

Visualization and Adjustment of Evaluation Functions Based on Evaluation Values and Win Probability

Shogo Takeuchi

Department of Graphics and Computer Sciences, the University of Tokyo, Japan
{takeuchi,kaneko,yamaguch}@graco.c.u-tokyo.ac.jp

Tomoyuki Kaneko

Kazunori Yamaguchi

Satoru Kawai

The University of the Air
kawai@acm.org

Abstract

We present a method of visualizing and adjusting the evaluation functions in game programming in this paper. It is widely recognized that an evaluation function should assign a higher *evaluation value* to a *position* with greater probability of a win. However, this relation has not been utilized directly to tune evaluation functions because of the difficulty of measuring the probability of wins in deterministic games. We present the use of win percentage to utilize this relation in positions having the same evaluation value as win probability, where the positions we used were stored in a large database of game records. We introduce an *evaluation curve* formed by evaluation values and win probabilities, to enable evaluation functions to be visualized. We observed that evaluation curves form a sigmoid in various kinds of games and that these curves may split depending on the properties of positions. Because such splits indicate that an evaluation function that is visualized misestimates positions with less probability of winning, we can improve this by fitting evaluation curves to one. Our experiments with Chess and Shogi revealed that deficiencies in evaluation functions could be successfully visualized, and that improvements by automatically adjusting their weights were confirmed by self-plays.

Introduction

The most successful approach in game programming has been game tree searches with the assistance of evaluation functions (Schaeffer 2000). An evaluation function with this approach should yield an *evaluation value* as an estimate of the win probability for a given *position*. A popular way of constructing an evaluation function is to make it a (linear) combination of evaluation primitives called *features*, and adjust the weights of the combination.

However, it is difficult, for computers and people, to find appropriate sets of features and their weights. As a result, strong programs for many games including Chess still use manually tuned evaluation functions. An individual can use self-play with two evaluation functions to determine which function is better. However, it is very time consuming to obtain statistically significant results with self-play.

We propose a novel method of testing whether an existing evaluation function can be improved by incorporating a

new feature into it, as well as a method of tuning its weight, by only using game records and the winner for each position. The idea is to utilize the winning percentage in positions with the same evaluation value as win probability, where the positions we use are stored in a large database of game records. Calculating the winning percentage is computationally inexpensive compared to self-play and to feature selection based on statistics (Guyon & Elisseeff 2003).

Win probabilities plotted against evaluation values form a sigmoid curve, and we call these *evaluation curves*. They may split depending on the properties of positions (e.g., whether the King is safe or not), and such splits indicate that the evaluation function that is visualized needs a new feature related to the properties. This visualization was confirmed to work well in our experiments on Chess, Othello, Go, and Shogi. We can therefore improve evaluation functions by incorporating such features and by adjusting their weights, so that multiple curves will fit together as closely as possible. Our experiments with self-play in Chess and Shogi revealed that evaluation functions with split curves were actually weak, and that automated adjustment successfully remedied this problem.

The paper is structured as follows. The next section reviews related research and a new method of visualizing and adjusting evaluation functions is then presented, followed by the experimental results. Finally, conclusions are drawn and future work is discussed.

Related Work

Much research has been devoted to the learning of evaluation functions in game programming since Samuel's seminal work on Checkers (Samuel 1959). Supervised learning can be effectively used to adjust weights, when appropriately labeled training positions are available. Supervised learning in Othello produced one of the strongest programs then available (Buro 1998). However, no evaluation functions have successfully been tuned in Chess and Shogi by directly applying supervised learning, due to the difficulty of mechanically labeling the positions. There is a method based on the correlation of preferences for positions in Chess (Gomboc, Marsland, & Buro 2003). However, this requires many positions to be assessed by grandmasters to determine which are preferred. Thus, its application is limited to domains in which such assessments can be done. Our method requires

no positions to be labeled except for the winners.

Temporal difference learning is another approach to adjusting weights and has been successful with Backgammon (Tesauro 1995). Learning variations have also been applied to Chess (Baxter, Tridgell, & Weaver 2000). However, temporal difference learning has not been adopted in top-level programs for deterministic games. This also involves additional computational cost to update the weights done by playing numerous games. Our method requires no need for extra play and is computationally efficient.

