Learning of Piece Values for Chess Variants

Technical Report TUD–KE–2008-07

Sacha Droste, Johannes Fürnkranz
Knowledge Engineering Group, Technische Universität Darmstadt

Abstract

In this paper we describe a set of experiments for learning a the values of chess pieces for the popular chess variants crazyhouse, suicide, and atomic chess. We follow an established methodology that relies on reinforcement learning from self-games to learn piece values and piece-square tables for these chess variants. The learned piece values are quite different from those of standard chess, and in several ways surprising, but they generally outperform those that have been previously used in the literature, and in implementations of computer players for these games. The results also underline the practical importance of piece-square value tables for tactical variants of the game.

Contents

1 Introduction

Chess is arguably the most popular game in the western hemisphere, which is not only enjoyed in its standardized version but also in numerous variants (Pritchard, 2007). Chess variants can differ from standard chess in various ways: they may change the board layout, the number of players, the movements of the pieces, the pieces themselves, the starting positions of the pieces, etc. Chess variants may have a long cultural tradition, like Japanese Chess (Shogi) or Chinese Chess (Xiangqi), or may have enjoyed a comparably recent popularity, such as Kriegspiel, where one only sees one's own pieces, composer Arnold Schönberg's Coalition Chess (Bündnis-Schach), a variant for four players played on a 10×10 board, or the 3-dimensional Star Trek Chess, which has enjoyed some popularity with the Star Trek TV series.

The rise of Internet Chess has also increased the popularity of certain chess variants. On chess servers like the *Free Internet Chess Server*, various chess variants enjoy immense popularity. One group of variants, most notably Fischerandom chess (Gligoric, 2003), differs from standard chess only through different starting positions. Conventional chess programs can be easily adapted to these variants.

However, another very interesting group of chess variants maintains the standard set of pieces and their movements, but changes the dynamics of the game in a way such that it can be expected that evaluation functions from standard chess cannot be re-used. For these variants, not even the value of the pieces is entirely clear. For example, what is the value of a queen in Suicide Chess, where the goal is to get rid of one's own pieces as quickly as possible? Does its high mobility make it a good piece (because it can be sacrificed at various places) or a bad piece (because it can be easily forced to capture)? What is the value of a pawn in hand in Crazyhouse chess, where the captured pieces change colors and can at any time be re-inserted instead of a move by the capturing party? Sacrifices on squares like f7 followed by a series of check-delivering pawn drops to force out the king are often the start of an irresistible attack in this chess variant. Or what is the value of a bishop in Atomic Chess, where captures cause explosions that remove every piece on a neighboring square?

In this paper, we try to answer such questions by using machine learning techniques to automatically tune evaluation functions for programs for these chess variants. In particular, we apply *temporal difference learning* to the problem of learning an evaluation function for the popular chess variants, Crazyhouse, Suicide Chess, and Atomic Chess. We closely follow the approach taken by Beal and Smith (1997; 1999a; 2001), who have applied this methodology to standard chess and Shogi. However, while Shogi uses several pieces that move differently than in regular chess, the variants that we consider in this paper differ from standard chess only slightly. In particular, the basic movements of the pieces are the same for all chess variants considered in this work, so that the results for these variants can easily be compared. One of our goals is to intuitively interpret the observed differences in the learnt values, and see whether they can be explained by the differences in the rules of the chess variants.

For each variant, we conduct two experiments. In the first experiment a simple material-only function is learnt while the second experiment uses piece square tables. The learnt function will be analysed through automated tournaments on one hand, and intuitive explanations of striking aspects of the learnt values on the other hand.

Our work does not provide new methodological insights, but, to our knowledge, there are no well-established piece values in these chess variants. Thus, we believe that our answers, which are based on established statistical learning techniques, are of considerable interest to human and computer players of these games.

2 Chess Variants

In this section, we will start with a brief introduction to the three chess variants that we will deal with in this work, namely Crazyhouse Chess, Suicide Chess, and Atomic Chess. For each variant, we will briefly sketch the rules of these games and discuss what aspects of the game were most interesting to us.

2.1 Crazyhouse

Crazyhouse Chess is derived from the popular 4-player chess variant Bughouse. In Bughouse, 2 teams face each other on two boards, each team plays black on one board and white on the other. The key twist is that if a player from Team A captures one of her opponents pieces, she may pass the piece to her partner, who can, instead of moving one of her own pieces, place the captured piece on any free square on the board (at any time in the game). Simple communications between partners (like "I need a knight" or "Don't move!") are typically allowed. The game is won by the team that first delivers a mate on one of the two boards or lost by the team that first runs out of time on one of the two boards. Clocks are an essential part of this game, because deadlocks have to be avoided, where, e.g., each team has a mate-in-2 on one of the two boards and thus the respective opponents on this board refuse to move.

Crazyhouse chess[1] is a two-player variant of this game. Whenever a player captures a piece, this piece changes its colour (from black to white or from white to black), and may be placed on the board at a later point in the game (alternative to a move). A common strategy is to collect multiple pieces in ones hand, and to later draw the opponents king out with a number of successive piece placements. If a promoted piece is captured (i.e., a piece that once was a pawn but was promoted to some other piece), it is transformed back to a pawn. Because of the magical change of colours, this game cannot be played with regular chess pieces, but only virtually[2] The game ends like regular chess, when one of the players mates the opponent or similar. Deadlocks cannot happen, and thus a clock is not necessary (but typically employed nevertheless).

Our main motivation for choosing this game is the unclear value of the pieces that a player holds in hand. For example, the combination of various pawns and knights in hand can often be a dangerous weapon because pawns can often be placed next to a king and either draw the king out or yield a protected square next to the king which can be used for other pawns or pieces, and knights are the only pieces that can deliver without being threatened by the king and without giving the opponent the chance to bar the check by dropping a piece.

2.2 Suicide Chess

There are several variations of Suicide Chess (aka as Giveaway Chess on some chess servers).[3] They all have in common that the goal of the game is no longer to mate the opponents king (and secondary goals are to take the opponents pieces) but to get rid of ones own pieces. All pieces, including the king, move like in conventional chess.

However, the king can be taken just like any other piece, checks do not have any significance and, in particular, do not restrict a players move choices. Instead, any other possibility to take one of the opponents pieces restricts ones move choices in the sense that one has to take a piece if this is at all possible (sometimes this has to be announced by the opponent, similar to a spoken "check"). If one can take several pieces, one can choose at will between those possibilities.

There are several possibilities for dealing with the situation when one player still has pieces, but no valid moves. According to the rules that are implemented at the *Free Internet Chess Server (FICS)*, this is a win for the player that has fewer pieces on board. In case of a tie, the game ends in a draw.

Contrary to regular chess, it is unclear how valuable the pieces are. For example, one may think that the queen is a very bad piece because it offers the opponent many opportunities for placing a piece so that it may be captured. On the other hand, it is also the piece with the largest number of move options, so that it can often allow the player to choose from multiple capture opportunities (which is better than having only one forced choice). It is also the piece that can be most easily disposed with. A pawn, on the other hand, does not provide many opportunities for an opponent, but is also quite hard to get rid of. There are no generally agreed-upon values for these and all other pieces of this chess variant.

[1] cf., e.g., `http://wiki.wildchess.org/wiki/index.php/Crazyhouse`

[2] It could be played with two sets of real chess pieces, but this may quickly become confusing. We are not aware of any real tournaments of this variant.

[3] cf., e.g., `http://wiki.wildchess.org/wiki/index.php/Suicide`

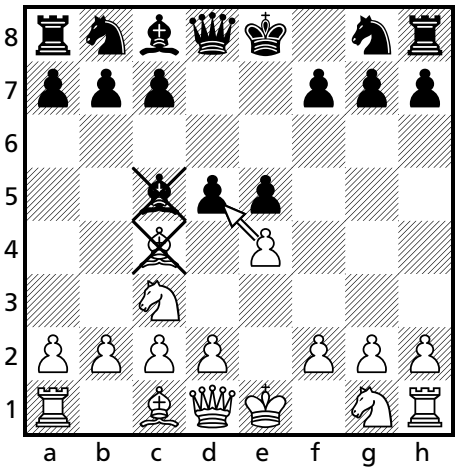


Figure 2.1: Explosions in Atomic Chess

A general heuristic[4] says that rooks are more valuable than kings, queens, and bishops (they are all equal), which are in turn more valuable than knights. Pawns are the least valuable pieces. As we will see, our learned values will slightly deviate from these specifications.

2.3 Atomic Chess

In Atomic Chess[5], capturing a piece leads to an explosion that extends from the square on which the piece is captured to all neighboring squares (horizontally, vertically, or diagonally). The effect of the explosion is that, in addition to the captured piece, the capturing piece and all non-pawns on neighboring squares are removed from the board. For example, if white captures on d5 in Figure 2.1, the pawns e4 and d5, as well as the Bishops on c5 and c4 will be taken off the board. The pawn on e5 remains (explosions only concern pieces).

Thus, there are direct and indirect threats to a piece: A direct threat is a threat to capture the piece as in regular chess, an indirect threat is the threat to capture one of its neighboring pieces. Naturally, one can also lose by indirect mate, when a piece adjacent to the king is captured.

Again, there are minor variations of the game, which do not further concern us here. We used the variant that is played at the Free Internet Chess Server (FICS).

Our main interest in this game was to see whether the increase in capturing power of each piece will have a significant effect on the piece values in comparison to regular chess.

[4] http://wiki.wildchess.org/wiki/index.php/Suicide

[5] cf., e.g., http://wiki.wildchess.org/wiki/index.php/Atomic

3 Reinforcement Learning in Chess

For each of the considered chess variants (including standard chess), we automatically tuned an evaluation function via reinforcement learning. As the approach follows established techniques, we only briefly sketch the main ideas in this section, but refer to the literature for details.

3.1 Features and Evaluation Functions

In our experiments, we use a simple linear evaluation function of the form

$$P(x) = \sum_f w_f \cdot f(x) \tag{3.1}$$

where $f(x)$ are elementary feature values that are computed for a given chess position x, and w_f are the weights that are automatically tuned in the learning phase. For brevity, we omit the board x from the notation wherever it is clear from the context, and simply refer to a feature f.

In this work, we consider two types of features:

Material: These features indicates for each piece type the difference in the number of pieces of that type for white and for black. In standard chess, these are 5 features (there is no feature for the king because both sides will always have exactly one king).

Piece-Square Tables: These features indicate for each piece type and for each possible square on the board, whether this square contains a piece of this type ($f = 1$) or not ($f = 0$). There is only a table for white, which is mirrored if black is to move. In standard chess, there are $5 \times 64 + 48 = 368$ features of this type (piece-square tables for pawns do not need the first and last row).

