NOTE

OPTIMAL ROBOT PLAY IN CERTAIN CHESS ENDGAME SITUATIONS

Michael Hartisch[1] *and Ingo Althöfer*[2]

University of Siegen, Siegen, Germany and Friedrich Schiller University, Jena, Germany

ABSTRACT

Robots are an aspiring factor in science, economy and society. We show that the adoption of human strategies and behavior is not always appropriate. Using the example of chess playing robots we demonstrate that the assumption of each move taking the same amount of time is invalid for robots at blitz chess. We present a different approach. Instead of optimizing the remaining number of moves in endgame play the actual time required for a move is considered and used for optimization. It is shown that this brings a significant change in gameplay.

1. INTRODUCTION

Since computers are able to beat human chess players easily[3] (Donninger and Lorenz, 2004) researchers are looking for new challenges. One approach is (1) to let a computer decide what move to play and (2) to let a *robot* perform this move in a physical environment. Chess robots have to recognize the current game position, compute a good move and perform this move in a reasonable amount of time. This combines several fields of research including artificial intelligence, robotics and optimization. Videos of robots playing chess can be found on the internet[4] and some professional robots play quite astonishing. In 2012, as a side event to the World Chess Championship in Moscow, two machines played five-minute blitz for the world championship title in robot chess. The robot CHESSKA created by Konstantin Kosteniuk defeated the KUKA MONSTER (henceforth KUKA) designed by the KUKA robotics company by 3.5 : 0.5. CHESSKA played much faster than its opponent and won three games since KUKA ran out of time.

This initiated the idea of investigating endgames from a different point of view. The usual approach is to minimize the number of remaining moves when being certain to win, while the player that is about to lose wants to delay the loss. However, when watching robots play chess, we can observe that there are some moves that take longer than others. Thus, it is too simplistic to rate the duration of each move equally. Especially when capturing pieces these robots need a lot of time to perform the move: the enemy piece has to be picked up and removed from the board. After that the own piece is moved to the target square and the game clock is pressed. In contrast, humans often pick up their own piece, move it to the piece to be captured and pick it up while placing their own piece. They sometimes even use this captured piece to press the clock. This kind of dexterity is not (yet) reached by robots. A second robot arm could be installed, but it could be argued that chess has to be played one handedly according to the FIDE laws of chess no matter who or what plays the game. Thus, we want to distinguish moves by means of the time needed to execute them.

In Section 2 we will introduce the time model used to evaluate a move and the method for building the endgame database. In section 3 some results are presented for the endgame with king+queen versus king+rook. Section 4 contains our conclusion.

[1]Chair of Technology Management, Faculty of Business Administration. email:michael.hartisch@uni-siegen.de
[2]Chair of Mathematical Optimization, Faculty of Mathematics and Computer Science. email:ingo.althoefer@uni-jena.de
[3] http://en.chessbase.com/post/adams-vs-hydra-man-0-5-machine-5-5
[4] for example http://blog.chessking.com/robot-blitz-chess-chesska-vs-kuka-monstr-game-one

2. MODELS AND METHODS

Below we discuss the time model (2.1) and the retrograde analysis (2.2).

2.1 The Time Model

Assume the following situation: Two robots are playing chess and Black has the king and the rook remaining on the board. White still has the queen and the king. The current position is a loss position for Black. However, **White is short of time** and wants to mate as fast as possible. In contrast, **Black has arbitrarily much time.** In this situation the usual endgame databases are useless, since a *loss in x moves* does not tell us how much time in seconds White will need to mate Black. To create an endgame database for this situation we introduce the following model: the location of the clock, (C) the depository for captured pieces (D) and the starting position of the white robot player (S) are displayed in Figure 1. The time needed for certain actions are as follows.

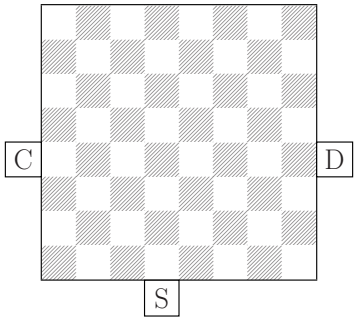

Figure 1: The location of the clock (C), the depository (D) and the starting position of the white robot arm (S).