Programs in Go strengthened by Monte Carlo methods have recently been dramatically improving (Bouzy & Helmstetter 2003; Kocsis & Szepesvari 2006). Although Monte Carlo sampling may be useful for estimating the win probability in other games, it obviously requires vastly more computation than our method.

Visualization and Adjustment

We can improve the existing evaluation function with our approach in the following way: (1) Draw evaluation curves for positions with various properties. If multiple curves appear (e.g., Fig. 1), they indicate that new features related to the properties are needed. (2) Prepare a new evaluation function with the new feature and draw the evaluation curves again. Improvement is accomplished if they fit into one curve. The weights of newly incorporated features can be automatically adjusted, as will be explained in the latter half of this section.

Visualization of Evaluation Functions

We will first discuss how evaluation curves reveal a problem with the evaluation function. The main idea is established on the principle that a good evaluation function should assign the same value to positions with the same win probability. Let us look at how well this holds by plotting the relation in graph form. The relation of win probabilities and evaluation values is plotted with evaluation values along the horizontal axis in Fig. 1 and the win probabilities along the vertical. Of course, the evaluation curve of a good evaluation function must be monotonously increasing. However, this is not sufficient to ensure that an evaluation function is sound.

Assume that we have an evaluation function only consisting of a material balance in Chess, and that we separately plot two evaluation curves first for all positions and then only for positions where the opponent’s King is threatened. The plotted evaluation curves will be split as seen in Fig. 1. The solid curve is for all positions and the dotted curve is for positions where the opponent’s King is unsafe. The reason for the split is that if two positions have a similar material balance, the position where the opponent’s King is unsafe should have a greater probability of a win for his rival. Evaluation values are not reliable if there are such splits. For example, assume that there are two positions X and Y, and that position X is at B and Y is at A in the figure. Then, the evaluation function incorrectly assigns a higher evaluation value to Y even though X has a greater probability of a win. Evaluation functions may generally assign values to different scales depending on the conditions, and these values on different scales are not comparable. We therefore propose

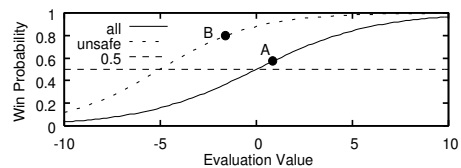


Figure 1: Example of a poor evaluation function

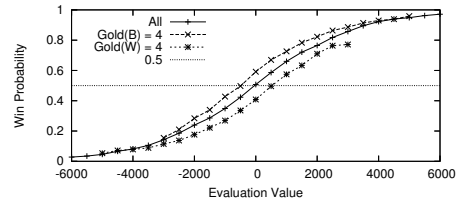


Figure 2: Evaluation curves in Shogi (with four Golds)

that evaluation curves be plotted under various conditions and then checked whether they split or not. We call a concrete proposition on a property of positions a *condition*. In the above example, the threatened king is a property, and a sample condition of the property is whether $\#attackers > 3$.

Evaluation Curves with Game Records We need the win probability for all evaluation values to be able to implement our method. We approximate this with the winning percentage because of the difficulty of measuring the win probability in a deterministic game. Assume that there are numerous game records, R , that contain unbiased positions. Utilizing R , we define the winning percentage as a function of evaluation value v and R as

$$\text{Win probability}(v, R) = \frac{|W_v(R)|}{|W_v(R)| + |L_v(R)|}, \quad (1)$$

where

$$\begin{aligned} P_v(R) &= \{p \in R | v - \frac{\delta}{2} \leq \text{eval}(p) < v + \frac{\delta}{2}\}, \\ W_v(R) &= \{p \in P_v(R) | \text{winner}(p) \text{ is the first player}\}, \\ L_v(R) &= \{p \in P_v(R) | \text{winner}(p) \text{ is the second player}\}. \end{aligned} \quad (2)$$

Here, p is a position in R and δ is a non-negative constant standing for an interval, whose appropriate value depends on the number of positions used and on the range of evaluation values. We first compute the evaluation value for each position in the game records to compute this win probability. We also determine the winner of all positions. Although it is usually difficult to determine the theoretical winner of a position, we used that of a game record as the winner of all positions that appeared in the record. This worked sufficiently well in our experience. Finally, we aggregate the numbers of wins $|W_v|$ and losses $|L_v|$ for each interval $[v - \frac{\delta}{2}, v + \frac{\delta}{2})$, and calculate the fraction using Eq. (1). It is occasionally better to use values returned by a shallow quiescence search in practice instead of a raw evaluation value in Eq. (2). This depends on the characteristics of particular games and details on these are discussed in the next section.

