A Moderately Successful Attempt to Train Chess Evaluation Functions of Different Strengths

Philip Paulsen PHILIP.PAULSEN@GMX.DE
Johannes Fürnkranz JUFFI@KE.TU-DARMSTADT.DE
TU Darmstadt, Knowledge Engineering Group, D-64289 Darmstadt, Germany

Abstract

In this paper, we report the results of experiments in which we trained four chess evaluation functions on games of weak chess players in four different rating groups, with the goal of reproducing computer players of that strength. Although the differences in playing strength between the players loosely correlates to the playing strength of the training data, this goal could not be achieved because the differences are much too small, and sometimes spurious. Nevertheless, the results are interesting because the learned functions exhibit a clear and systematic difference in some of the learned positional parameters (e.g., the importance of open ranks and lines), and show some unexpected but consistent differences to previous results (e.g., a much lower value for the queen).

1. Introduction

In strategy games, *opponent modeling* typically addresses the goal of forming a model of the opponent's playing strategies in order to be better able to exploit his or her weaknesses. Our motivation is different: we do not want to make the opponent stronger, we want it to play at the same skill level as the player. We think that this is the first study following this goal in the realm of strategy games. A brief discussion of related work in opponent modeling can be found in (Paulsen & Fürnkranz, 2010).

Commercial chess programs often have simple means of adjusting the playing strength by limiting the thinking time, the search depth, or by inserting random moves or blunders. We intend to follow a different path: we learn chess evaluation functions from games of players of different skill levels, from beginner's to amateur level, and compare the functions both qualitatively as well as in match play.

Appearing in *Proceedings of the ICML Workshop on Machine Learning and Games*.

2. The Chess Program

For our experiments, we used Tom Kerrigan's Simple Chess Program (TSCP) 1.81[1], which is particularly easy to use and modify. Its main focus is on conceptually clear programming, not on playing strength and speed, which were also not our primary objectives. It uses a standard alpha-beta negamax search with some quiescence search. According to two rating lists[2] it has a playing strength of approximately 1700, but it is unclear how this strength relates to the playing strengths of the players of the collected games, because computer ratings and human ratings develop in largely independent pools of players, so that their scales need not be comeasurable.

TSCP uses a simple linear evaluation function of the form

$$h(\mathbf{P}) = \sum_f w_f \cdot f(\mathbf{P}), \tag{1}$$

where $f(\mathbf{P}) \in \{0, 1\}$ are elementary feature values that are computed for a given chess position $\mathbf{P}$, and w_f are their weights. These weights will later be automatically tuned in the learning phase. For brevity, we omit the board $\mathbf{P}$ from the notation wherever it is clear from the context.

The features and weights used in TSCP are shown in Table 1. In addition, it uses the piece-square values tables, which are shown in the top row of Figure 2, indicating a bonus (or malus) for positioning the pieces at a certain square. As can be seen, the simple tables used in TSCP encode simple heuristics like "develop your pieces" (the original squares of the pieces are bright), "castle" (the target squares of the king after castling are dark, and the intermediate squares bright), or "center your pieces" (which is reasonable for the knight, but rather questionable for the bishop). There are no piece-square tables for rook and queen, and there is a separate table for the king in the endgame, where it typically becomes a very active piece.[3]

[1] `http://www.tckerrigan.com/Chess/TSCP`

[2] `http://www.computerchess.org.uk/ccrl/404/` and `http://wbec-ridderkerk.nl/` (edition 15)

[3] In TSCP, an endgame begins if the sum of the opponent's piece values (except pawns) drops below 1200.

Table 1: Features and features weights used in TSCP's evaluation function.

