attacked by pieces C and D of opposing colors. And in the case of **Figure 5** above, a defender may be “deflected” to win the piece it is defending. Defensive relationships may be thought of in a chain or directed graph, with each piece defending another and the safety of a piece being considered in relation to its connection to a defensive group.

§3A: Uses for a qualitative chess engine

There are many benefits to reopening the pursuit of qualitative reasoning in chess. The first and most clear value proposition is that qualitative reasoning is likely to serve as a more plausible model for how humans think about the game. This is evidenced by the fact that as Chase and Simon found, chess players do not “see” the whole board at once, but rather in chunks of interrelated pieces. Even if the details of human mental models differ slightly from the implementation of a qualitative reasoning engine, it will be able to provide a traceable account of its decision-making process, an important step towards explainability.

Current top chess engines reason about chess in ways that are quite contrary to human intuition.

Stockfish uses full-width search, considering each move in each position without prejudice and assigning numerical values to each position. As we saw from the analysis from GM Akobian, qualitative evaluations are far more meaningful to humans.

Other chess engines approach chess in an even more alien way. Specifically, it is unlikely that any engine which makes heavy use of neural evaluation functions will model human-derived organic strategies in ways which chess players will recognize. At the far end of this spectrum is the fully neural Maia chess engine, but even Lc0's Monte-Carlo tree search precludes consideration for cognitive plausibility.

Qualitative chess engines which are able to better reproduce the types of chess reasoning used by top human chess players are also likely to serve as better pedagogical tools for those interested in studying chess. This applies at every level, from beginner to grandmaster. The skill level of such a chess engine would be quite easily tunable simply by disabling more advanced knowledge from the knowledge base. This is a far more natural method of "handicapping" than the search depth limitations used in current chess engines. Each piece of knowledge becomes a tunable parameter to the engine. As students learn concepts, the corresponding representations in the knowledge base could be enabled, allowing for gradual learning in a far more accessible way. In fact, it is likely true that a qualitative chess engine could outperform human grandmasters (who often teach chess to others) in this respect.

Finally, it is likely that a qualitative engine would become a key component of a first line of defense against cheating in chess. Most cheating is performed by using assistance from a chess engine during online games with unsuspecting opponents. Consulting a functionally omniscient computer program can thus provide a cheater with a theoretically insurmountable advantage.

In an interview with the Perpetual Chess Podcast, Chris Callahan of the popular chess website LiChess.org stated that the majority of employees of the website work primarily to detect cheaters and yet the problem still persists (Perpetual Chess Podcast, 2021, 38:00). By exploiting the difference between conventional full-width engines like Stockfish and a qualitative evaluation, those working to detect cheaters will be better equipped to detect "suspicious" moves. However, qualitative chess engines are unlikely to be able to completely replace human moderation.

§4: Intentional limitation

In order to encourage reasoning and gameplay which resembles human games, this paper also proposes a regular computer chess tournament be held between chess engines. However, because we are not interested in the best overall chess engine, but one which can reason like a human might, the rules of the tournament will be adjusted in several key ways to discourage brute-force computational methods.

Because we expect few entrants in early iterations of this special tournament, engineering an automatic enforcement mechanism for the limitations stipulated in this paper are likely to be unnecessary. Engine compliance may simply be verified through manual inspection. Future iterations may include further safeguards, potentially separating the position evaluation function and directly counting the number of invocations while arbitrating the tournament to directly verify compliance.

§4A: Position limitation

Firstly, competing chess engines will be limited in the number of board positions they can evaluate during any one move. Because human

grandmasters evaluate around 100 positions before making a move, the tournament arbitration system will artificially impose this limitation on all competing engines.

This cap immediately creates an issue for full-width chess engines because of chess' high branching factor. Were an engine to evaluate each possible move, it would perform quite poorly in board positions with many possible plies and replies available, rarely reaching a depth of more than 2 or 3. As a result, any engine which naively assesses a chess board would perform quite poorly in this setup.

The practical upshot of the position limitation is that the engine will be incentivized to gather as much relevant information about a position as possible rather than optimizing for the maximum number of positions.

§4B: Position saliency

Additionally, engines will be required to implement scheduling logic which takes the time remaining into consideration. While this creates the immediate problem of how an engine should allocate its time, it creates the ancillary challenge of evaluating a position's quiescence. Positions which are "quiet" and have few forcing moves require less evaluation than positions in which there are many non-forcing moves.