We call an evaluation curve using all positions a *total curve*. We call an evaluation curve using part of the positions for which a condition holds a *conditioned curve*. How well the evaluation function is working under the conditions can be found by comparing the total curve and conditioned curves. For example, the solid curve in Fig. 2 indicates the total evaluation curve obtained for Shogi (details explained in the next section). The dotted (broken) curve is a conditioned curve for the positions where the first (second) player monopolizes four Golds.¹ The “B” in the figures in this paper denotes the first player and the “W” the second player. We can actually observe a gap between the total curve and conditioned curve in this figure under both conditions. Thus, this evaluation function should incorporate a feature representing the monopolization of Gold.

Improving Evaluation Functions

Once we have found a condition whose conditioned curves deviate from the total curve, we can design a new evaluation function incorporating a new feature representing the condition. Let p be a position, $e(p)$ be an old evaluation function, and $e'(p)$ be a new evaluation function with a vector of weights, \mathbf{w} . The new evaluation function with a new feature, f , would be $e'(p) = e(p) + w_1 f(p)$, where the output of the new feature, $f(p)$, is 1 (-1) when the first (second) player has four Golds in p and 0 otherwise for the previous example of Shogi. See Eq. (3) and (4) of our experiments, for more complex modifications to the evaluation function of GPS Shogi.

We then need to carry out supervised learning to adjust the newly incorporated weights. However, we do not have appropriate evaluation values for positions used as training examples. We therefore adjust the weights by optimizing the prediction of win probability for the positions, which only requires game records. Here, we introduce two methods, i.e., **MLM** and **LS**.

Because evaluation curves form a sigmoid as has been confirmed by numerous experiments that will be discussed later, it is acceptable to use logistic regression which maximizes the likelihood of training examples (**MLM**). Let $g(p)$ be the win probability of a position approximated by the sigmoid transformation of $e'(p)$: $g(p) = \frac{1}{1 + \exp(-w_0 \cdot e'(p))}$. The likelihood for a training position, p_i , is defined as

$$\text{likelihood}(p_i, y_i) = g(p_i)^{y_i} (1 - g(p_i))^{(1-y_i)},$$

where y_i denotes the winner of the i -th training position whose value is 1 (0) if the winner is the first (second) player. Finally, weights $\hat{\mathbf{w}}$ are determined so that the product of the likelihood of each position is maximized:

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \prod_i \text{likelihood}(p_i, y_i).$$

As an alternative, weights can be determined with least squares (**LS**) by minimizing the summation of the squared

¹Because a Gold in Shogi is an important piece, it is empirically known that the win probability of a player having all four Golds tends to be higher than that usually predicted by a material balance. Note that captured pieces can be reused in Shogi.

errors between y_i and $g(p_i)$:

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \sum_i (y_i - g(p_i))^2.$$

Experimental Results

Let us first discuss how effective visualization was in the experiments we did on Chess, Othello, Go, and Shogi, and then the quality of evaluation functions adjusted with our method of self-play in Chess and Shogi.

Game Programs and Game Records We will first explain the game programs and records we used in our experiments. We used one of the best programs for each game to ensure the experiments were meaningful and valid. We also chose open source programs so that we could modify their evaluation functions.

Chess: We worked with Crafty² version 20.14. We used 45,955 positions made available by the International Correspondence Chess Federation (ICCF³) as the game records. We did not use records of draws to avoid complications with determining the probabilities of wins.

Othello: We used Zebra⁴ which is a top-level Othello program. We used 100,000 positions played at GGS⁵ as the game records.

Go: We used GNU Go⁶ version 3.7.4 and 56,896 records played on a nine-by-nine board at KGS.

Shogi: We used GPS Shogi⁷, which was a finalist at the world computer shogi championship in 2005. We used 90,000 positions from the Shogi Club 24 as the game records. We employed a checkmate search for Shogi in up to 10,000 nodes for each position, from the first position to the last in a record to determine the winner of each record. If a checkmate was found, the player for the position was determined to have won.