Piece Values		Positional Features	
		double pawn	-10
pawn	100	isolated pawn	-20
bishop	300	backward pawn	-8
knight	300	passed pawn	20
rook	500	open file	15
queen	900	half-open file	10
		rook on 7th rank	20

3. Training

The key idea for using a set of games for training an evaluation function is to interpret the move $\mathbf{m}_i(\mathbf{P})$ played by a player in a certain position $\mathbf{P}$ as a preference statement $\mathbf{m}_i \succ \mathbf{m}_j$ for all $j \neq i$, i.e., it is assumed that a player plays $\mathbf{m}_i$ because s/he prefers it over all alternatives $\mathbf{m}_j$. Moreover, we assume that a player prefers a move because s/he prefers the position resulting from this move over all positions that can be reached via alternative moves, i.e.,

$$\mathbf{m}_i \succ \mathbf{m}_j \Leftrightarrow \mathbf{P}_i \succ \mathbf{P}_j$$

where $\mathbf{P}_i = \mathbf{m}_i(\mathbf{P})$ is the position resulting from making move $\mathbf{m}_i$ in position $\mathbf{P}$

These preference statements $\mathbf{P}_i \succ \mathbf{P}_j$ are then used for training the ranking support vector machine of Joachims (2002).[4] Its key idea is to reinterpret the preference statements as constraints on the evaluation function, i.e.,

$$\mathbf{P}_i \succ \mathbf{P}_j \Leftrightarrow h(\mathbf{P}_i) > h(\mathbf{P}_j).$$

As the function h is linear, the latter part is equivalent to

$$h(\mathbf{P}_i - \mathbf{P}_j) \quad = \quad \sum_f w_f \cdot (f(\mathbf{P}_i) - f(\mathbf{P}_j)) > 0$$

Thus, essentially, the training of the ranking SVM corresponds to the training of a classification SVM on the pairwise differences $\mathbf{P}_i - \mathbf{P}_j$ between positions $\mathbf{P}_i$ and $\mathbf{P}_j$.

A remaining problem is how to integrate the learning into the search algorithm. In particular, we have to face the problem that the characteristics of the position at the root node of the search are often completely different from the characteristics of the positions at the leaf, which are those that are actually evaluated with the evaluation function h. Thus, we do not want to adapt the evaluation of the root position, but the evaluation of the so-called *dominant position* of the search, i.e., the leaf position in the search tree from which the evaluation has been propagated back to the root of the search tree. To this end, each of the positions $\mathbf{P}_i$ in the training data is encoded with the features of the position $d(\mathbf{P}_i)$ encountered in the principal variation starting with $\mathbf{m}(\mathbf{P}_i)$. This problem has already been recognized and solved by Samuel (1959) but seemed to have been forgotten later on. It was rediscovered independently in the context of reinforcement learning by (Beal & Smith, 1997) and (Baxter et al., 1998).

[4] Available from `http://svmlight.joachims.org`. We used SVM^{light} in preference mode, i.e., `svm_learn -z p`. The optimized and faster SVM^{rank} was not yet available at the time of the experiments.

Table 2: Characteristics of the training data.

Elo	# games	# positions	# examples
1000	3.000	20.732	614.246
1200	5.499	39.170	1.185.524
1400	10.499	75.196	2.325.861
1600	10.500	75.114	2.352.536

4. Data

For training evaluation functions with different strengths, we collected game protocols of players with approximately 1000, 1200, 1400, and 1600 Elo rating points.[5] The choice of these rating ranges was primarily motivated by the fact that we intended to weaken TSCP, whose strength was listed with 1700 rating points (but on a separate computer rating, which may not be entirely comparable to the human rating pool).

The games were selected from a 3.5 million game database,[6] so that the white player had one of the four ratings above, with a tolerance of $\pm$ 25 points. From each game, we extracted all positions that occurred after intervals of 10 plies (5 moves). For each such position, we trained the ranking SVM so that it will prefer the position after the played move over all alternative positions. On average, we had about 29 alternatives (negative examples) for each position (positive example). Table 2 shows the number of games, position, and generated training examples. Note that we imposed an upper limit on the number of games for playing strength 1600. Each training position was characterized with 332 features, as described above.