These basic features are combined with weights w_f, which reflect their relative importance. For example, in standard chess, a knight typically counts three pawns, i.e., the knight weight will be three times as high as the pawn weight ($w_N = 3 \cdot w_P$). Thus, if white has two pawns for a knight, the feature f_P will have the value 2 and the feature f_N will have the value -1, yielding a material position evaluation of

$$\begin{aligned} P &= w_P \cdot f_P + w_N \cdot f_N + w_B \cdot f_B + w_R \cdot f_R + w_Q \cdot f_Q = \\ &= 2 \cdot 1 + 3 \cdot (-1) + 3 \cdot 0 + 5 \cdot 0 + 9 \cdot 0 = -1 \end{aligned}$$

Determining such feature weights automatically from experience is the subject of this paper. Note that the absolute value of the feature weights is irrelevant, only their relative values to each other determine which of a pair of positions is evaluated higher. However, they are typically normalized in terms of pawn values (i.e., $w_P = 1$).

For brevity, the above example only considered the material evaluation. If piece-square values are added, the number of features and corresponding weights increases, but the shape of the function remains the same.

3.2 Reinforcement Learning

The key idea of *reinforcement learning* (Sutton, 1988; Tesauro, 1992; Sutton and Barto, 1998) is to reinforce good moves by increasing the weights of features that contributed positively to the selection of this move, and by downgrading bad moves. Essentially, the *credit assignment problem*, i.e., the problem of deciding which moves in a game were bad and which ones were good, is simply solved by assuming that all moves in a won game were good, and all moves in a lost game were bad. If many million games are played, statistics will ensure that good moves will occur more frequently in won games than in lost games.

Temporal-difference learning, the reinforcement learning algorithm that is most frequently used in game playing, is able to achieve a faster convergence by adapting the evaluation functions of a position at time t not towards the final outcome of the game, but towards the evaluation of the position at time $t+1$. More precisely, the update rule of $TD(\lambda)$

$$\Delta W_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} (\lambda^{t-k} \nabla_W P_k) \tag{3.2}$$

where α is a learning rate, P_{t+1} and P_t are two successive position evaluations, and λ is a parameter that influences how strongly positions P_k with $k < t$ should influence the weight adaptation of the current position. For example, $\lambda = 0$ means that only the last position P_t will have an influence, for $\lambda = 1$ all past positions of this game have the same influence.

For computing the weight update $\nabla_W P_k$, we first transform the linear evaluation function P' into a sigmoid function with values between 0 and 1. A typical function is

$$P(x) = \frac{1}{1 + e^{-\omega P'(x)}} \tag{3.3}$$

where ω is a parameter that influences the steepness of the curve. For this type of function, the weight update for a single feature f of the linear function is

$$\nabla_W P_k = f \cdot P_k \cdot (1 - P_k) \tag{3.4}$$

3.3 Reinforcement Learning for Deep Searches

The greatest success story of reinforcement learning is Tesauros TD-Gammon, which achieved world champion strength entirely by learning from self-play (Tesauro, 1992; 1995). However, backgammon programs have a rather shallow search (three ply and no search extensions for TD-Gammon). In games like chess, deep searches are necessary for expert performance. A problem that has to be solved for these games is how to integrate learning into the search techniques. In particular in chess, one has the problem that the position at the root of the node often has completely different characteristics than the evaluation of the node. The solution for this problem is to base the evaluation on the *dominant position* of the search. The dominant position is the leaf position in the search tree whose evaluation has been propagated back to the root of the search tree (i.e., the position reached in the principal variation).

Consequently, one should adapt the weights of the dominant positions and not the weights of the root position to ensure the estimation of the weight adjustments is based on the position that was responsible for the evaluation of the current board position. This problem has already been recognised and solved by Samuel (1959) but seemed to have been forgotten later on. For temporal-difference learning, this solution was rediscovered independently by Beal and Smith (1997) and Baxter et al. (1998b). Beal and Smith (1998) subsequently applied this technique for learning piece-values in Shogi. A proper formalisation of the algorithm can be found in (Baxter et al., 1998a), (Baxter et al., 2000), and (Baxter et al., 2001).

For our experiments, we followed the description of the TDLeaf(λ) algorithm (Baxter et al., 1998c). The learning rate α was set using Temporal Coherence (Beal and Smith, 1999b). The key idea of this technique is to maintain a local learning rate α_f for each individual feature f, which degrades when the weight value for this feature stabilises (i.e., when the changes in the positive and the negative direction tend to equalise over time).

We will not further go into the technical details here, but refer to these original publications. A good overview of reinforcement learning in chess-like games can be found in (Ekker, 2003), surveys of learning approaches in computer chess in (Skiena, 1986) and (Fürnkranz, 1996), and general overviews of machine learning in computer game-playing in (Fürnkranz, 2001) and, more recently, (Fürnkranz, 2007).

4 Experimental Setup

As in most previous works, we used reinforcement learning to train an evaluation function from automated self-play only. It never played against a human player and no endgame or opening databases were used; i.e. no expert knowledge has influenced the experiments.

4.1 Learning Piece Values and Piece-Square Values

For each of the considered chess variants (including standard chess), we conducted two experiments. Each experiment consisted of multiple runs, whose results were averaged at the end.

The *small experiments* consisted of 10 runs 2000 games each. Within these, a material-only evaluation function was learned, similar to the setup in (Beal and Smith, 1997). The number of weights to be learned resulted from the chess variants. In standard and Atomic chess, one value was learned for each piece except of the king, in Suicide Chess an additional value was learned for the king, and in Crazyhouse additional values were learned for the pieces a player holds in the hand.

The *big experiments* consisted of 20 runs 10,000 games each, and learned an evaluation function composed of material values and piece-square tables as described by Beal and Smith (1999a). In all of the examined variants a separate piece-square table was learned for every piece, including the king.[1] Beal and Smith (1999a) state that it depends on the phase of the game if the king's value is higher at the lower end of the board or in the centre, however, this does not necessarily have to be true for chess variants. For this reason (and to ease comparison later) the king received his own piece-square table in all of the variants. This lead to $6 \times 64 = 384$ additional weights in the big experiments.[2]

One could think of reducing the number of parameters by exploiting the vertical symmetry of the board. We deliberately did not do this because in many of these chess variants the asymmetry between king and queen side is much more pronounced than in standard chess. For example, in Crazyhouse, long castling rarely ever happens, and much of the action focusses on attacks on the king side. Similarly, in Atomic Chess, attacks via f7 are deadly. As we will see, these dynamics are clearly visible from the learnt piece-square values.

4.2 Chess Engine

For our experiments, we adapted the open source chess engine SunSetter[3], which can deal with standard chess, Bughouse, and Crazyhouse. The program provides a simple implementation of standard computer chess techniques (cf., e.g., (Levy and Newborn, 1991)) uses a conventional alpha-beta search (Knuth and Moore, 1975) with quiescence search (Kaindl, 1982). It also has a primitive learning algorithm in the form of persistent transposition tables (Breuker et al., 1997). In all experiments, we used the chess engine with search depth 5, complemented with a quiescence search. Opening and endgame libraries were turned off.

We extended the chess engine, so that it is in addition able to play Suicide and Atomic chess, and, most importantly, integrates a reinforcement learning algorithm for learning feature weights as described in the previous section. The weights to be learned were all set to 1 at the beginning of each run, in accordance with (Beal and Smith, 1997). To reduce (or even prevent) the repetition of games during the learning phase, which is necessary so that self-play does not converge towards repeating the same game over and over again, ties in the evaluation of possible moves in a given situation were broken randomly. For the material-only evalution, this happened quite frequently and was sufficient to avoid repeating games. In the experiments for learning piece-square tables, this was insufficient because the piece-square tables will give a different evaluation to almost every possible situation on the board. Here, we slightly randomised the output of the evaluation function (about 1% of the evaluation value) in order to ensure a minimum level of exploration.

[1] In Crazyhouse piece-square tables were learned only for pieces on the board, not for the placement of pieces in hand.

[2] 16 of these weights, those for pawns on first and last rank, are irrelevant and will never change. For implementation reasons, we nevertheless kept them.

[3] `http://sourceforge.net/project/showfiles.php?group_id=61676`

4.3 Evaluation

Our main interest was to see whether the learned values make sense for a chess player. To objectively ensure their quality, the learned values from both experiments for each variant were benchmarked in a concluding tournament. We have also directly compared the learnt values to values found in the chess program Sjeng, which is also able to play various chess variants.[4]. Following Beal and Smith, each pair of programs played 2000 games.

[4] `http://www.sjeng.org/indexold.html`, we used version 11.2

5 Standard Chess

In order to ensure the correctness of our implementation, we also repeated the experiments in standard chess of Beal and Smith (1997; 1999a). We report them here so that we can directly compare our results in the chess variants to those obtained for standard chess. For each game variant, we will start with the results of the two experiments (direct learning of piece values and learning via piece-square values), and then try to interpret and compare the learnt values.

5.1 Piece Values

Figure 5.1 shows the average course of the 5 values learnt throughout the 2000 games played. Note how the values barely change after the 500-th game. This shows that the chosen length of 2000 games per run is sufficient The final average values with their standard deviations are depicted further below, in Figure 5.3.

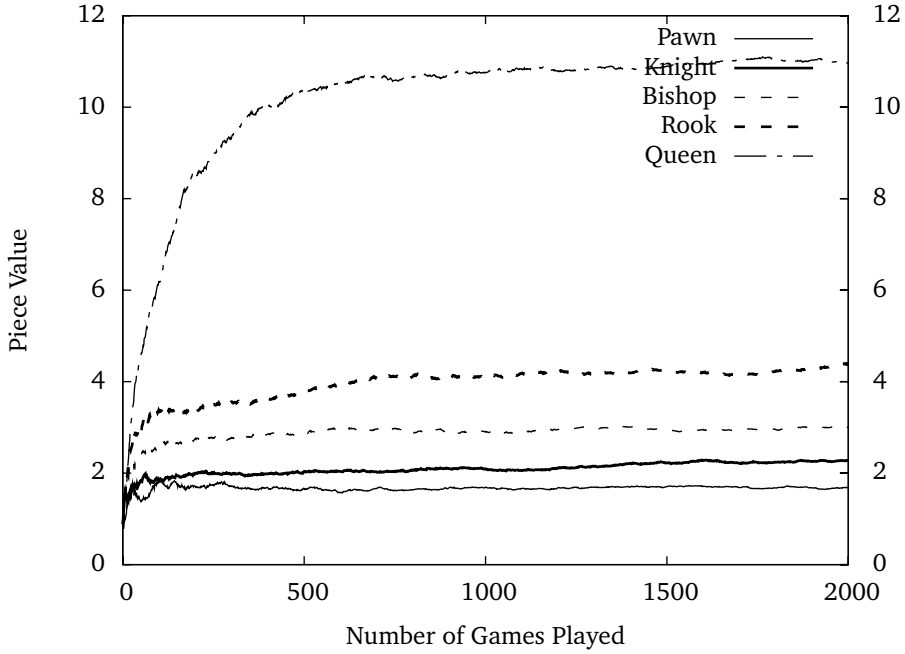


Figure 5.1: Development of material values

For practical reasons, we have used a different normalisation for each chess variant. We will present our results in this normalisation, but summarise our results later on with the standard normalisations, where pawns equal to 1 (Table 9.1). Here, we use a normalisation that scales the bishop's value to 3.