Evaluation Curves in Various Games

Here, we will present the evaluation curves for four games and discuss practical issues. We omitted intervals that consisted of fewer than 1,000 positions for all evaluation curves.

Figure 3 (left) plots the evaluation curves for Chess. We focused on “King Evaluation” (*KE*), which is a feature used in Crafty that estimates how safe the King is. The conditions used were $KE \geq 50$ or $KE \leq -50$. We used two evaluation functions. The first was the original evaluation function for Crafty, and the second was a modified one whose *KE* was intentionally turned off. We can see that the conditioned curves with the turned-off version (plotted with black and white squares) are vastly different from the total curve, and that conditioned curves with the original version (plotted with crosses and asterisks) are closer to the total curve. The conditioned curves are also still not that close to the total curve in the original version. This is notable around a

²[ftp://ftp.cis.uab.edu/pub/hyatt/](http://ftp.cis.uab.edu/pub/hyatt/)

³<http://iccf.com/>

⁴<http://radagast.se/othello/>

⁵<http://www.cs.ualberta.ca/~mburo/GGS/>

⁶<http://www.gnu.org/software/gnugo/gnugo.html>

⁷<http://gps.tanaka.ecc.u-tokyo.ac.jp/{gpsshogi,osl}/> (rev.2602)

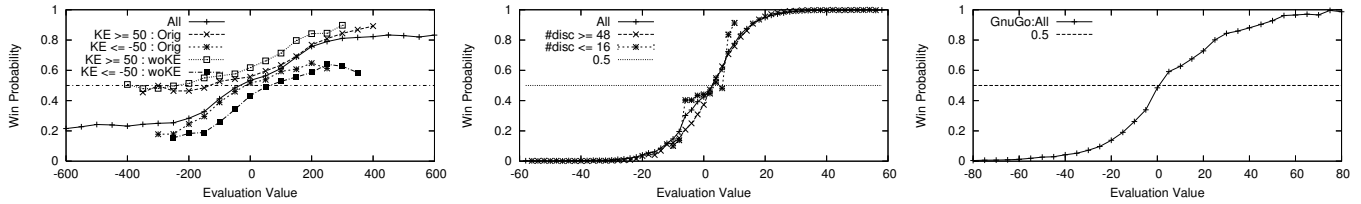


Figure 3: Evaluation curves (left: Chess, with quiescence search, $KE \geq 50$, center: Othello, number of stones, right: Go)

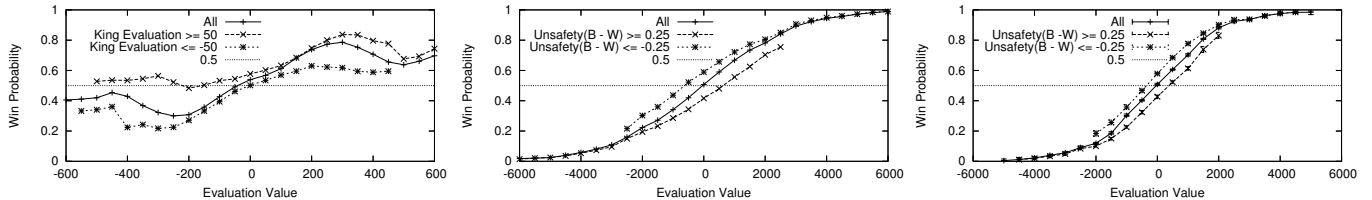


Figure 4: Evaluation curves (left: Chess without quiescence search, $KE \geq 50$, center: Shogi without quiescence search, difference in KUs , right: Shogi with quiescence search)

Table 1: Results of adjusting weights in Chess

Method	MLM	LS	Original
Bishop Evaluation	1.75	1.33	(1.00)

win probability of 0.5 and the evaluation values are less than 100. This means that there is a great deal of room to improve the original evaluation function used in Crafty. The graph suggests that an evaluation function should be so non-linear that it returns at least -100 for positions where $KE \geq 50$. We used values returned by a quiescence search with depth 1 in Chess, instead of the raw evaluation values in Eq. (2). We will discuss the details later.