[5] Elo ratings are a commonly used rating scale for chess players. Roughly, players with 1000 Elo points are beginners, with 1600 average club players, with 2000 strong club players, with 2400 grandmasters, and with 2800 world champion. The rating scale is designed so that a 200 points difference roughly corresponds to a winning probability of 0.75 for the stronger player (Elo, 1978).

[6] Available from `http://chessdb.sourceforge.net/`

Table 3: Run-time and $\xi\alpha$-estimates of error, recall, and precision of the trained SVMs.

Elo	run-time	error	recall	precision
1000	24,098	34.46%	66.96%	72.42%
1200	17,251	33.26%	67.84%	73.30%
1400	87,603	33.55%	67.28%	72.74%
1600	86,638	33.40%	67.18%	72.89%

5. Results

5.1. Move Prediction

Table 3 shows the results of a $\xi\alpha$-estimation of the predictive performance of the trained SVMs.[7] It seems that the correct move can be predicted reasonably well for all different training sets. About $2/3$ of the correct moves (moves that were actually made and thus labeled as positive) are correctly predicted as having a positive label, and about $3/4$ of the predictions for a positive label are correct.

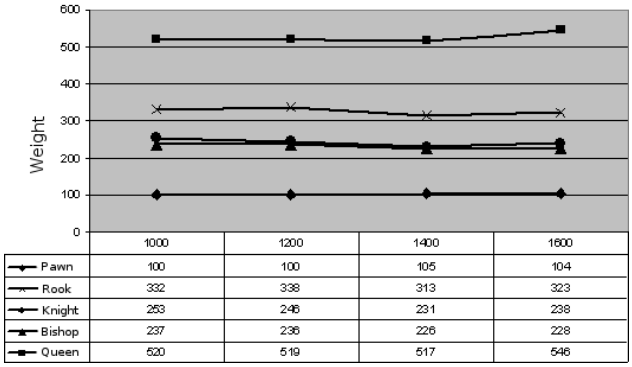
5.2. Piece Values

Figure 1 shows the learned feature weights for the four learned evaluation functions. Before we look at the results in more detail, we note that in general, the very low variance between the values of the four independently trained functions is quite encouraging and supports the view that the observed regularities and differences are not due to chance phenomena.

For some of the pieces, the learned values are quite consistent with those found in (Beal & Smith, 2000) and (Droste & Fürnkranz, 2008). For example, we can see in Figure 1(a) that a bishop is only worth about $2\frac{1}{2}$ pawns (usually 3), and a rook equals only a pawn and a bishop (usually 2P + 1B). This correspondence to previous results is not self-evident, because the values obtained in these publications have been learned from different data (strong master vs. weak amateur games) and with an entirely different learning algorithm (reinforcement vs. supervised learning).

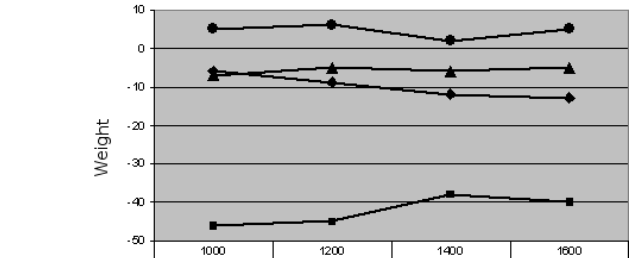
However, other results differ a bit from common chess knowledge. For example, a bishop is generally found to have a slightly higher value than a knight (see, e.g., Fischer et al., 1972), while in our results, knights were consistently learned to have a somewhat higher value in all four playing strengths. Although the difference is rather small (approximately 10 points on average), its consistent appearance lets us assume that this is systematic, and that weaker players tend to give a somewhat higher value to knights, or at least are more willing to trade bishops for knights.

[7] $\xi\alpha$-error is an upper bound on the leave-one out cross-validation error, but has the advantage that it can be easily computed from a trained SVM without the need for performing additional training via cross-validation (Joachims, 2000).



