On Strength Adjustment for MCTS-Based Programs

I-Chen Wu,[*] Ti-Rong Wu,[*] An-Jen Liu,[*] Hung Guei, Tinghan Wei

Department of Computer Science, National Chiao Tung University, 1001 University Road, Hsinchu, Taiwan, ROC
{icwu,kds285,andyliu,ghung,ting}@aigames.nctu.edu.tw

Abstract

This paper proposes an approach to strength adjustment for MCTS-based game-playing programs. In this approach, we use a softmax policy with a strength index z to choose moves. Most importantly, we filter low quality moves by excluding those that have a lower simulation count than a pre-defined threshold ratio of the maximum simulation count. We perform a theoretical analysis, reaching the result that the adjusted policy is guaranteed to choose moves exceeding a lower bound in strength by using a threshold ratio. The approach is applied to the Go program ELF OpenGo. The experiment results show that z is highly correlated to the empirical strength; namely, given a threshold ratio 0.1, z is linearly related to the Elo rating with regression error 47.95 Elo where $-2 \leq z \leq 2$. Meanwhile, the covered strength range is about 800 Elo ratings in the interval of z in $[-2,2]$. With the ease of strength adjustment using z, we present two methods to adjust strength and predict opponents' strengths dynamically. To our knowledge, this result is state-of-the-art in terms of the range of strengths in Elo rating while maintaining a controllable relationship between the strength and a strength index.

Motivation

Artificial intelligence in computer games has made significant progress in recent years, especially after DeepMind's AlphaGo (Silver et al. 2016) defeated human Go champions by a large margin in 2016. DeepMind then followed up their success with AlphaGo Zero (Silver et al. 2017b) to further improve the playing strength without requiring human knowledge, resulting in much stronger programs against earlier versions of AlphaGo. Both AlphaGo and AlphaGo Zero incorporate deep neural networks into Monte Carlo tree search (MCTS) (Browne et al. 2012; Coulom 2006; Kocsis and Szepesvári 2006), which itself had been a major breakthrough that was responsible for more than ten years of rapid growth in computer games, particularly computer Go, before AlphaGo was announced.

Since AlphaGo Zero, many other programs such as FineArt (Tencent AI Lab 2018), Leela Zero (Pascutto 2018), and ELF OpenGo (Tian et al. 2018) have successfully reproduced the AlphaGo Zero algorithm. AlphaGo Zero's method was also applied to other games such as chess and shogi, reaching strength levels much higher than human champions and other top programs (Silver et al. 2017a).

Super-human level game playing programs capture the imagination and fascination of society at large; for human players, these programs pose an interesting challenge and offer opportunities for learning (Hunicke and Chapman 2004; Demediuk et al. 2017; Paulsen and Fürnkranz 2010; Sephton, Cowling, and Slaven 2015). However, it is also important to fit program difficulty to appropriate levels for human players. On the one hand, human players may lose interest if the game program is too weak; on the other hand, excessive difficulty tends to lead to frustration (Hunicke and Chapman 2004). From our observation, in the context of learning with programs, it is difficult to offer feedback for human players if they are constantly on the losing side. Thus, in order to achieve an overall better game experience, and to improve the learning process for players, it is imperative to balance program difficulty accordingly. The fundamental goal is to offer programs with a wide variety of strength levels.

A simple and straightforward method to offer different program strengths is to reduce the total thinking time, or the total simulation count in MCTS, if MCTS is used. However, with this method, the search tree's relatively smaller size leaves the program vulnerable to tactical traps. For example, the ladder problem in Go is one of the most elementary shapes taught to human players; for programs, however, search is often required to handle ladders properly. It has been shown that when adjusting program strength through reduction, simulation count and playing strength do not form a linear relationship (Sephton, Cowling, and Slaven 2015). In fact, once the number of simulations fall below a certain threshold, the program playing strength drops catastrophically.

[*] Equal Contribution.

Another straightforward approach is to offer one program for each strength level, e.g., train one network for each difficulty. A good example of this type of strength adjustment is Paulsen and Fürnkranz's (2010) work on training chess evaluation functions for different strengths. However, this approach usually requires large amounts of time and effort to tune and test the programs. This is especially difficult for games like Go, where strength levels span a wide range from 30 kyu to professional 9 dan (Hollosi and Pahle 2018), about a 3000 Elo rating difference.