5.2 Piece-Square Values

The learnt piece-square values are shown in figure 5.2. The style of the diagram follows the style suggested by Beal and Smith (1999a). In fact the diagram here looks almost like an inversion of the diagram in (Beal and Smith, 1999a). The white square in the diagram designates the position at which the according piece reaches its peak value. Respectively, the black square shows the position with the lowest value. Underneath each diagram the minimum and the maximum values are given (i.e., the values of the white and the black square).

Figure 5.2.1 shows how the value of a pawn increases with his rank on the board. The ranks 1 and 8 just hold their initial value since their squares are never occupied by a pawn and are thus never modified. Figure 5.2.2 shows that a knights value decreases with increasing proximity to the boards edge. This can be explained by his reduced mobility. In contrast to the other officers (rook, tower, queen) the knight has less valid moves on the border than in the middle of the board. Likewise the bishop has lower values closer to border, however, looking at the minimum and maximum values,

the effect does not bare the same relevance here. The increased value of the rook on the higher ranks is explained by Beal and Smith in (Beal and Smith, 1999a) by the ability to threaten the opponents pawns and king. The min and max values in the kings diagram are not the total values of the king but just the normalised entries in the piece-square table, since in standard chess the base value of the king is hypothetically ∞. The bright squares at the bottom of the diagram probably result from an advantageous position of the king in the middle of the game. The bright squares in the upper area (around f6) are probably due to an advantageous position during endgame when it is important for him to preserve his mobility and to avoid getting cornered.

5.3 Comparison of the Learnt Values

The learnt base values and the piece-square tables were used to calculate average piece values. In this calculation the entries of the piece-square tables were weighted by the frequency of a piece visiting the according squares, again following (Beal and Smith, 1999a). For this purpose the frequencies of the pieces visiting all the squares have been stored and summed up throughout the learning process. Figure 5.3 shows the average material values obtained from 20 runs. On top of each bar, we also show the standard deviation of the obtained estimate.

For comparison, the values from both experiments are shown in table 5.1 together with those taken from BS97 (Beal and Smith, 1997) and BS99 (Beal and Smith, 1999a) and the original values from Sunsetter C10. The values in this table, though, have all been normalised in the same manner to ease comparison.

	Avg. P.-Sq.	Piece Val.	Sunsetter	BS97, Trial C	BS99, Fullboard
Pawn	1,05	1,20	0,96	0,96	0,97
Knight	2,82	2,43	2,87	2,14	2,69
Bishop	**3,00**	**3,00**	**3,00**	**3,00**	**3,00**
Rook	4,51	4,35	4,49	4,51	4,50
Queen	9,38	9,48	8,60	8,82	9,21

Table 5.1: Comparison of different sets of material values (standard chess)

The sets of values are all quite similar. Minor differences can be explained by the statistical nature of of $TD(\lambda)$. Different games played during the learning process lead to slightly different values. This is, for example, confirmed by the different results of the trials A to E in (Beal and Smith, 1997) and the standard deviations shown in Figure 5.3.

The results of the tournament comparing these values are given in Table 5.2. The entries in the configuration column have the following meanings:

learnt: material-only evaluation function using the learnt piece values

traditional: material-only evaluation function and traditional material values

original: material-only evaluation function and material values taken from Sunsetter C10

average p-sq: material-only evaluation function and material values set to the average piece-square values of Table 5.1 (column Avg. P.-Sq.)

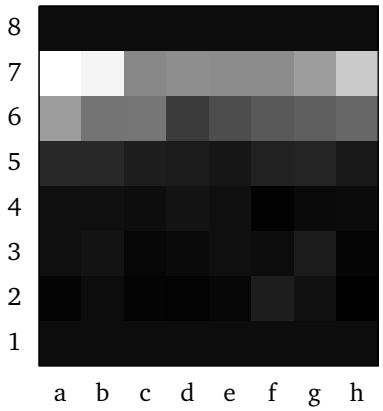
piece-square: the same evaluation function as above and the respective average learnt configuration values

Sjeng: The chess engine Sjeng[1] version 11.2

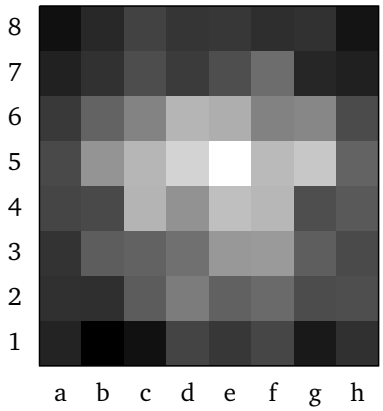
Configuration	Won	Lost	Draw	Ratio
learnt vs. traditional	1246	715	39	63,28%
learnt vs. original	1037	906	57	53,28%
learnt vs. average p-sq	971	993	36	49,45%
piece-square vs. learnt	1735	245	20	87,25%
piece-square vs. Sjeng	2	1997	1	0,13%

Table 5.2: Results of the tournament for standard chess

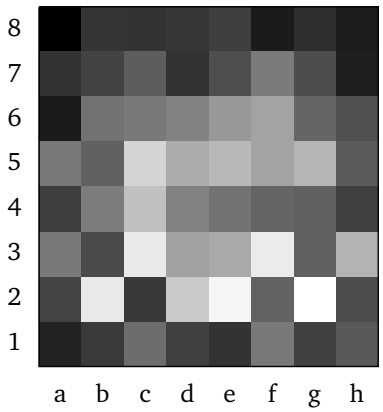
[1] http://www.sjeng.org/indexold.html



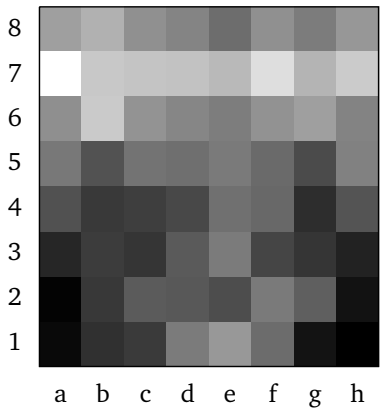
5.2.1: Pawn (Min/Max: 0,92/2,86)



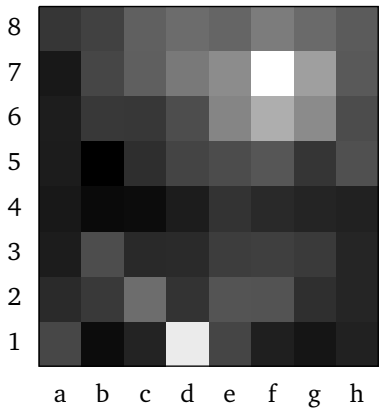
5.2.2: Knight (Min/Max: 2,61/3,13)



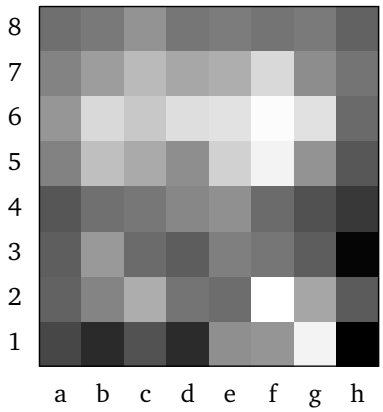
5.2.3: Bishop (Min/Max: 2,83/3,12)



5.2.4: Rook (Min/Max: 4,41/4,9)



5.2.5: Queen (Min/Max: 9,1/9,55)



5.2.6: King (Min/Max: 0,60/0,85)

Figure 5.2: Piece-Square tables in standard chess

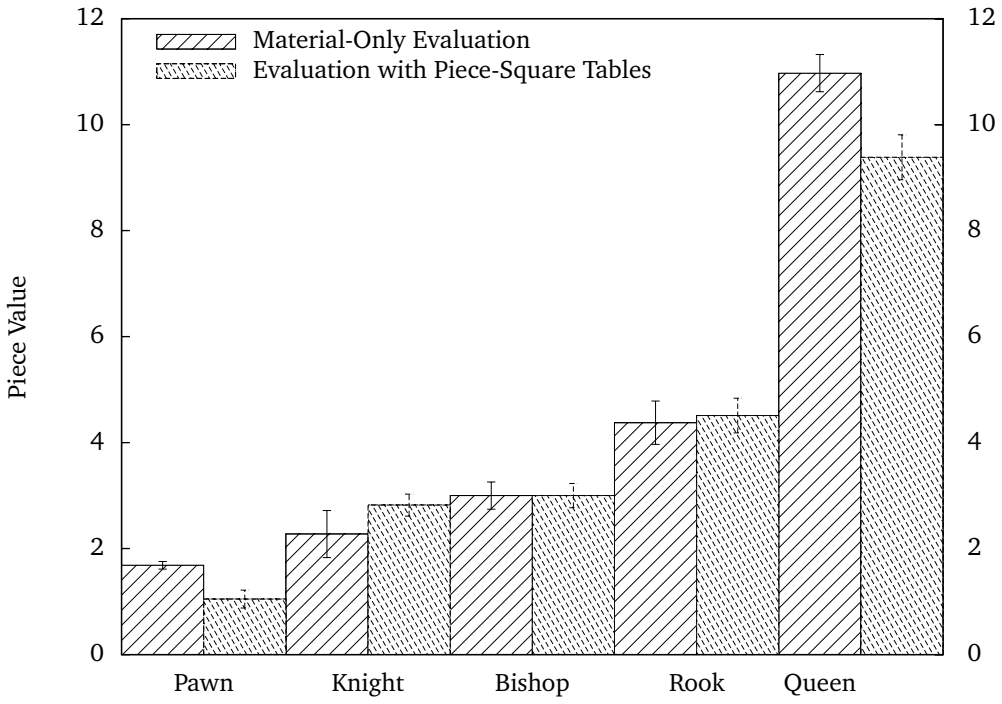


Figure 5.3: Final material values of the pieces

The results show how the learnt values actually lead to stronger play than values retrieved by other means. They also show that the engine plays significantly stronger with piece-square tables than with a material-only evaluation function. It is interesting to see that the winning ratio of learnt values against the averaged piece-square table values is slightly less than 50%. The last line shows a comparison to a fully fledged chess engine, and clearly demonstrates that (not surprisingly) piece-square tables alone are not enough for a strong chess evaluation function.