Figure 3 (center) plots the evaluation curves for Othello. The broken (dotted) curve is an evaluation curve for positions that have more (less) than 47 (17) stones. Note that around a win probability of 0.4, the gap between the two curves amounts to 7, which is not at all negligible.

All the graphs in Fig. 2 and 3 confirm that evaluation curves are sigmoid in various games including Go (see Fig. 3, right). We also found conditioned curves deviated from total curves under various conditions in Shogi.

Importance of Quiescence Searches Most programs in various games including Chess use quiescence searches because evaluation values are unreliable for tactical positions. We used evaluation values as in game tree searches for the leaves of principal variations obtained by a quiescence search with depth 1 to draw Fig. 3 (left). Note that the curves are monotonously increasing in the figure. Fig. 4 (left) shows an evaluation curve in Chess without a quiescence search, while the other configurations are the same in the two figures. Here, the curves are, surprisingly, not monotonously increasing. A comparison of both graphs suggests that these fluctuations are caused by unreliable evaluations of tactical positions.

Quiescence searches, in contrast, do not have as large an impact on evaluation curves in Shogi, even though they are also adopted by most Shogi programs. Figure 4 (center) plots evaluation curves for Shogi without a quiescence search, and Fig. 4 (right) plots curves with a quiescence search with depth 8 based on the method used in $KFEnd^8$ with additional consideration given to threats. They show the conditioned curves for the condition where the difference of “King Unsafety” (KU) of the two players ≥ 0.25 (whose range is 0, 1). Here, “Unsafety (B-W)” is the difference between the KUs of the first player and that of the second player. We can see that both evaluation curves are quite similar. This suggests that the evaluation functions in Shogi have more tolerance to tactical positions.

Quality of Adjustments in Chess

Here, we present the results of adjusting the weights in Chess. Because we did not know what new features would be incorporated into Crafty, we focused on existing features. We turned off features by setting the weight to zero, and then tested how well the weight was recovered with our method. Figure 5 plots the evaluation curves for the feature of “Bishop Evaluation” (BE), which evaluates the mobility and development of Bishops. In all three graphs, the broken (dotted) curve is an evaluation curve for the positions whose BE is more (less) than or equal to 50. The graph at right is for the original evaluation function of Crafty, and the graph at left is for a modified one whose BE was turned off. We can see that the conditioned curves differ from the total curve in the graph at left.

We then adjusted the weights of BE with MLM and LS. The center graph in Fig. 5 plots the curves for the evaluation function adjusted by LS. We can see that the conditioned curves in the graph are much closer to the total curve. Table

⁸<http://www31.ocn.ne.jp/~kfend/>

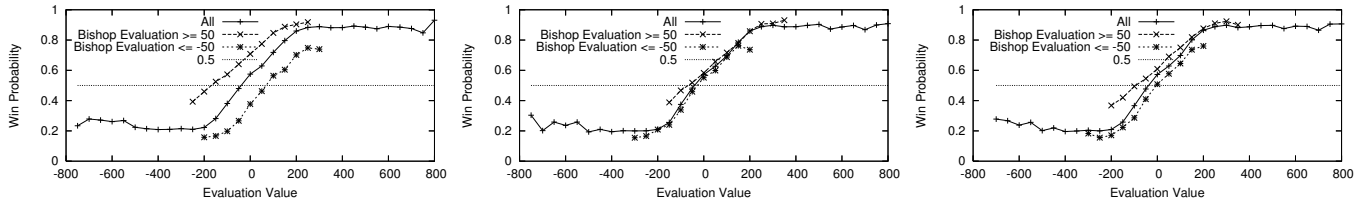


Figure 5: Evaluation curve in Chess (left: without Bishop Evaluation, center: adjusted by LS, right: original Crafty)

Table 2: Results for self-play (wins - losses - draws)

Chess					
MLM v.s. Turn off	31 - 11 - 30	MLM v.s. Crafty	18 - 22 - 32		
LS v.s. Turn off	38 - 15 - 19	LS v.s. Crafty	16 - 18 - 38		
Crafty v.s. Turn off	36 - 17 - 19				
Shogi					
MLM v.s. Orig.	52 - 28 - 0	MLM v.s. Hand	35 - 42 - 3		
Hand v.s. Orig.	59 - 21 - 0				

I summarizes the weights adjusted by MLM and LS in terms of relative values where the weight of the original Crafty was 1.0.