Movement across the board: The robot arm is restricted to move along the files and ranks of the chess board. The movement from one square to an adjacent one takes 1 time unit (TU), regardless of whether a piece is carried or not. Moving from the starting position (S) to d1 takes 1 TU, as well as the movement from a4 to the clock (C) and from h4 to the depository (D) and vice versa.

Picking up a piece: Picking up an own or an enemy piece takes 3 TU, regardless of the piece types. This action can only be performed when the arm is located above the piece.

Putting down a piece: Putting down an own piece takes 3 TU. This action can only be performed when the arm is located above the target square.

Drop a piece: Dropping an enemy piece into the depository takes 1 TU. This action can only be performed when the arm is located above the depository (D).

Operate the clock: Pressing the clock button takes 2 TU. This action can only be performed when the arm is located above the clock (C).

Thinking process: Moves are taken from a precomputed data base. This is assumed to cost no time at all.

There are two basic moves the robot can execute: pure moves and capture moves.

Pure Move: The robot only moves one of its pieces. The time needed is composed as follows.

Movement of the robot arm to the own piece
\+ Picking up the piece
\+ Movement to the target square
\+ Putting down the piece
\+ Movement of the arm to the clock
\+ Pressing the clock

Capture Move: The robot captures an enemy piece. The time needed is composed as follows.

Movement of the robot arm to the enemy piece
\+ Picking up the enemy piece
\+ Movement to the depository
\+ Dropping the enemy piece
\+ Movement of the arm to the own piece
\+ Picking up the own piece
\+ Movement to the target square
\+ Putting down the own piece
\+ Movement of the arm to the clock
\+ Pressing the clock

We assume that there are no castling moves in this endgame. Note that the movement of the arm back to the starting position takes place but is irrelevant for the time, since the clock is pressed before. Further, note that for the sake of simplicity the time model demands the clock to be pushed even after delivering checkmate.

The following example illustrates the time cost for a move. Given the position in Figure 2 we investigate the move of the queen from c2 to f5, which is a capture move. The time cost composes as follows:

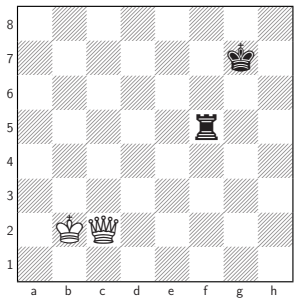

Figure 2: Example position with White to move. Moving the queen from c2 to f5 takes 44 TU, while moving from c2 to c7 only takes 22 TU.

$$\begin{aligned} & 7 \text{ (Arm to rook)} + 3 \text{ (Pick up rook)} + 4 \text{ (Move to depository)} + 1 \text{ (Drop rook)} \\ + \; & 8 \text{ (Arm to queen)} + 3 \text{ (Pick up queen)} + 6 \text{ (Move to target square)} + 3 \text{ (Put down queen)} \\ + \; & 7 \text{ (Arm to clock)} + 2 \text{ (Press clock)} = 44 \text{ TU} \end{aligned}$$

Comparing this capture move with the move from c2 to c7, which only takes 22 TU, we see that it is easy to cut the required time in half. Nevertheless, of course, we do not only want to save time by making fast moves, but we also want to make progress towards mate. Therefore, retrograde analysis was used to calculate the time-optimal endgame strategy.

2.2 Retrograde Analysis

To calculate the time to mate for every game position when both players play optimal the retrograde analysis algorithm was used. This algorithm evaluates game positions by examining backwards, starting from terminal positions which are in our case positions in which Black is mated. Positions which result in a win for Black are not part of this paper.

Retrograde analysis is also referred to as backward induction and was first introduced by von Neumann and Morgenstern (1944). Richard Bellman (1957) introduced the Principle of Optimality and developed the mathematical framework. Ströhlein (1970) was the first to use retrograde analysis to create chess endgame databases in 1970. Ken Thompson's (1986) thorough explanation was the foundation for our implementation.

3. RESULTS

We first take a look at some special positions to illustrate how severe the change in the optimization criteria is.