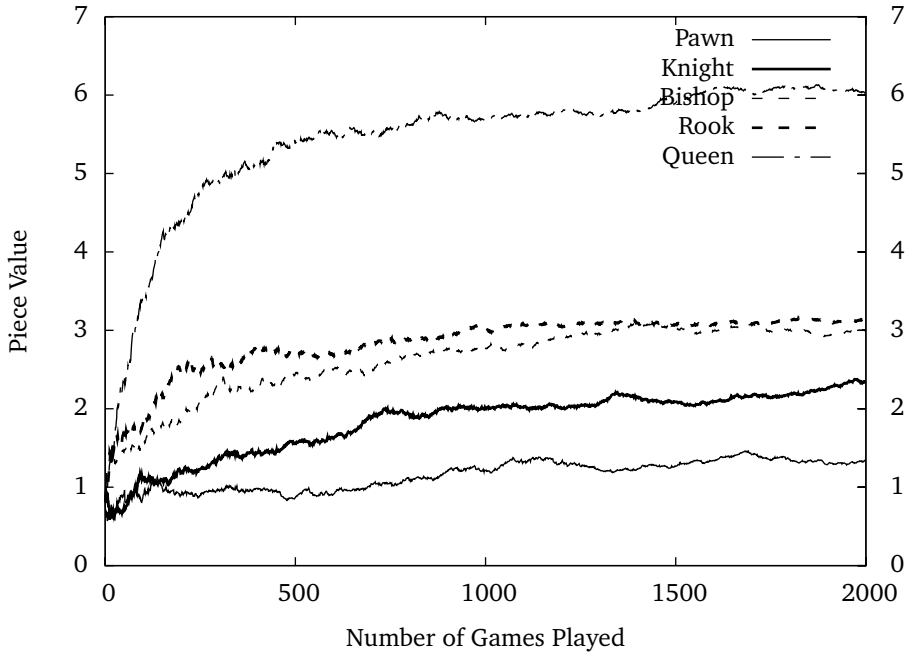


Figure 6.1: Development of piece values on the board (Crazyhouse)

6 Crazyhouse

6.1 Piece Values

The development of the average learnt values for crazyhouse pieces is shown in Figure 6.1. The values have been normalised like the values from standard chess, i.e. a bishop on the board has a value of 3.

The diagrams here resemble those from standard chess. The general appearance of the curves is in accordance with those in Figure 5.1 and the values of pawns and bishops have barely changed, however, both of the stronger pieces rook and queen have significantly decreased in value. The queens value has almost halved while the rook has almost the same value as the bishop now. The low value of the queen probably results from the generally higher mobility on the board which is caused by the dropping moves. In comparison with these teleportation moves, the queens natural high mobility doesn't look so impressive anymore.

The development of the on-hand pieces' values is shown in Figure 6.2. They have also been normalised with the on-board bishop value, i.e., their final values are directly comparable to the on-board values. In particular knights, rooks and pawns are worth more on the hand than they are on the board, which is consistent with our practical experience with this game. In all likelihood this is caused by the strong correlation between a piece's position on the board and its value. The learnt piece-square values will show if this theory holds since in that case the difference between the highest and the lowest value of a specific piece should be rather big.

It should also be noted that we made the simplifying assumption that the value of an on-hand piece is independent of the other pieces that the player is holding. This is almost certainly not the case. If we do not hold any pawns yet, the value of a new pawn in hand will typically be higher than if we are already holding three pawns.

Interestingly, the convergence of the on-hand values is slower than in Figure 6.1. The reason for this is the fact that these features seldomly have a value different from 0. This is because, for example, the queen usually gets dropped again immediately after it has been taken to the hand.

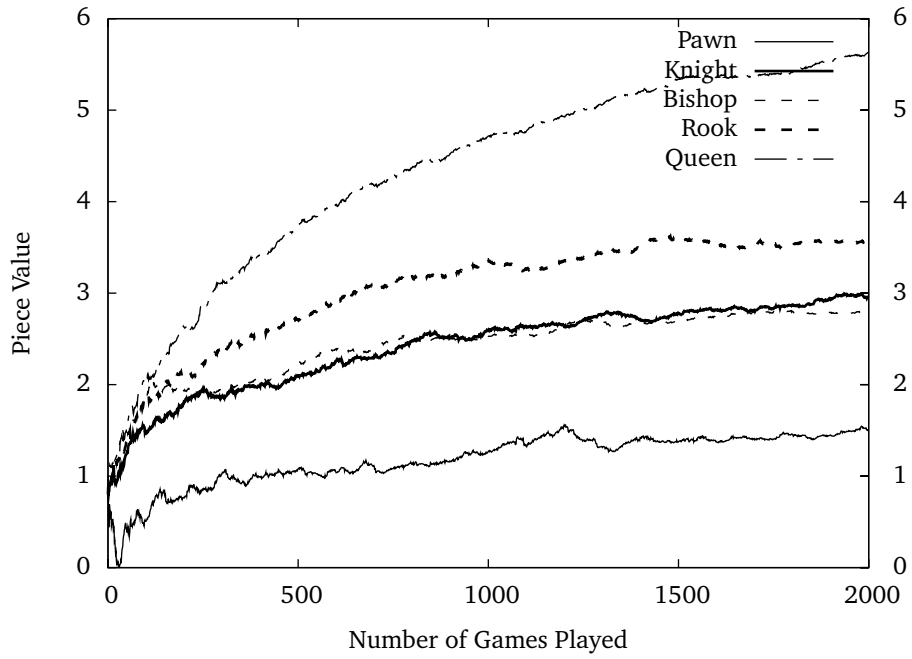


Figure 6.2: Development of material values of on-hand pieces (Crazyhouse)

6.2 Piece-Square Values

Figure 6.3 shows the learnt piece-square tables for Crazyhouse. The diagrams resemble those from standard chess but there are still some apparent differences whose explanation is based again on on-hand pieces and dropping moves. First, we can see that the assymmetry between king and queen side is much more pronounced than in standard chess, and that for all pieces, there is a clear preference for the king side. This reflects the fast dynamics of the game, where long castling rarely ever happens (even short castling is considerably rarer than in standard chess). Long castling would typically have to be followed by a king's move to b1 to eliminate the imminent danger of a rook or queen drop on a1, and thus loses much more time than short castling. As a consequence, pawn drops, a major attacking device in this game, almost never happen on the queen side, but always near the opponent's king (to lure him out) or near the own king (to increase his protection and prevents piece drops in these places). Esssentially, similar patterns can be observed for all pieces.

The pawns diagram differs particularly strongly from its standard chess counterpart. Since a pawn can placed on the seventh rank via a dropping move its value does not necessarily increase with its rank. The high values on the higher ranks can still be explained by the imminent promotion, however, the high values on the lower ranks show the increased importance of king safety. In particular the f-pawn is clearly discouraged from moving because doing so will give the opponent the chance to start a dangerous attack with a pawn drop on f2.

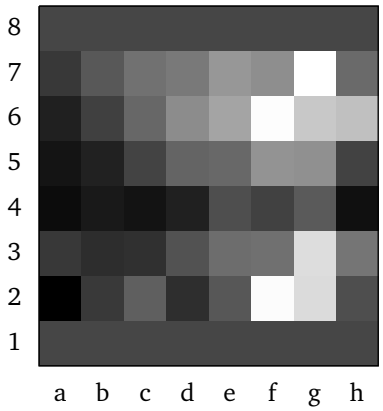
The most valuable square for the knight (Figure 6.3.2) has moved from the centre position e5 to f6, where it immediately threatens the starting position of the opponents king. A knight's check from f6 is often the start of a deadly attack, and clearly this is also the favourite spot for a dropping move.

The recognition of any apparent pattern in Figure 6.3.3 is difficult. It is notable, though, that the difference between the lowest and the highest value is bigger than in standard chess. This leads to the assumption that the absolute position of the bishop on the board is more important in crazyhouse than it is in standard chess. The strong preference for fianchettos (squares b2 and g2) has somewhat diminished, which is also consistent with our personal experience (although we would think that the weight for a bishop on g2 is still too high considering its vulnerability to pawn droppings on f3 or h3. One can also see that Bishop droppings on g7 and f6 (attacking the rook and penetrating the enemy's king side) are quite good.

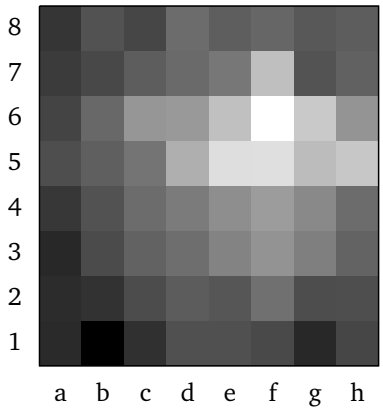
The rooks domain of usefulness on the board is now restricted from the upper ranks in general to the upper right corner of the board (g8 and h8), where it can be expected to do the most harm to the opponents king safety.

The queen, too, has undergone a comparable change. Its diagram still has a bright area in the upper right part of the game board, however, the square d1 has lost its high value and vanishes into a generally dark zone. Just like the bishop the queens difference in value between the best and the worst square has strongly increased.

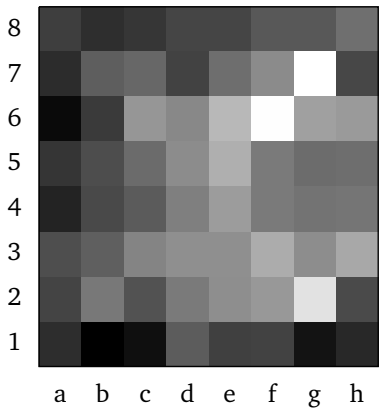
The king is clearly safest in the corner, where it has the least number of neighboring squares that need to be protected from drop pieces. The large grey area in the top half of Figure 6.3.6 results from the abolition of the endgame. Since no



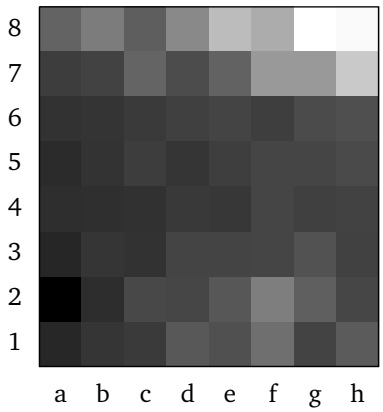
6.3.1: Pawn (Min/Max: 0,66/2,14)



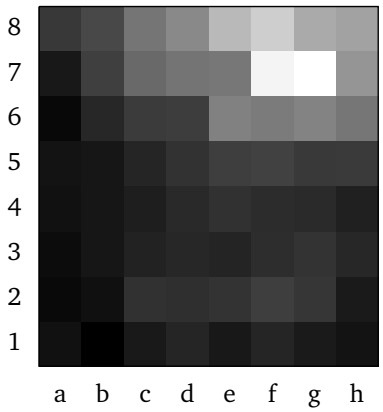
6.3.2: Knight (Min/Max: 1,53/3,97)



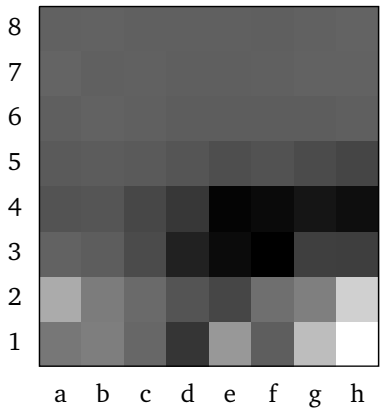
6.3.3: Bishop (Min/Max: 2,38/3,87)



6.3.4: Rook (Min/Max: 2,3/4,8)



6.3.5: Queen (Min/Max: 4,38/6,91)



6.3.6: King (Min/Max: -0,97/4,25)

Figure 6.3: Piece-Square tables in Crazyhouse

piece can ever permanently leave the board, no situation arises that corresponds to the usual endgame in standard chess. The bright area from Figure 5.2.6 does not reappear here and the upper ranks (5–8) are barely used by the king. For this reason these squares' values remained practically unchanged.