This paper reviews a strength adjustment (SA) approach based on the softmax policy and proposes our modification in the following section (Strength Adjustment). We apply the method to the open source Go program ELF OpenGo (abbr. ELF for the rest of this paper) and demonstrate that the method can be easily used to adjust the program strength, covering a range of over 800 Elo rating. In the Strength Analysis section, this paper presents a hypothesis and performs theoretical analyses to justify the empirical strengths shown in the Strength Adjustment section. Having demonstrated that the program strength can be adjusted with relative ease, we introduce methods to adjust the strength dynamically in the section of Dynamic Strength Adjustment. Finally, we provide discussion and summarize our contributions in the Conclusion section.

Strength Adjustment

In this section, we first review past work on strength adjustment, then present our modifications to the method. We apply the modified approach to the Go program ELF and provide empirical data.

Past Work

For strength adjustment, Sephton et al. (2015) presented a method for MCTS-based game-playing programs using a simple softmax policy as follows. Given strength index z, choose moves i with probability $N_i^z / \Sigma_j N_j^z$, where N_i is the number of simulations on move i in MCTS. For simplicity of discussion in the rest of this paper, let $N_i \geq N_j$ if $i < j$, i.e. N_1 is the maximum.

Conceptually, z is the inverse of the softmax temperature. When z is higher, the policy tends to choose the move with higher simulation counts, which tends to be a higher-quality move as is the case with MCTS. When z approaches infinity, the moves with the highest simulation counts are guaranteed to be chosen, and thus policy exhibits the same behavior as the original MCTS. When $z = 0$, all the moves are chosen with equal probability, i.e., moves are chosen randomly. When z approaches negative infinity, the moves with the lowest simulation counts are chosen, i.e. the policy tends to choose the lowest quality moves.

Thus, z can serve as an index of strength. Sephton et al. (2015) showed through experiments that z is correlated to the empirical strength, but experiment only covered six trials on z (from 1 to 6) for the game Lords of War, and the differences of win rates for these values of z are ranged from 5% to 24%, equivalent to a range of 100 Elo rating.

As above, when z is low, the policy tends to choose low quality moves. However, in MCTS, many moves are not visited during simulation, or in some cases, visited very few times only because of the exploration bias. For this reason, it is not a good idea to allow the policy to choose the lowest-quality moves, which would result in a much weaker program or unpredictable behavior.

In order to avoid choosing the very lowest-quality moves, Sephton et al. suggested choosing the first n best moves as candidates, where n is a given fixed value. However, it is still possible to choose a very low-quality move, e.g., in the case that only one move is viable while the others are extremely bad, the policy is still likely to choose bad moves.

Our Approach

In our approach, we follow the softmax policy to choose moves via the strength index z. However, from our observation, it is critical to screen the candidate moves. For this issue, our approach is to use a threshold ratio R_{th} to avoid choosing moves with small simulation counts in order to ensure the quality of moves. Namely, given a threshold ratio R_{th}, we only consider the moves i with $N_i \geq N_1 \times R_{th}$ as candidates. Assuming that the move quality is correlated to the simulation count (we discuss this in greater detail in the Strength Analysis section), this approach ensures that the qualities of the chosen moves are higher than the screened moves which do not reach the threshold. At the very least, the modified policy is less likely to choose extremely bad moves, as mentioned in the previous subsection.

For a high threshold ratio, more low-quality moves are filtered. When $R_{th} = 1$, the move with the highest simulation count is always chosen, behaving the same way as the original MCTS. In contrast, for a low threshold ratio, many low-quality moves are not filtered. Thus, it is important to set a reasonable threshold ratio, where the goal is to filter most low-quality moves, while simultaneously allowing reasonable moves to be considered.

In contrast to the previous work (Sephton, Cowling, and Slaven 2015), our empirical results in the next subsection show that strengths can be adjusted across a wide range over 800 Elo rating with the threshold ratio 0.1 and the interval of z in $[-2,2]$. Thus, our approach is very suitable for games that are considered to have very high depth (Cauwet et al. 2015).

Empirical Results

We apply the above approach to the Go program ELF and present the experiment results. All the experiments are performed on machines equipped with one GTX 1080Ti GPU, one Intel Xeon E5-2683 v3 (14 cores in total), 2.6 GHz, 128 GB memory, and with Linux. All games are played with one second per move, using one GPU and six CPU cores. For each benchmark, 250 games are played against a baseline, ELF with $R_{th} = 0.1$ and $z = 0$. Note that we do not use the original ELF, equivalent to $z = \infty$, as the baseline since it is much too strong for some trials, such as when $z = -\infty$.