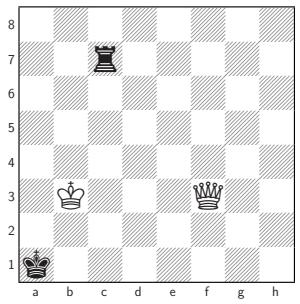

Figure 3: Position with Black to move and a certain loss for Black. When playing time-optimal Black can take 88 TU off White's clock.

Let us investigate the situation given in Figure 3 with Black to move and a certain loss for Black. The *normal* endgame takes three moves and goes as follows:

	White	Black	TU for White move
1.	..	Kb1	
2.	Qe4+	Rc2	20 TU
3.	Qxc2+	Ka1	36 TU
4.	Qb2+	1:0	16 TU

This sequence takes 72 TU for White. However, when both players play time-optimal the black player manages to take 88 TU off White's clock:

	White	Black	TU White move
1.	..	Rb7+	
2.	Qxb7	Kb1	48 TU
3.	Qh1+	1:0	40 TU

This sequence consists of only two moves, but is favorable for Black since White takes 16 TU longer than the *normal* endgame sequence.

Of course it is not always the case that Black is able to increase the time needed by the white player to achieve a win. Let us consider the game position with White to move in Figure 4. Here White could win within one move

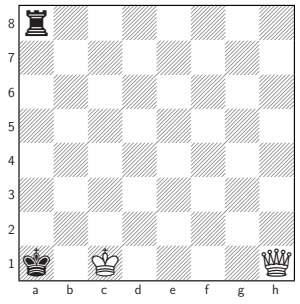

Figure 4: Position with White to move and a certain win for White. Capturing the rook would take 58 TU. Instead White can perform three moves requiring only 56 TU also resulting in a win.

by capturing the rook. However, this move takes 58 TU. Instead of this move the white player should move the king to c2 resulting in the following move sequence:

	White	Black	TU White move
1.	Kc2+	Ka2	16 TU
2.	Qb1+	Ka3	24 TU
3.	Qb3+	1:0	16 TU

These three moves take only 56 TU off White's clock.

Note that the location of the clock, the depository and the starting position of the robot is crucial to this analysis. When flipping the position given in Figure 4 we obtain the position in Figure 5. Remember that the position

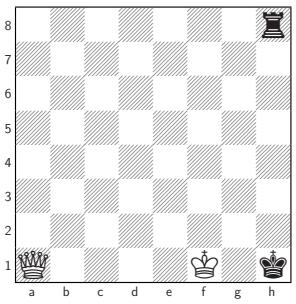

Figure 5: Position with White to move and certain win for White. Capturing the rook would take 66 TU and is optimal for both move-optimization and time-optimization.

of the clock is still on the left hand side and the depository is located on the right hand side. In this position the capturing move (Qxh8+ 1-0) is both time-optimal **and** move-number-optimal and takes 66 TU, whereas the (mirrored) three presented moves above (1. Kf2+ Kh2 2. Qg1+ Kh3 3. Qg3+ 1-0) would take 20+28+22=70 TU. Thus, different positions of the special locations (clock, depository and starting position) change the outcome of the analysis significantly.

Let us now consider all positions with White to move, with White having at most its queen and Black having at most its rook on the board, in addition to their kings. We are interested in how big the time difference