6.3 Comparison of the Learnt Values

The average material values for pieces on the board that have been calculated from the piece-square tables are shown in Figure 6.4, together with their standard deviation.[1] The same normalisation has been used as in section 5 for standard chess.

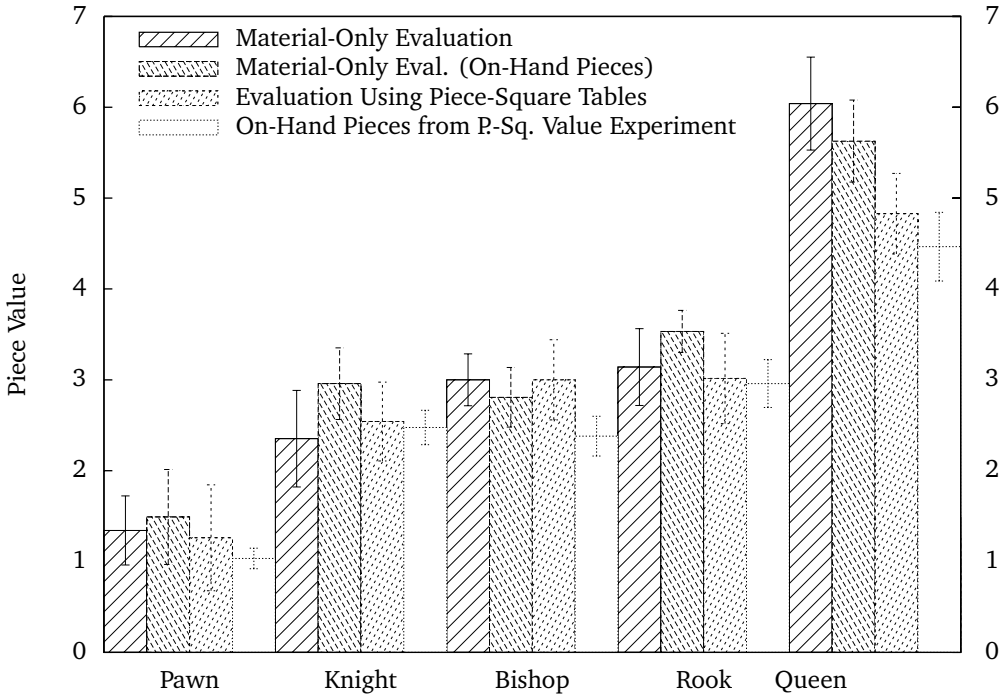


Figure 6.4: Final material values of the pieces

Table 6.1 compares the learnt piece values, the averaged piece-square values, the values from Sunsetter 7e, and the values used by Sjeng.

	Avg. P.-Sq.	Piece-Value	Sunsetter	Sjeng
Pawn	1,26	1,34	1,54	1,30
Knight	2,54	2,35	2,95	2,74
Bishop	**3,00**	**3,00**	**3,00**	**3,00**
Rook	3,02	3,14	3,08	3,26
Queen	4,83	6,04	6,00	5,87
Pawn (Hand)	1,03	1,49	1,54	1,30
Knight (Hand)	2,48	2,96	2,95	2,74
Bishop (Hand)	2,38	2,81	3,00	3,00
Root (Hand)	2,96	3,53	3,08	3,26
Queen (Hand)	4,47	5,63	6,00	5,87

Table 6.1: Comparison between different sets of material values (Crazyhouse)

The results of the tournament are presented in Table 6.2. The entries in the configuration column have the following meanings:

[1] The values of the on-hand pieces correspond (normalised) to the ones learnt. No calculational blending with piece-square tables of any kind has been performed.

learnt material-only evaluation function using the learnt piece values

traditional material-only evaluation function and traditional material values; on-hand pieces have the same values as pieces on the board.

original material-only evaluation function and material values taken from Sunsetter 7e

Sjeng (Values) material-only evaluation function and material values taken from file eval.c from Sjeng

average p-sq material-only function and material values set to the calculated ones in Table 6.1

piece-square the same evaluation function as above and the respective learnt configuration values

Configuration	Won	Lost	Draw	Ratio
learnt vs. traditional	1352	646	2	67,65%
learnt vs. original	1317	680	3	65,93%
learnt vs. Sjeng (Values)	1322	674	4	66,20%
learnt vs. average p-sq	813	1184	3	40,73%
piece-square vs. learnt	1928	71	1	96,43%

Table 6.2: Results of the tournaments (Crazyhouse)

With a win ratio of about 66% the first three lines show that even a small optimisation of piece values can lead to a stronger play. The observation that we made for standard chess, that the learnt values perform worse than those that are calculated as weighted averages of the piece-square tables, aggravates to a degree which can not solely be explained by statistical effects.

The win ratio of the program using piece-square tables even surpasses the ratio from the according standard chess tournament. This is in compliance with the increased differences between the minimum and maximum values in Figure 6.3. This confirms that due to the high mobility (pieces in hand can be placed at any square on the board), good piece placement gains importance, and thus piece-square tables are even more important for crazyhouse than they are for standard chess. This is also witnessed by the increase of the span between the minimum and maximum piece-square values for most pieces.

7 Suicide

7.1 Piece Values

Figure 7.1 shows the development of the learnt piece values for Suicide Chess. Note that all values are negative because we use the same chess engine for all chess variants, and this engine tries to maximise the material score. Usually a material advantage is advantageous, and for that reason, positive values are learnt for all the pieces. In Suicide, however, it is typically bad to have more pieces on the board than the opponent. Because of this, the pieces receive negative values, thereby leading to a high evaluation score if the player has *less* pieces on the board than his opponent.

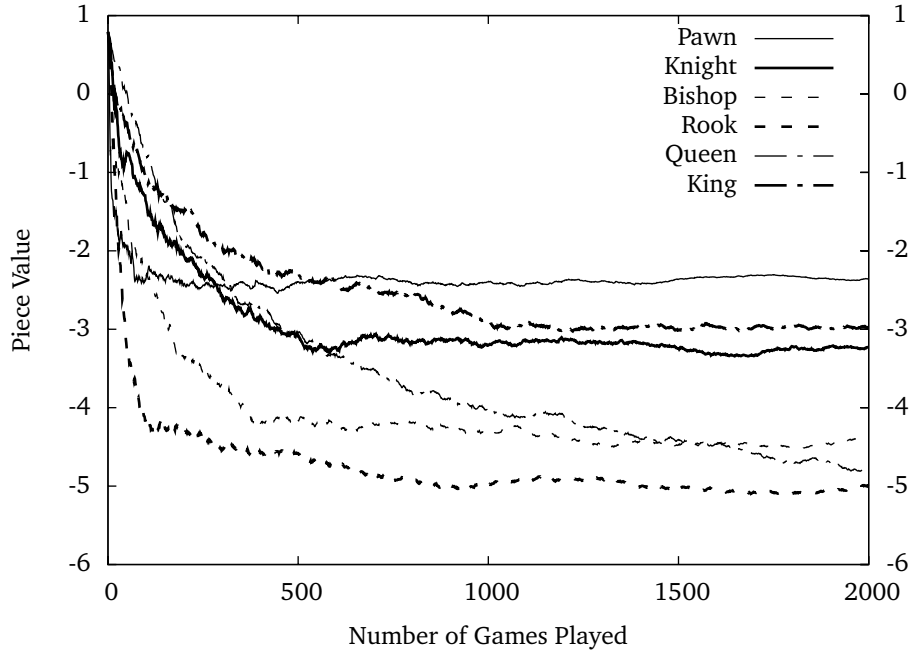


Figure 7.1: Development of material values (Suicide)

Very low (very negative) values, like the ones of the rook and the queen, signify that it is *bad* to have these pieces on the board and it is *good* if the opponent has these pieces on the board. The rooks and queens values are due to their many move options. Anyone who has played Suicide before is bound to have noticed how difficult it is to bar the queen from attacking any of the opponents pieces. The rook, however, poses a bigger problem than the queen, because the queen usually has *multiple* options of attack. That leaves at least a few tactical options and the queen can frequently quickly be moved towards her own demise. On the other hand it is easy for the opponent to position his pieces in a way, that enables the players rook to capture exactly *one* piece without afterwards being captured himself. Following this principle a player can often construct a critical chain of capturing moves. The knight is barely more dangerous than the king because it is so easy to bereave him of a majority of his move options by putting him close to the border of the board. Similarly, the pawns (absolute) low value can be explained by their limited capture options. Even if a pawn ever captures a piece, it is almost impossible to create a capture chain as mentioned above with the rook. Neither does a promotion pose any problem since suicide allows pawns to be promoted to kings.

The final values of the pieces are shown in Figure 7.3 together with their respective standard deviations. The normalisation here has been conducted to lead to a value of -5 for the rook. This choice enabled a better comparability with the other values, which we will discuss further below (Table 7.1).

7.2 Piece-Square Values

In Figure 7.2 the learnt piece-square tables are shown for suicide. The diagrams 7.2.1–7.2.6 seem to be rather chaotic in comparison to the diagrams of the other chess variants. A certain symmetry is apparent in the diagrams of the bishop and the rook. But since the very bright and very dark squares are isolated and no gradients are visible the interpretation is rather troublesome, however, the quite big difference between the minimum and maximum values indicates that the differences are not due to chance fluctuations.

For pawns one can, e.g., see that centre moves are not good openings. Computer analysis in this game has, e.g., shown that the opening moves h4, f4, Nf3, Nc3 , d3, d4, e4, h3, b4 are all losses for white. With the exception of e4 and b4 these all have a negative piece-square value. However, one should be cautious with such an interpretation, as the learnt values reflect the value of a square over the entire game, not only in the first move. Some other observations that can be made include, e.g., that knights are strong on f7, but bad on c3 and f3, bishops and rooks are good on a6 and h6 (this is the square where they can be most quickly captured). The king has a much higher mobility, and is good on most squares, but it is strongly discouraged to move out of its original square, etc. Particularly interesting is the queens table, which shows that the queen has only one particularly strong square, which is d7. In the opening, the queen can be captured on this square by four pieces, and it will in addition open the opponens queens file where possibly other pieces can be disposed.