Table 1 (below) shows the win rates and the relative Elo rating of the ELF versions with $R_{th} = 0.1$ and with different z against the baseline. Note that the shown Elo ratings are relative to the original ELF which is set to 0 for simplicity of analysis. Since ELF follows the process of training AlphaGo Zero with 20 blocks, its real Elo rating is supposed to be between 4000 and 5000 (Silver et al. 2017b).

z	Win rate (±errors)	Elo rating (±errors)
∞	97.6% (±1.9%)	0 (-106, +289)
2	94.4% (±2.9%)	-153 (-78, +133)
1.5	92.4% (±3.4%)	-210 (-70, +107)
1	91.2% (±3.6%)	-237 (-66, +98)
0.5	71.6% (±5.7%)	-483 (-46, +52)
0	50.0%	-644
-0.5	35.6% (±6.1%)	-747 (-48, +44)
-1	21.6% (±5.2%)	-868 (-59, +49)
-1.5	13.2% (±4.3%)	-971 (-76, +58)
-2	12.4% (±4.2%)	-983 (-79, +59)
-∞	7.2% (±3.3%)	-1088 (-111, +71)

Table 1. The win rates (against ELF with z=0) and Elo ratings (relative to the original ELF) with respect to z when $R_{th} = 0.1$.

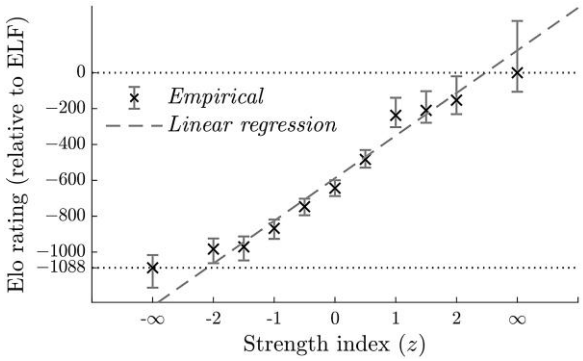

Figure 1. The correlation between z and Elo rating when $R_{th} = 0.1$.

Figure 1 shows the correlation between z and the Elo ratings. Interestingly, both are highly correlated with a low linear regression error 47.95 Elo, in terms of the Elo rating, when z is between -2 to 2. In addition, the range of strength is very wide, covering 1088 Elo rating for all z and 830 for the interval of z in $[-2,2]$.

Furthermore, Figure 2 depicts the correlation between z and the Elo rating for different threshold ratios, 0, 0.02, 0.05, 0.1, 0.25, and 0.5. All games are also played against the same baseline as above. From the Figure 2, the correlation between Elo ratings and z is also highly correlated to R_{th} in most cases. A higher value of z usually corresponds to higher Elo ratings.

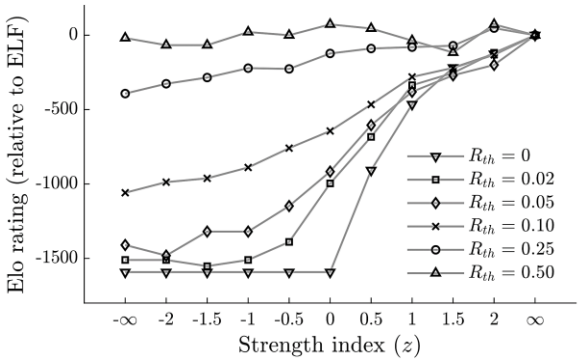

Figure 2. Elo rating (relative to ELF) in different threshold ratios and strength indices.

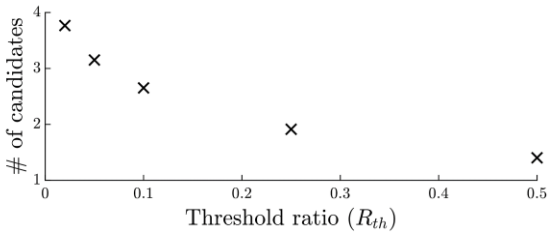

Figure 3. The number of candidates with respect to R_{th}.

We observe that high values of R_{th} are not appropriate. For example, when $R_{th} = 0.5$, the Elo rating has no significant changes across different values of z. An intuitive explanation for this is that with a high threshold ratio, most candidate moves are filtered, so the value of z does not matter as much. Figure 3 shows that the average number of candidates is only 1.4 for $R_{th} = 0.5$, and 1.9 for $R_{th} = 0.25$. Another effect is that the adjusted strength range is narrower, e.g., smaller than 500 Elo rating for $R_{th} = 0.25$.

On the other hand, for low threshold ratios, the Elo rating drops quickly, and the strength for different values of z show no difference, e.g. when $R_{th} = 0$ and $z \leq 0$, and when $R_{th} = 0.02$ and $z \leq -1$.