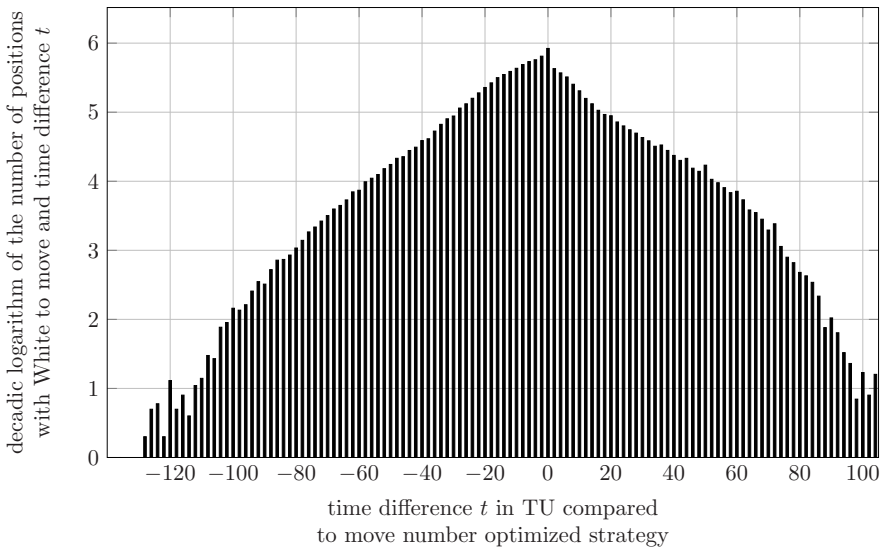

Figure 6: Chart displaying the logarithmic number of positions with White to move and a time difference t.

can become when comparing our time-optimal strategy to the usual move-number-optimized endgame databases. Figure 6 shows the logarithmic number of positions with White to move with regard to the time difference when comparing move-optimized and time-optimized strategies. A negative time difference for a position means that with time-optimal play less time is needed to achieve a checkmate. Note that uneven time differences cannot

occur due to our time model and are omitted in the plot. On the one hand, there are positions in which the white player can save more than 130 TUs compared to normal endgame play. On the other hand there are positions in which more than 100 TUs are added to White's clock, when both players use time optimized strategy. The chart for positions with Black to move looks quite similar and is omitted at this point.

Finally, let us briefly look at the largest occurring differences in the moves until mate. There are game positions in which five moves less are executed when both sides are using the time-optimal strategy compared to both sides using move-number-optimized strategy. However, in some positions our strategy leads to a game sequence with six moves more until mate. This also shows that there is a significant difference between optimizing the time until mate and optimizing the number of moves until mate.

4. CONCLUSION

We showed that when robots play chess under strict conditions the usual endgame tables may lead to suboptimal decisions. The assumption in human play that each move takes the same amount of time is invalid for robots. Thus, when running out of time it is necessary to optimize the actual time needed instead of the number of remaining moves until mate.

Our results highly depend on the used time model and the estimated time for the movements. Furthermore, they depend on the robot under investigation. Some robots are able to move diagonally and others might be able to pick up pieces faster than others. Nevertheless, our research shows that there is another way to defend oneself in situations with a lack of time, at least when you are a robot or play against robots. For human play our results might not be as helpful, since the assumption that each move requires the same amount of time is quite reasonable (of course without taking the thinking time into account).

Further research could deal with the advantage a player can gain when being aware of this time-optimal strategy whereas the opponent only uses the classic databases. Also, we only investigated situations in which one player is short of time. In a situation where both players are under time pressure a bicriterial retrograde analysis is required: we have to decide between minimizing our time and maximizing the time the opponent needs to respond to our actions.

We discussed the influence of real time duration of moves. In a forthcoming counterpart we will analyze a simple version of *robot frisbee play*, where instead of normal pieces frisbee disks are used, and the moves are throws of these frisbees[5]. Of course, imperfect players (humans or robots) are not always performing perfect throws, letting the frisbee land on wrong cells of the board. Taking the probability of such imperfection into account often forces players to make moves which would be suboptimal in normal play. Games like Yavalath[6], Nine Men's Morris and Go could be interesting when such imperfect moves are involved.

5. REFERENCES

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition. ISBN 9780691146683.

Donninger, C. and Lorenz, U. (2004). The Chess Monster Hydra. *Field Programmable Logic and Application* (eds. J. Becker, M. Platzner, and S. Vernalde), Vol. 3203 of *Lecture Notes in Computer Science*, pp. 927–932. Springer Berlin Heidelberg. ISBN 978–3–540–22989–6.

Neumann, J. von and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press.

Ströhlein, T. (1970). *Untersuchungen über kombinatorische Spiele*. Ph.D. thesis, Fakultät für Allgemeine Wissenschaften der Technischen Hochschule München.

Thompson, K. (1986). Retrograde Analysis of Certain Endgames. *ICCA Journal*, Vol. 9, No. 3, pp. 131–139.

[5] http://www.althofer.de/robot-play/frisbee-robot-go.jpg
[6] http://www.cameronius.com/games/yavalath/