7.3 Comparison of the Learnt Values

Again, averaged values of the pieces have been calculated from the piece-square tables. The calculated values are listed in Table 7.1 together with the directly learnt piece values. In their ordering, the values are quite consistent, but the values originating from the piece-square tables are somewhat lower relative to the rook. The order is also quite consistent with general wisdom in this game, where the importance of the pieces is given as `R > K = Q = B > N > P`[1], meaning that kings, queens, and bishops have approximately the same value, lower than rooks and higher than knights. Pawns have the lowest value. In our experiments, the king has received a much lower weight in our experiments. It is unclear to us whether this is a significant deviation or whether this can be attributed to, e.g., the fact that the endgame has certainly had a lower importance in our experiments than in expert play.

	Piece-Square Values	Piece Values	Sjeng (Losers)	Sjeng (Suicide)
Pawn	-2,89	-2,36	1,14	0,50
Knight	-3,44	-3,22	4,57	5,00
Bishop	-3,68	-4,41	3,86	0,00
Rook	**-5,00**	**-5,00**	**5,00**	**5,00**
Queen	-3,92	-4,79	5,71	1,67
King	-2,81	-2,98	14,29	16,67

Table 7.1: Comparison of different sets of material values (Suicide)

For comparison the table also includes values used by Sjeng in two variants, Suicide Chess and Losers Chess.[2] The difference between the two is that in Losers Chess, the king cannot be captured. What is surprising here, is the high value for the king. For the Losers variant, we would not need any value for the king, and for the Suicide variant the value of the king seems to be excessively high. The reason is probably that Sjeng uses a very complex evaluation function, which is optimised for Suicide, as opposed to the general one used in our experiments. Nevertheless, these values seem to be quite weak.

Results from the tournament are listed in Table 7.2. The entries in the first column have the following meanings:

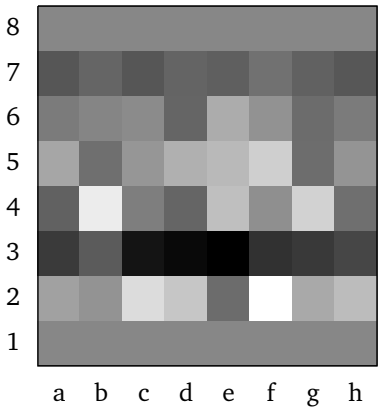
learnt material-only evaluation function using the learnt piece values

traditional material-only evaluation function and inverted traditional material values, i.e. the values have been multiplied with -1. The kings value has been set to -2 because he moves and captures almost like a pawn with some added mobility.

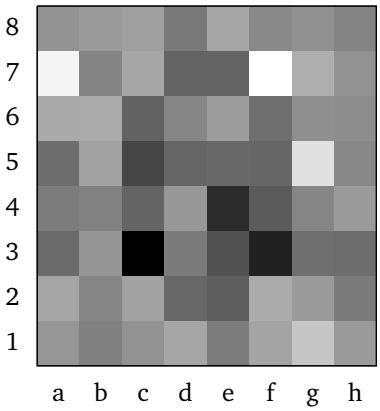
Sjeng (Suicide) material-only evaluation function and the material values used for Suicide by Sjeng. All these values have been multiplied with -1 for adaption to the evaluation function used by DaSunsetter.

[1] `http://wiki.wildchess.org/wiki/index.php/Suicide`

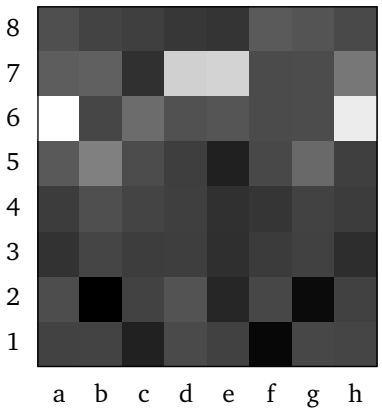
[2] These values can be found in the file `eval.c`



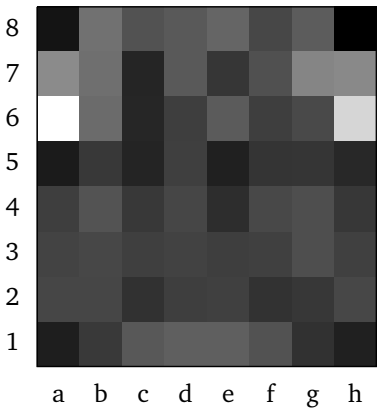
7.2.1: Pawn (Min/Max: -3,76/-2,45)



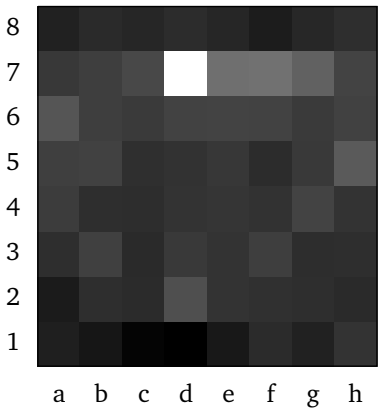
7.2.2: Knight (Min/Max: -4,09/-2,99)



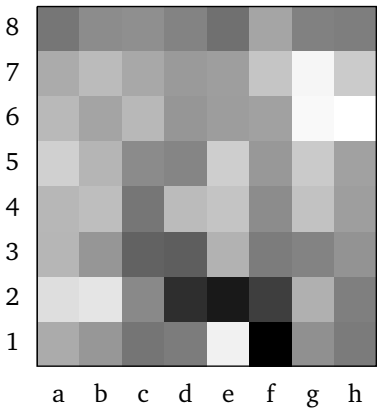
7.2.3: Bishop (Min/Max: -4,13/-1,64)



7.2.4: Rook (Min/Max: -5,43/-3,07)



7.2.5: Queen (Min/Max: -4,20/-1,64)



7.2.6: King (Min/Max: -3,15/-2,66)

Figure 7.2: Piece-Square tables in Suicide

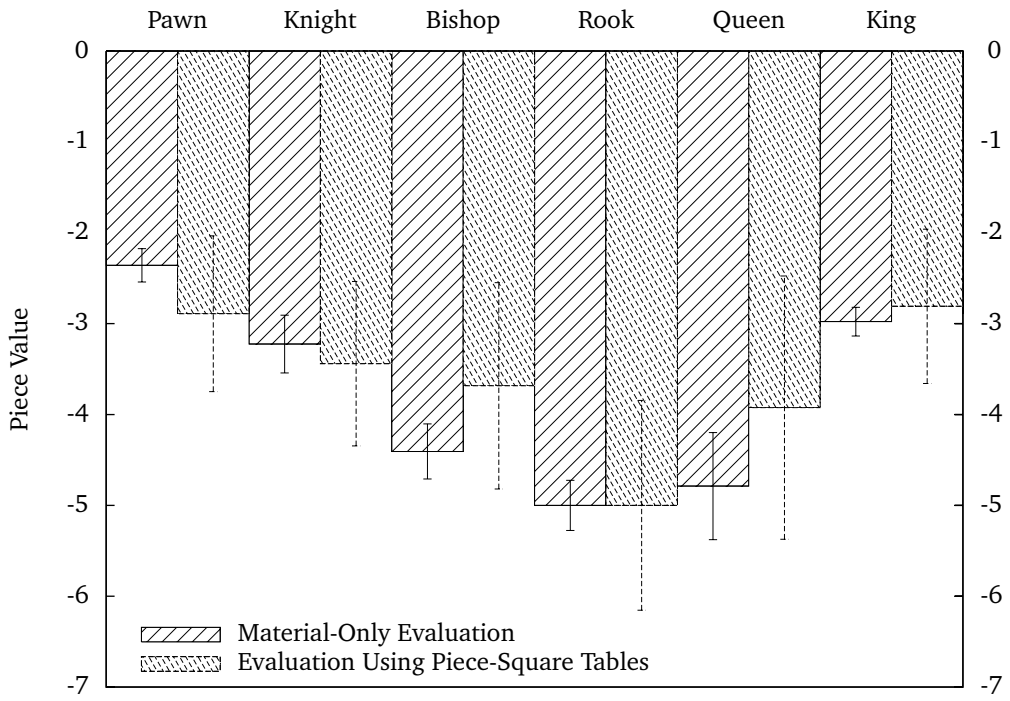


Figure 7.3: Final material values of the pieces

Sjeng (Losers) material-only evaluation evaluation function and the material values used for Losers by Sjeng. All these values have been multiplied with -1 for adaption to the evaluation function used by DaSunsetter.

average p-sq material-only evaluation function and the average piece-square values shown in Table 7.1

piece-square the same evaluation function as above with the learnt piece-square values

Configuration	Won	Lost	Draw	Ratio
learnt vs. traditional	1018	934	48	52,10%
learnt vs. Sjeng (Suicide)	1790	185	25	90,13%
learnt vs. Sjeng (Losers)	984	910	106	51,85%
learnt vs. average p-sq	986	948	66	50,95%
piece-square vs. learnt	1742	243	15	87,48%

Table 7.2: Results of the tournaments (Suicide)

Despite the plausibility of the learnt values, it seems that a simple inversion of traditional values yields almost the same results as the learnt values. The win ratio is statistically significant with a p-value of 6% according to the conservative sign test, but we would nevertheless have expected a larger advantage here.

The bad performance of the suicide values of Sjeng confirms the suspicion that Sjeng uses for Suicide a strongly specialised evaluation function. The values taken from Losers just perform better because of their similarity with the learnt values, however, it shows that even substantial noise in the values (see Table 7.1) has only a slight impact on the results of the games.

Again, the engine with piece-square tables plays significantly better than the material-only version, which confirms that the learnt piece values are not due to chance fluctuations. The exact placement of the pieces on the board seems to be of similar importance as in standard chess.

8 Atomic

8.1 Piece Values

The normalisation of the Atomic values has been conducted in a way that leads to a value of 2,5 for a knight. The curves in Figure 8.1 show barely any changes after the 1000-th game. This means that the algorithm converges nicely for Atomic and the 2,000 games conducted per run are absolutely sufficient.