This requirement immediately motivates a qualitative chess engine to recognize the futility of falling prey to the Horizon Effect. The Horizon Effect causes engines to waste many position calculations pursuing delaying moves which amount to hopeless rabbit trails, while any human evaluation can quite quickly understand the futility and terminate his search. A qualitative analysis which took this factor into consideration would be able to save a great deal of position calculations, behaving more like a human player.

Conclusion

Given that computers have achieved and sustained superhuman capabilities in the domain of chess, the next frontier is not in building yet stronger engines, but in harnessing the immense power to reason about the game in ways that humans do. Qualitative reasoning provides many advantageous tools to this end. Specifically, qualitative spatial calculi and analogical retrieval promise to provide novel and intuitive ways to reason about previously seen moves and think about the game.

References

- Barnes, D. (2019, April 30). What is the average number of legal moves per turn? Chess Stack Exchange. Retrieved June 3, 2021, from https://chess.stackexchange.com/questions/23135/what-is-the-average-number-of-legal-moves-per-turn/24325#24325
- Berliner, H. J. (1975). Chess as problem solving: the development of a tactics analyzer.
- Chase, W. G., & Simon, H. A. (1973). Perception in chess. Cognitive Psychology, 4(1), 55-81. 10.1016/0010-0285(73)90004-2
- Chess.com. (2020, October 31). Chess.com Computer Ratings: Nov. 2020. Chess.com. https://www.chess.com/article/view/chess-com-computer-ratings-nov-2020
- Chessify. (2021, January 22). NPS - What are the "Nodes per Second" in Chess Engine Analysis. Chessify. https://chessify.me/blog/nps-what-are-the-nodes-per-second-in-chess-engine-analysis
- Forbus, K. D., Denter, D., & Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. Cognitive Science, 19(2), 141-205. 10.1016/0364-0213(95)90016-0
- Frey, P. W., & Atkin, L. R. (1979). Creating a Chess Player. Byte, 4(1), 126-145. https://archive.org/details/byte-magazine-1979-01/mode/2up
- Leela Chess Zero. (2020, May 25). Technical Explanation of Leela Chess Zero. Leela Chess Zero.

https://lczero.org/dev/wiki/technical-explanation-of-leela-chess-zero/
- Lichess. (2021, May 16). lichess.org open database. Lichess. Retrieved June 6, 2021, from https://database.lichess.org/#puzzles
- McClain, D. L. (2017, January 16). Hans Berliner, Master Chess Player and Programmer, Dies at 87. The New York Times. https://www.nytimes.com/2017/01/16/business/hans-berliner-master-chess-player-and-programmer-dies-at-87.html
- McIlroy-Young, R., Sen, S., Kleinberg, J., & Anderson, A. (2020). Aligning Superhuman AI with Human Behavior: Chess as a Model System. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 10.1145/3394486.3403219
- Perpetual Chess Podcast. (2021, April 8). Chris Callahan of LiChess.org on their Twitch program, cheat detection,Open source engines and more [Video]. Youtube. https://www.youtube.com/watch?v=bmIFdrUVHXw
- Russell, S. J., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach (Third Edition ed.). Prentice Hall.
- Saint Louis Chess Club. (2013, March 11). Play Against the King's Indian Defense - GM Varuzhan Akobian - 2013.02.27 [Video]. YouTube. https://www.youtube.com/watch?v=h80Mu4N6oYI
- Slate, D. J., & Atkin, L. R. (1983). CHESS 4.5—The Northwestern University chess program. In Chess Skill in Man and Machine (pp. 82-118). Springer, New York, NY. 10.1007/978-1-4612-5515-4_4
- Stockfish. (2021, June 3). Stockfish. Stockfish: UCI Chess Engine. Retrieved June 3, 2021, from https://github.com/official-stockfish/Stockfish
- UoMCompsci. (2012, June 25). Kasparov vs Turing [Video]. Youtube. https://www.youtube.com/watch?v=wrxdWkjmhKg
- Wilkins, D. (1980). Using Patterns and Plans in Chess. Artificial Intelligence, 14(2), 165-203. 10.1016/0004-3702(80)90039-9
- Winston, P. H. (1992). Artificial Intelligence (3rd ed.). Addison-Wesley Publishing Company. https://courses.csail.mit.edu/6.034f/ai3/rest.pdf