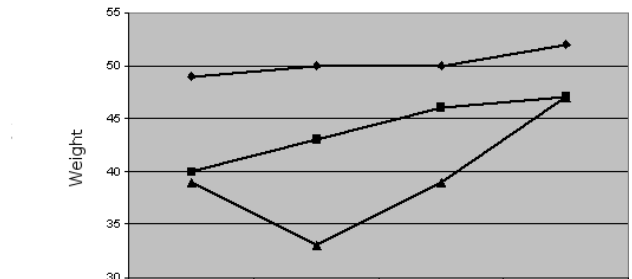
	1000	1200	1400	1600
Pawn	100	100	105	104
Rook	332	338	313	323
Knight	253	246	231	238
Bishop	237	236	226	228
Queen	520	519	517	546

(a) piece values



	1000	1200	1400	1600
double pawn	-46	-45	-38	-40
isolated pawn	-6	-9	-12	-13
backward pawn	-7	-5	-6	-5
passed pawn	5	6	2	5

(b) pawn evaluation



	1000	1200	1400	1600
Open File	49	50	50	52
Half-Open File	40	43	46	47
R on 7th Rank	39	33	39	47

(c) rook evaluation

Figure 1: Development of evaluation function weights

Noticeable are also the comparably low values for rook and queen. For example, a rook is typically worth 5 pawns, while we here observe only about 3.3 pawns. Interestingly, this has also been observed in (Droste & Fürnkranz, 2008; Table 1), where a rook is worth about 3.6 pawns. More striking is the low value of the queen with slightly more than 5 pawns, which is unprecedented. This can also not been explained with a possibly too high value for the pawn, because the queen's value is also degraded in comparison to the other pieces. It is unclear how to interpret this deviation, but our best guess is that games of weak chess players