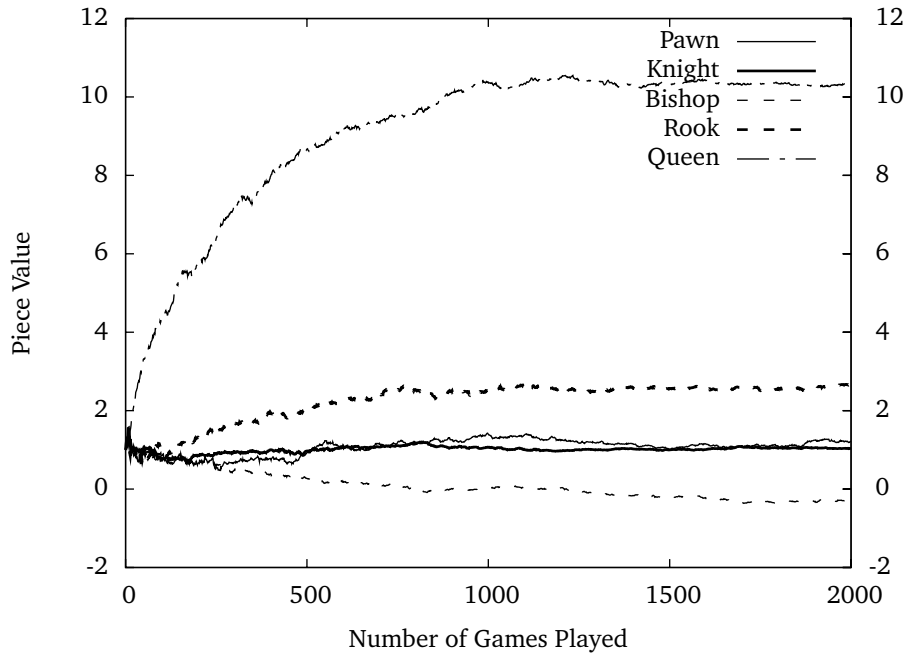


Figure 8.1: Development of material values (Atomic)

Notice the extremely high value of the queen which is hard to explain. The fact that (because of the explosion) every piece can only capture once per game leads to the assumption that with a skilled capture, a piece has to capture at least as many of the opponents pieces as are needed to compensate for the loss of the capturing piece itself. With such a high value as the queens in relation to the other pieces' values, however, this is barely achievable. Even if the queen (≈25 points) captures two enemy rooks (≈12 points), both knights (5 points), and a pawn[1] (≈2,5 points) the capturing player still loses more points than his opponent. Because of this, the queens value is probably due to her high potential to threaten the king. Her high mobility and the potential of indirect check or check-mate give her the ability to sometimes threaten the king in two different ways[2] at the same time and thereby check-mate him all by herself.

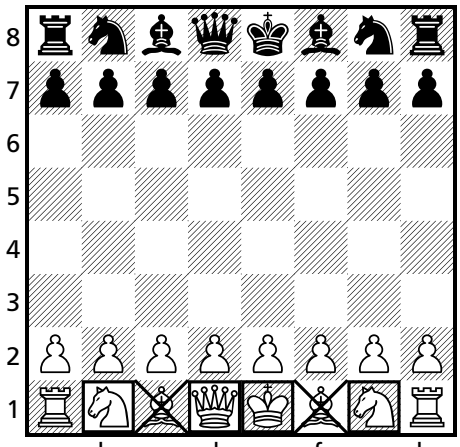
Somewhat surprising is also the slightly negative value of the bishop. We are not expert enough to this game to confirm or disprove this statement. However, in a recent on-line book on atomic chess (Blackburn), the bishop does not play a prominent role. For example, the book contains separate chapters on queen, rook, and knight tactics, but none for the bishop. The bishop is only discussed in certain endgames. An explanation for the negative value might be its unfortunate starting position. A threat to the bishop on its starting square usually also threatens the queen or the king (see Figure 8.2.1).

8.2 Piece-Square Values

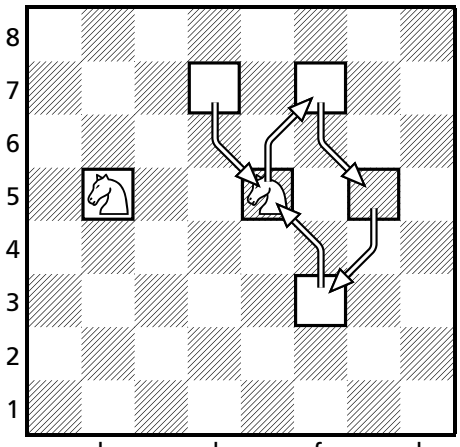
Figure 8.3.1 looks similar to its standard chess counterpart: A gradual transition from light at the bottom to dark at the top is apparent and probably the reason for this is again the promotion of a pawn upon reaching the eighth rank.

[1] At most one pawn can be captured at once since pawns are not affected by the explosion.

[2] This is also possible in standard chess but in Atomic it is much easier to achieve.



8.2.1: Pieces threatened when a bishop is captured



8.2.2: Advantageous positions for the knight

Figure 8.2: Diagram with the pieces on the board

The knights diagram looks rather random at first but a closer look reveals some patterns. Many of the slightly lighter patches are just a knights move apart (see Figure 8.2.2). In particular, the pattern reflects that 1.Nf3 is an extremely dangerous opening to which only f6 or e5 are playable to prevent both 2.Ne5 or 2.Ng5, both with imminent doom on f7. All these squares have very high values for the knight. On the other hand, its starting positions are particularly bad. This does not come as a surprise as a similar threat can appear here as shown in the previous section with Figure 8.2.1.

In this context, the bishops diagram comes as a small surprise. In contrast to the results of directly learnt piece values, the bishop has a clearly positive value, with a small difference between the minimum (2,07) and the maximum (3,33) values. Also, the starting positions (c1 and f1) do not seem to be valued negatively. Overall, however, the image looks rather random, and the low span between min and max values leads to the conclusion that bishop placements are not particularly important for the game.

The rook diagram clearly shows the danger of rooks in this game. They can mate the opponent king from the distance by, e.g., moving over e1 to e8 or perform a similar manoeuvre to reach one of its neighboring squares. The queens diagram is quite similar, but it has a much larger dark area, which probably results from the danger of capturing neighboring pieces in case of an explosion if the queen gets captured. However, even in this dark area, the queens value still exceeds the other pieces' values.

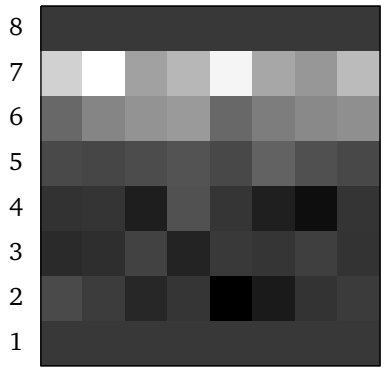
The reason for the upper area of the board being gray in the kings table (Figure 8.3.6) is again, that the king barely ever visited these squares. King safety is also a very important concept in Atomic Chess. Interestingly, the white and the black square in the diagram lie right next to each other: e1 is the safest square for the king, e2 the most dangerous one. The large difference between their (numerical) values illustrates that in Atomic Chess the king rarely ever moves in won games. In the experiment, this leads to the high value of square e1. If the game ever reaches a point where one player has to start moving his king around he usually loses the game. This leads to low or negative values respectively for the squares d2, e2, f2 which are visited most often by an escaping king.

8.3 Comparison of Learnt Values

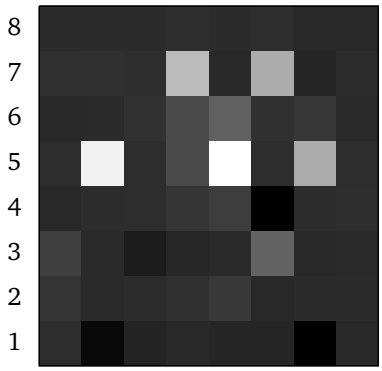
Figure 8.4 shows calculated average values of the piece-square values with their respective standard deviations and the learnt material values. Table 8.1 shows the final numeric values. Unfortunately, no suitable comparison values could be found for Atomic.

	Piece-Square Values	Piece Values
Pawn	1,95	2,91
Knight	**2,50**	**2,50**
Bishop	2,66	-0,77
Rook	5,19	6,37
Queen	13,98	24,77

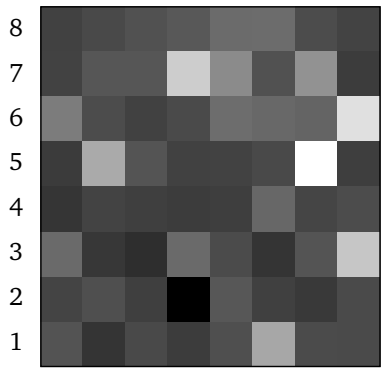
Table 8.1: Calculated and learnt material values in Atomic



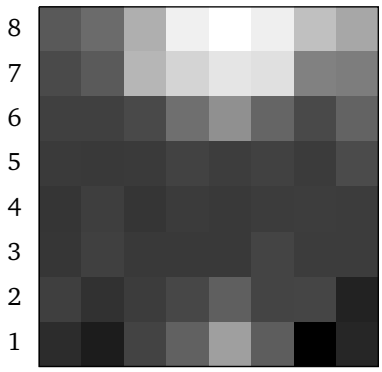
8.3.1: Pawn (Min/Max: 0,54/6,81)



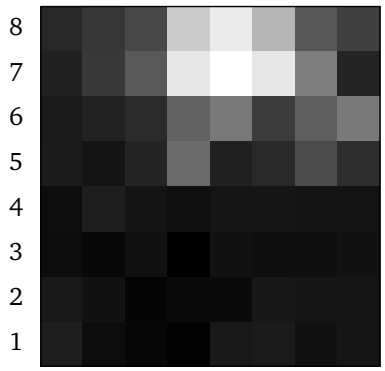
8.3.2: Knight (Min/Max: 1,74/5,67)



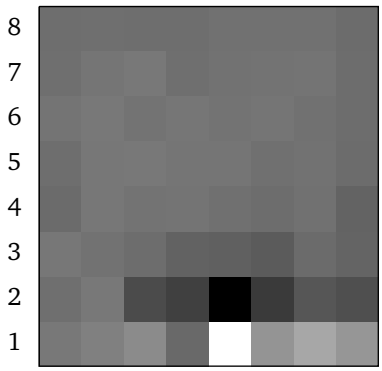
8.3.3: Bishop (Min/Max: 2,07/3,33)



8.3.4: Rook (Min/Max: 4,48/7,73)



8.3.5: Queen (Min/Max: 13,25/22,90)



8.3.6: King (Min/Max: -2,5/5,54)