We conducted 72 self-plays between programs before adjustment, two programs after adjustment, and the original Crafty to find whether there were any improvements. Each player was given 10 minutes per game. The results are summarized in the upper half of Table 2. The programs after adjustment (MLM and LS) had more wins than those before adjustment (Turned off) and they were statistically significant with a significance level⁹ of 5%. Therefore adjustments based on our method effectively improve the evaluation functions. As there were no significant differences between the adjusted evaluation functions (MLM and LS) and the original of Crafty, automatic was as effective as manual adjustment.

Quality of New Evaluation Functions in Shogi

We introduced a new evaluation feature to Shogi, the difference in the “King’s Unsafety” (KU)s for both players.

The conditioned curves of the evaluation function in GPS Shogi differ from the total curve as shown in Fig. 4 (center), when there is a large difference between the KU s of both players. We therefore prepared a new evaluation function and adjusted its weights with our methods. GPS Shogi originally had two kinds of evaluation functions. The first one was for the opening (e_o) and for evaluating the material balance, as well as the combination of pieces to take the development of pieces into account. The second one was for the endgame (e_e) and for evaluating the relative positions of the Kings and the other pieces. They were combined by a progress rate pr whose range was 0, 1:

$$e(p) = (1 - pr) \cdot e_o + pr \cdot e_e. \quad (3)$$

⁹These were measured with a program that took draws into account (<http://groups.google.com/group/rec.games.chess.computer/msg/764b0af34a9b4023>, posted to rec.games.chess.computer).

Table 3: Results of adjusting weights in Shogi

	w_1	w_2		w_1	w_2
MLM	-115	83	Hand	-125	50

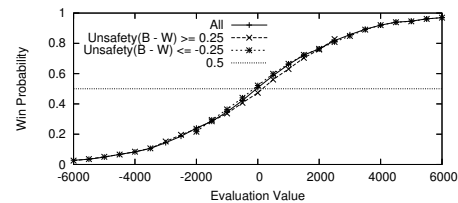


Figure 7: Evaluation curve in Shogi (difference in KU s, adjusted by MLM)

We then designed a new evaluation function that incorporated two new features, i.e., f_a and f_d :

$$e'(p) = (1 - pr) \cdot e_o + pr \cdot (e_e + w_1 \cdot f_a + w_2 \cdot f_d), \quad (4)$$

where f_a represents the difference in KU s measured using attacking pieces and f_d represents the difference measured using the defending pieces. Here, the differences are multiplied by the rate of progress in $e'(p)$ because it is empirically known that such differences are of more importance near the endgame. Equation (4) becomes equivalent to Eq. (3) when its weights w_1 and w_2 are 0.

Table 3 compares the weights adjusted by MLM as well as those manually adjusted. We can see that they have similar values. The evaluation curves after adjusting them with MLM are plotted in Fig. 7. (We have omitted manually adjusted curves because they are very similar to those in Fig. 7). The conditioned curves are much closer to the total curves than those in Fig. 4 (center).

We conducted 80 self-plays between programs before adjustment and two programs adjusted by MLM and manually to find whether there were any improvements. We used positions after 30 moves from the professional game records as the initial positions for the self-plays. Each player was given 25 minutes per game. The results are summarized in the lower half of Table 2. The program with the new evaluation function (MLM) had more wins against the original program (Orig.), and it was statistically significant with a significance level of 5% in a binomial test. Adjustments based on our method therefore effectively improved evaluation functions. There were no statistically significant differences between adjustments done by MLM and those done manually.