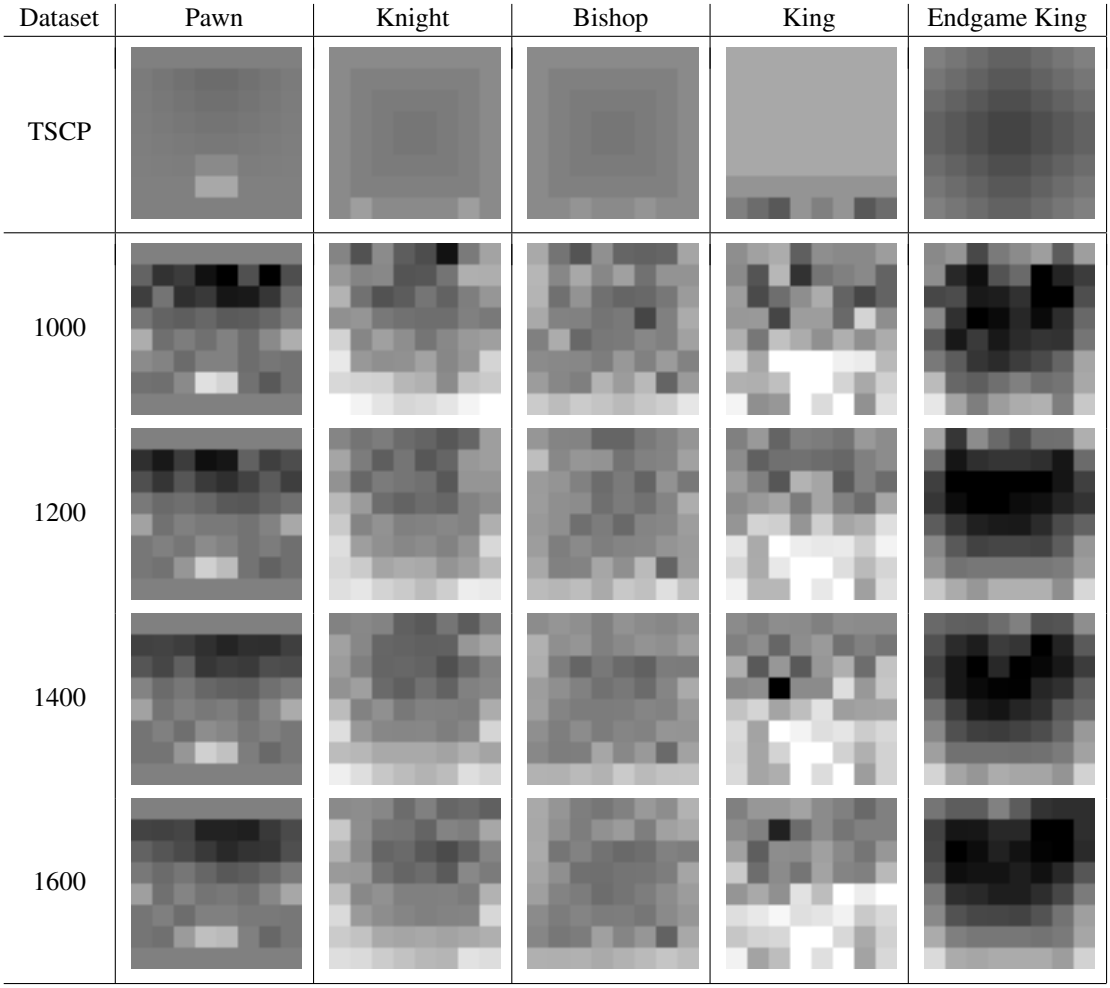


Figure 2: Original piece-square value tables of TSCP and those learned for four different playing strengths. Dark squares indicate a high positive value for a piece at this square, bright squares are high negative, and medium gray is neutral.

are much more erratic than games of strong chess players. While the advantage of a minor piece is typically more than enough to guarantee victory among stronger players, weak players have a much higher probability of blundering even after a strong advantage, like the win of a queen. Maybe this is reflected in these learned piece values. The slight increase of the queen's value at the transition from 1400 to 1600 is consistent with that hypothesis, but additional evidence is needed to back up this hypothesis.

5.3. Positional Features

Figure 1(b) shows the evolution of the pawn-related features. We first observe, encouragingly, that the sign has always been learned correctly, i.e., good (passed pawns) and bad features (double, isolated, and backward pawns) have been recognized as such. The most striking difference to the manually set values of Table 1 is the much higher (absolute) value of a double pawn, which is consistent with the heuristic "avoid double pawns", which is typically highly over-rated by weaker players. Most interesting are the results on the evolution on the rook-related features (Figure 1(c)). In all three cases, the importance of these positional features, which together essentially encode the rooks' mobility and their potential to invade the enemy camp, show a consistent increase in importance. This result is quite plausible and it is tempting to assume that this reflects the increased playing strength of the players.

However, it is not entirely clear, whether common misconceptions of weaker players (e.g., over-rating double pawns, under-rating the importance of free lines for the rooks) can be found in the game traces. This would imply that, if a certain feature is deemed overly important by a player, it actually receives a higher importance in game play. While such self-fulfilling prophecies are not entirely implausible (one can, e.g., imagine that a weak player tries harder to avoid a double pawn than a stronger player, so that he will only get

a double pawns in much weaker positions, and their influence on the outcome is, indeed, higher), we have to keep in mind that this is all speculation. An alternative interpretation is simply that as playing strength increases, positional features like open files have a stronger and stronger influence on the final outcome of the game, whereas the results of weaker players are primarily dominated by tactical blunders such as losing a piece.

5.4. Piece-Square Tables

Figure 2 shows a graphical visualization of the learned piece-square tables for the four playing strengths.[8] Some of the values appear to have a high variance, which happens primarily for piece-square combinations that rarely occur in practical play. For example, most of the king's positions only occur a few times in the training data, which may cause a few erratic values like the high value for the king at c5 for the 1400 rating function. However, most of the values are perfectly sensible, and one can clearly recognize typical chess heuristics such as "move the center pawns in the opening", "advanced pawns are good" "a knight on the rim is grim", the bishop fianchetto on g2, the king in the center is dangerous in the opening (castle quickly!) but essential in the endgame. Overall, these values are quite reasonable, but, again, no clear patterns can be detected over an increase in playing strength.