Figure 8.3: Piece-Square tables in Atomic

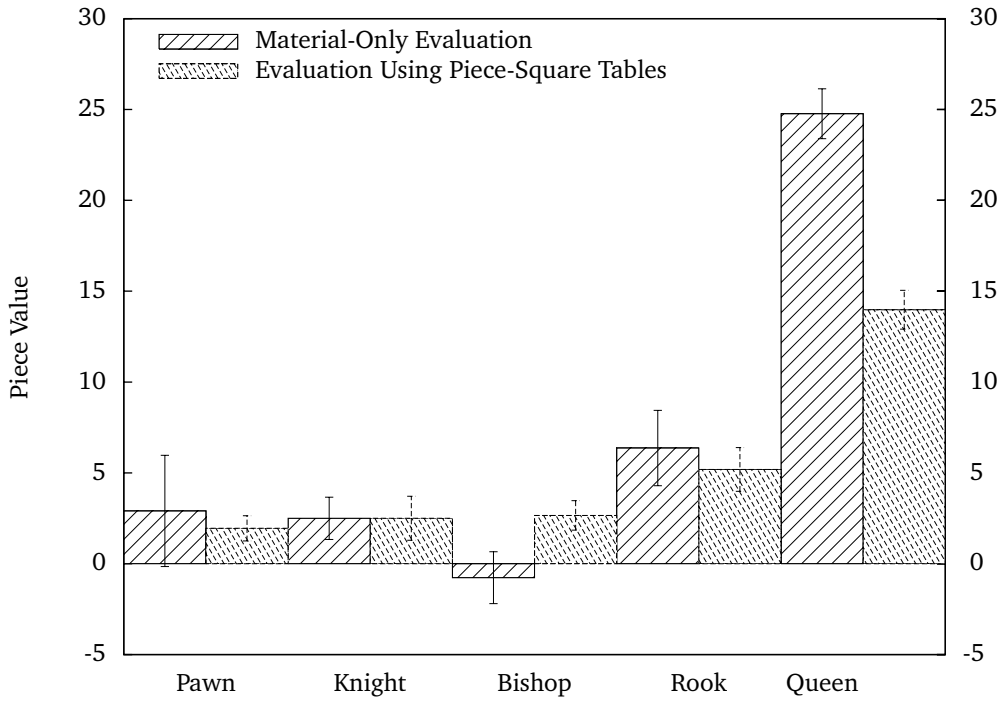


Figure 8.4: Final normalised material values of the pieces (Atomic)

Configuration	Won	Lost	Draw	Ratio
learnt vs. traditional	1231	753	16	61,95%
learnt vs. average p-sq	1186	797	17	59,73%
piece-square vs. learnt	1924	75	1	96,23%

Table 8.2: Results of the tournament (Atomic)

The results of the tournament shown in Table 8.2 lead to the same conclusion as the two previous sections. The learnt values induce a winning ratio of 61,95% meaning stronger playing and a better evaluation function. In this experiment, directly learning the piece values was more successful than averaging their piece-square values. The key difference between these two versions is the much higher relative value of the queen and the slightly negative value for the bishop. The very good performance of the piece-square tables confirms the relatively high importance of the absolute positions of pieces on the board.

9 Summary

The key result of our work are concrete values for the pieces of popular chess variants. Table 9.1 summarises the best-performing values for each variant, this time normalised so that a pawn has the (absolute) value 1. The first two lines contain values resulting from weighted averages of piece-square values, the last two have been directly learned.

variant	pawn	knight	bishop	rook	queen	king
standard	1.0	2.7	2.9	4.3	8.9	∞
crazyhouse	1.0	2.0	2.4	2.4	3.8	∞
suicide	-1.0	-1.4	-1.9	-2.1	-2.0	-1.3
atomic	1.0	0.9	0.0	2.2	8.5	∞

Table 9.1: Summary of the best-performing values, normalised to a pawn.

These values have to be interpreted with a bit of caution in the sense that we expect that minor optimisations could still improve them. For example, in Suicide Chess, the values resulting from averaging the piece-square tables were a bit different for most pieces (except the rook), but performed almost as well. On the other hand, for Atomic Chess, the alternative version that did not have a 0 for the bishop performed considerably worse. It is, nevertheless, quite counter-intuitive to attribute no importance to losing a bishop (the weight was, in fact, even slightly negative), but we believe this deserves further investigation.

More importantly, we must keep in mind that these values have been obtained based on material-only evaluation and it is, of course, not clear how these values will fare in the context of a fully fledged evaluation function. However, we are nevertheless quite confident that these values are reasonable, as can also be witnessed by the values we have obtained for standard chess, which are completely in line with standard chess theory (a bishop is a bit more than a knight, and both are typically a bit less than three pawns, etc.).

We have also obtained piece-square tables for each of these variants, which show for each piece how valuable it is if placed on a particular square. Not surprisingly, in all variants, the piece-square tables clearly outperformed a material-only evaluation function. However, in the Crazyhouse and Atomic, where tactics play a much stronger role than in standard chess, this performance gain was much more pronounced (the material-only version won less than 4% of the games, as opposed to more than 12% for Suicide and standard chess). In these games, the piece-value tables clearly showed the squares that are most likely to deliver a deadly attack to the uncastled king.

A possible point of criticism towards this approach is that the piece-square tables are static in the sense that they cannot adapt to new situations. For example, we have discussed that in Crazyhouse Chess, long castling is rare, and that the piece-square values reflect this by completely neglecting the queen side. This is good if we have an opponent that plays according to the same generally accepted principles, but it might be a major weakness if the opponent tries to exploit the program's weakness by a deliberately castling long-side, not because it is an objectively good move, but simply because it knows that the program prefers king-side squares no matter what. This is a valid point, but one that goes beyond this study. To solve this, one would need something like piece-square values that are relative to the position of the two kings. The problem can be found in regular chess as well (for a similar reason, e.g., chess programs use different piece-square-values for the king in the middlegame and the endgame), but it is somewhat more pronounced here, because the dynamics of the game depend strongly on the positions of the two kings. It is an open question how to address this problem adequately.

Another question that can be addressed in future work in this area is to learn entire evaluation functions for these chess variants. For example, a small experiment was conducted at the end of the project where a new 14-weight evaluation function for suicide was trained. The idea behind this evaluation was a very primitive recognition of situations that contain the potential of capture chains. The engine with this special evaluation played 2000 games against an engine with learnt piece-square tables and won almost 90% of those games.

However, in addition to the crucial question of the scalability of the techniques, the key problem here is to come up with useful features for an evaluation function. While research in computer chess has developed a large set of useful features that can be used in evaluation functions for standard chess, it is less clear what features should be used in the evaluation functions for these chess variants. In particular, static positional features will probably play a much smaller role in variants like Crazyhouse, which is dominated by tactics. In the ground-breaking work on his checkers player, Samuel (1959) already proposed techniques for automatically tuning the evaluation which pioneered later work on

reinforcement learning, such as the techniques that we used in this paper. He concluded his paper by making the point that the most promising road towards further improvements of his approach might be *"... to get the program to generate its own parameters for the evaluation polynomial"* instead of learning only weights for manually constructed features. Eight years later, in a follow-up paper, Samuel (1967) remarked that the goal of *"... getting the program to generate its own parameters remains as far in the future as it seemed to be in 1959."* Another forty years later, we still do not have much to add to that.

Bibliography

J. Baxter, A. Tridgell, and L. Weaver. A chess program that learns by combining TD(lambda) with game-tree search. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, pages 28–36, Madison, WI, 1998a. Morgan Kaufmann. 7

J. Baxter, A. Tridgell, and L. Weaver. Experiments in parameter learning using temporal differences. *International Computer Chess Association Journal*, 21(2):84–99, 1998b. 7

J. Baxter, A. Tridgell, and L. Weaver. Tdleaf(lambda): Combining temporal difference learning with game-tree search. *Australian Journal of Intelligent Information Processing Systems*, 5(1):39–43, 1998c. ISSN 1321-2133. 7

J. Baxter, A. Tridgell, and L. Weaver. Learning to play chess using temporal differences. *Machine Learning*, 40(3):243–263, September 2000. 7

J. Baxter, A. Tridgell, and L. Weaver. Reinforcement learning and chess. In Fürnkranz and Kubat (2001), chapter 5, pages 91–116. 7

D. F. Beal and M. C. Smith. Learning piece values using temporal differences. *International Computer Chess Association Journal*, 20(3):147–151, 1997. ISSN 0920-234X. 3, 7, 8, 10, 11

D. F. Beal and M. C. Smith. Learning piece-square values using temporal differences. *International Computer Chess Association Journal*, 22(4):223–235, Dec. 1999a. 3, 8, 10, 11

D. F. Beal and M. C. Smith. Temporal coherence and prediction decay in TD-learning. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 564–569. Morgan Kaufmann, 1999b. 7

D. F. Beal and M. C. Smith. Temporal difference learning applied to game playing and the results of application to Shogi. *Theoretical Computer Science*, 252(1-2):105–119, 2001. Special Issue on Papers from the Computers and Games 1998 Conference. 3

D. F. Beal and M. C. Smith. First results from using temporal difference learning in Shogi. In H. J. van den Herik and H. Iida, editors, *Proceedings of the First International Conference on Computers and Games (CG-98)*, pages 113–125, Tsukuba, Japan, 1998. Springer-Verlag. 7

L. Blackburn. *The Atomic Chess Book*. `http://virtual.parkland.edu/lblackburn/Atomic/atomic.html`. 23

D. M. Breuker, J. W. H. M. Uiterwijk, and H. J. van den Herik. Information in transposition tables. In H. van den Herik and J. Uiterwijk, editors, *Advances in Computer Chess 8*, pages 199–211. Universiteit Maastricht, Maastricht, 1997. ISBN 9062162347. 8

R.-J. Ekker. *Reinforcement Learning and Games*. Master's thesis, RijksUniversiteit Groningen, Groningen, The Netherlands, 2003. 7

J. Fürnkranz. Recent advances in machine learning and game playing. *ÖGAI Journal*, 26(2), 2007. URL Special Issue on Computer Game Playing. 7

J. Fürnkranz. Machine learning in games: A survey. In Fürnkranz and Kubat (2001), chapter 2, pages 11–59. 7

J. Fürnkranz. Machine learning in computer chess: The next generation. *International Computer Chess Association Journal*, 19(3):147–160, September 1996. 7

J. Fürnkranz and M. Kubat, editors. *Machines that Learn to Play Games*. Nova Science Publishers, Huntington, NY, 2001. 29

S. Gligoric. *Shall We Play Fischerandom Chess?* Batsford Chess Books, 2003. ISBN 978-0713487640. 3

H. Kaindl. Quiescence search in computer chess. *SIGART Newsletter*, 80:124–131, 1982. 8

D. E. Knuth and R. W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975. 8

D. N. Levy and M. Newborn. *How Computers Play Chess*. Computer Science Press, 1991. 8

D. B. Pritchard. *The Classified Encyclopedia of Chess Variants*. John Beasley, 2nd revised edition, 2007. ISBN 978-0955516801. 3

A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3 (3):211–229, 1959. 7, 27

A. L. Samuel. Some studies in machine learning using the game of checkers. II — recent progress. *IBM Journal of Research and Development*, 11(6):601–617, 1967. 28

S. S. Skiena. An overview of machine learning in computer chess. *International Computer Chess Association Journal*, 9 (1):20–28, 1986. 7

R. S. Sutton. Learning to predict by the methods of temporal differences. In *Machine Learning*, volume 3, pages 9–44, 40 Sylvan Road, Waltham, MA 02254, U.S.A., 1988. GTE Laboratories Incorporated, Kluwer Academic Publishers, Boston. 6

R. S. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998. 6

G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8:257–278, 1992. 6, 7

G. Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, March 1995. 7