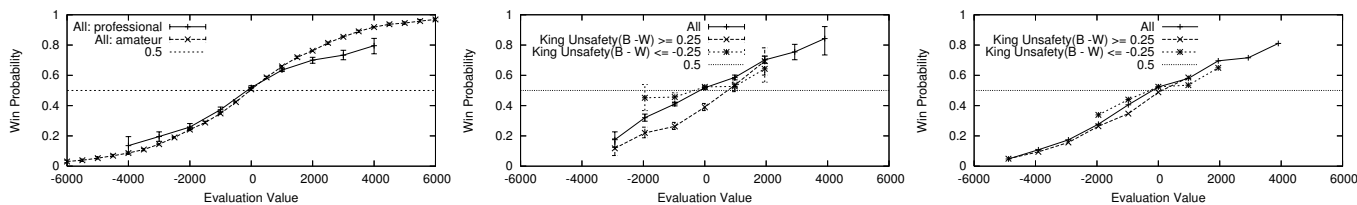


Figure 6: Evaluation curves in Shogi (left: amateur records v.s. professional records, center: before adjustment (professional), right: after adjustment (professional))

Dependence on Game Records We conducted additional experiments with professional game records. We used 603 records from the 59th Junisen, a professional championship tournament in Shogi. Figure 6 (left) plots the total evaluation curves for the professional records, as well as those for amateur records (Shogi Club 24). Because there were an insufficient number of professional records, we used intervals consisting of more than 100 positions and added error bars for the confidence interval of 5%. We can see that the probability of wins for the professional records increases more gradually than that for the amateur records. This suggests that difficult positions appear more often in professional game records for computers.

Figure 6 (center) plots the evaluation curves for the original evaluation function. Although the curves are not as clearly sigmoid due to the limited number of the records, we can see that the conditioned curves differ from the total curve in the professional records, as well as in the amateur records (Fig. 4). Figure 6 (right) plots the evaluation curves for the new evaluation function adjusted by MLM in the previous section. The conditioned curves are much closer to the total curves for the professional records, even though the evaluation function was adjusted using the amateur records. Evaluation functions adjusted by using amateur records are thus also expected to be effective in professional records.

Concluding Remarks

We proposed a method of visualizing and adjusting evaluation functions based on evaluation curves formed with evaluation values and win probability. We proposed the use of win percentages in positions having the same evaluation values to approximate win probability, where the positions we used were stored in a large database of game records.

Evaluation curves form a sigmoid and may split depending on the properties of positions, and such split curves indicate that some features are missing in the evaluation function that is visualized. Evaluation curves are therefore useful for testing the effectiveness of new features related to split conditions. We can improve evaluation functions with effective new features once they are found. The computational cost of visualization is much less than that with statistical tests or self-play. Our experiments revealed that visualization works well with major programs in Chess, Shogi, Othello, and Go. We also proposed supervised learning of weights in evaluation functions, so that split curves would fit the total curve. The experiments with self-play in Chess and Shogi demon-

strated that evaluation functions with split curves were actually weak, and that automated adjustment successfully remedied the problem.

We manually choose the properties of positions to be tested in evaluation curves at present using empirical knowledge about the target game. Automating these is an interesting topic for further research toward fully and automatically generating evaluation functions. The experiments with quiescence searches also suggest that this visualization could be extended to test the soundness of search algorithms.

Acknowledgment

We would like to thank Dr. Akihiro Kishimoto of Future University-Hakodate and some anonymous referees for providing us with beneficial feedback on the paper.

References

- Baxter, J.; Tridgell, A.; and Weaver, L. 2000. Learning to play chess using temporal-differences. *MACHINE LEARNING* 40(3):243–263.
- Bouzy, B., and Helmstetter, B. 2003. Monte Carlo Go developments. In *Advances in Computer Games. Many Games, Many Challenges*, 159–174. Kluwer.
- Buro, M. 1998. From simple features to sophisticated evaluation functions. In *Proceedings of the First International Conference on Computers and Games*, 126–145. Tsukuba, Japan: Springer-Verlag.
- Gomboc, D.; Marsland, T. A.; and Buro, M. 2003. Evaluation function tuning via ordinal correlation. In *the Advances in Computer Games Conference*, volume 10, 1–18.
- Guyon, I., and Elisseeff, A. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research* 3:1157–1182. Special Issue on Variable and Feature Selection.
- Kocsis, L., and Szepesvari, C. 2006. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, volume 4212, 282–293. Springer.
- Samuel, A. L. 1959. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development* 3(3):211–229.
- Schaeffer, J. 2000. The games computers (and people) play. *Advances in Computers* 50:189–266.
- Tesauro, G. 1995. Temporal difference learning and TD-Gammon. *Communications of the ACM* 38(3):58–68.