5.5. Match Play

While the above qualitative results looked quite encouraging, the final test, of course, has to be to see how the four different evaluation functions fare in competition. For performing round robin tournaments between the players we used the Galis Winboard Tournament Manager.[9] We played three series of 100 games between each pair of opponents, each series for search depths 1, 2, and 4 ply (plus quiescence search). For search depths 2 and 4 we also included the original TSCP program into the comparison, making up for a total of $300 + 400 + 400 = 1,100$ games for each of the four variants. Table 4 shows the results.

For a 1-ply search, the results are dominated by a large number of draws, which were caused by frequent three-move repetitions. We had expected, that for this search depth, the advantage of a better heuristic should be most apparent. The expected effect then showed, to some extent, for a 2-ply search. Here we can clearly observe a decrease in playing strength that correlates with our expectations, except that no difference could be observed for the evaluation functions at the 1200 and 1400 level, and that there are some minor outliers in the pairwise comparisons (e.g., the 1400 evaluation functions scores worse than the 1200 function against both 1000 and 1600). However, the differences in playing strength are much too small in comparison to the differences in playing strength from the original games. A difference of 200 Elo rating points should correspond to a winning percentage of roughly 75%, whereas even 1600 wins somewhat less than $2/3$ of its games against 1000, which corresponds to a difference in playing strength of approximately 100 rating points. For 4-ply search, the results are quite irregular again, with all functions from 1200 to 1600 performing roughly at the same level, with only the 1000 evaluation function being clearly inferior.

In summary, we can observe differences in the learned functions which correlate roughly with our expectations, but the magnitude of the differences is considerably smaller than we hoped for.

Table 5: Results of play on the FICS chess server under varying time controls and search depths. The last column shows the average rating difference to the 1000 player.

	1 ply		flexible depth		avg.
	Blitz	Stnd	Blitz	Stnd	$\pm$
1000	1088	1348	1839	2003	0.00
1200	1143	1383	1874	2075	+49.25
1400	1122	1397	1891	2043	+43.75
1600	1105	1398	1898	2029	+38.00

5.6. Internet Play

As a final test, we let the four players loose on the free internet chess server FICS,[10] where they can meet competition of varying strengths, primarily human but also computers. All programs played in two different modes, blitz (time controls 3 minutes or 5 minutes for an entire game, or 2 minutes with a 12 seconds increment per move) and standard (15 or 20 minutes for the game). In both cases we played with a search depth limitation to 1 ply and without such a limitation (the depth was then chosen dynamically to fit the chosen time controls). Thus, each evaluation function was run in four different configurations.

Table 5 shows the results after about 1000 Blitz games and ca. 500 Standard games for each search depth. The results essentially confirm the findings of the previous section: only the 1000 player is reliably worse than the others, whereas the differences between the other three are spurious, and within the regular variance of chess ratings.[11]

[8] The exact numerical values can be found in the appendix of (Paulsen, 2009).

[9] `http://wbec-ridderkerk.nl/html/downloada/Galis.html`

[10] `http://www.freechess.org/`

[11] Ratings on chess servers are updated incrementally after each game. Typical rating changes after a won game are 0 to 15 points, depending on the strength of the opponent.

Table 4: Results of the four trained functions against each other and against TSCP's functions. The winner is shown in the columns (e.g., 1600 scores 63.5 points against 1000 in the 2-ply setting).

	1 ply				**2 ply**				**4 ply**			
	1000	1200	1400	1600	1000	1200	1400	1600	1000	1200	1400	1600
1000	—	49.5	50.5	45.0	—	59.0	56.0	63.5	—	68.5	58.0	53.0
1200	50.5	—	54.0	53.0	41.0	—	54.0	57.0	31.5	—	46.5	61.5
1400	49.5	46.0	—	50.0	44.0	46.0	—	61.0	42.0	53.5	—	54.0
1600	55.0	47.0	50.0	—	36.5	43.0	39.0	—	47.0	38.5	46.0	—
TSCP	—	—	—	—	36.5	37.0	36.5	43.5	22.0	35.5	38.0	26.5
$\sum$	**155.0**	**142.5**	**154.5**	**148.0**	**158.0**	**185.0**	**185.5**	**225.0**	**142.5**	**196.0**	**188.5**	**195.0**
$+$	35	12	18	19	130	171	167	210	121	174	171	173
$=$	240	261	273	258	56	28	37	30	43	44	35	44
$-$	25	27	9	23	214	201	196	160	236	182	194	183

6. Conclusions

In this paper we reported the results of a study that aimed at adjusting the parameters of a chess evaluation function to the skill level of a class of human players in a database of games. In the light of the results of match play among the learned players and the results against a variety of players on the free internet chess server, we have to conclude that this goal could not be achieved.

Nevertheless, we have also seen evidence that the learned functions were, indeed, of different strengths, albeit at a much smaller scale. Larger performance differences could only be observed by variations in the search depths, not by minor variations in the evaluation function. In retrospect, this result is not entirely unexpected, because fine-tuning an evaluation function of a chess program is a difficult, time-consuming process with small gains. In a sense, we have now complemented this knowledge with the experience that one can also not do much harm, even if one deliberately tries to construct a weak (but still sensible) evaluation function.

As a side result of this study, we also note that supervised training of ranking SVMs on game playing data seems to work. Even though some aspects of the learned evaluation functions looked surprising (e.g., the low value of the queen), the low variance in the four learned functions reassures us that this is not due to chance. We speculated about possible reasons for these unexpected results, but further studies are necessary to obtain a definite answer.

Acknowledgments

We would like to thank the people whose freely available software we used, most notably Thorsten Joachims for SVM-light, Tom Kerrigan for TSCP, Shane Hudson and David Kirkby for Scid/ChessDB, Tim Mann for XBoard and WinBoard, Nikola Bacic for the Galis Winboard Tournament Manager, Leo Dijksman for the WBEC Ridderkerk Web-site, and the maintainers of FICS for countless hours of serious fun.

References

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

Beal, D. F., & Smith, M. C. (1997). Learning piece values using temporal difference learning. *International Computer Chess Association Journal*, *20*, 147–151.

Beal, D. F., & Smith, M. C. (2000). Temporal difference learning for heuristic search and game playing. *Information Sciences*, *122*, 3–21. Special Issue on Heuristic Search and Computer Game Playing.

Droste, S., & Fürnkranz, J. (2008). Learning the piece values for three chess variants. *International Computer Games Association Journal*, *31*, 209–233.

Elo, A. E. (1978). *The rating of chessplayers, past and present*. New York: Arco. 2nd edition.

Fischer, R. J., Margulies, S., & Mosenfelder, D. (1972). *Bobby Fischer teaches chess*. Bantam Books.

Joachims, T. (2000). Estimating the generalization performance of an SVM efficiently. *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)* (pp. 431–438). Stanford University, Stanford, CA: Morgan Kaufmann.

Joachims, T. (2002). Optimizing search engines using clickthrough data. *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-02)* (pp. 133–142). ACM Press.

Paulsen, P. (2009). Lernen unterschiedlich starker Bewertungsfunktionen aus Schach-Spielprotokollen. Master's thesis, TU Darmstadt, Knowledge Engineering Group. Diplomarbeit.

Paulsen, P., & Fürnkranz, J. (2010). *A moderately successful attempt to train evaluation functions of different strengths* (Technical Report TUD-KE-2010-07). TU Darmstadt, Knowledge Engineering Group.

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