Thus, judging from both Figure 2 and Figure 3, the threshold ratios of 0.05 and 0.1 appear to be suitable for our needs. For simplicity of analysis, 0.1 will be used as the threshold ratio, unless otherwise stated.

Strength Analysis

The above empirical results show that the strengths are highly correlated to z. In fact, between $-2 \le z \le 2$ and a threshold ratio of 0.1, z and the strength show a near linear relationship with regression error 47.95 Elo. However, the strength or Elo rating should be fixed when z approaches ∞ or $-\infty$. Thus, intuitively, the curve of the Elo rating strength according to the z value should be shaped similar to a logistic function. Applying logistic regression (Hosmer Jr, Lemeshow, and Sturdivant 2013), the curve is close to a logistic function with error 26.00 Elo ($\beta_0 = -0.25$, $\beta_1 = 1.16$).

This section investigates this conjecture of logistic regression from a theoretical perspective. First, we review the generalized Bradley-Terry model. Second, we present a hypothesis on move strength. Then, from theoretical analysis, we show that the derived strengths are close to the empirical strengths. We calculate the regression error between the derived and the empirical strengths to justify the hypothesis.

Generalized Bradley-Terry Model

The Bradley-Terry model has been the foundation of various ranking systems, including the Elo rating system. The model is used to estimate the strengths of players and predict the win rates among these players. Note that *moves* mentioned in the Past Work Subsection can be viewed as players here. Namely, each player i is associated with a positive value γ_i representing the strength of i, and the probability that i wins over j is $\gamma_i/(\gamma_i + \gamma_j)$. Obviously, the higher γ_i is, the higher the winning rate (implying a stronger player). The Elo rating of individual i is $Elo_i = 400 \log_{10} \gamma_i$ (Coulom 2008). For simplicity of discussion in this paper, we also define the rating $E_i = \ln \gamma_i$, whose corresponding Elo rating is $Elo_i = 400(\log_{10} e)E_i$.

The Bradley-Terry model has also been generalized to handle competitions involving more than two players (Coulom 2007; Hunter 2004). Namely, the probability that i wins among n players, 1, …, n, is formulated as $\gamma_i / \sum_j^n \gamma_j$. Another generalization is to allow competitions among teams, instead of players. The strength and its corresponding rating of a team of n players is estimated as

$$\gamma^{all} = \prod_i^n \gamma_i, \text{ and } E^{all} = \sum_j^n E_j. \quad (1)$$

In this paper, we also define the average strength and rating of a team of n players to be

$$\gamma^{avg} = \left(\prod_i^n \gamma_i\right)^{\frac{1}{n}}, \text{ and } E^{avg} = \left(\frac{1}{n}\right)\sum_j^n E_j. \quad (2)$$

This is useful when n is not fixed. In addition, consider a team that can choose one and only one player to participate and choose player i with probability π_i, where $\sum_j^n \pi_j = 1$. Thus, the strength and rating of the team are

$$\gamma = \prod_i^n \gamma_i^{\pi_i}, \text{ and } E = \sum_j^n \pi_i E_j, \quad (3)$$

respectively, for the reason as illustrated below. For example, let $\pi_i = N_i / \sum_j^n N_j$. We can consider the team composed of $\sum_j^n N_j$ players, among which the number of players i is N_i. Thus, the average strength and rating of the team are the same as γ and E in formula (3), respectively.

Hypothesis

As mentioned above, moves with higher simulation counts N_i in MCTS normally tend to have higher quality. Following this notion, we present a hypothesis for further theoretical analysis. Assume that given a position the strength of move i is proportional to N_i^H. Here, H denotes a conjectured strength index for moves to be selected in MCTS in the previous sections. Namely, let $\gamma_i = c \times N_i^H$. Here, c is a constant coefficient with respect to the same game position, i.e. different positions may have different relative strengths, and therefore will have a different value of c.

The rating of move i is $E_i = \ln \gamma_i = \ln c + H \ln N_i$. For the simplicity of analysis, we use E_i in the following analysis without loss of generality. If the Elo rating is preferred, Elo_i can be obtained by a simple conversion, as described above.

Let $\gamma(z)$ and $E(z)$ denote the overall strength and rating following the above method for strength adjustment, which chooses among all moves i using the softmax policy $\pi_i(z) = N_i^z / \Sigma_i (N_i^z)$. From the above Bradley-Terry model for team strength, we can derive that

$$\gamma(z) = \prod_i \gamma_i^{\pi_i(z)}, \text{ and} \quad (4)$$

$$\begin{aligned} E(z) = \ln \gamma(z) &= \sum_i \pi_i(z) E_i \\ &= \sum_i \pi_i(z)(\ln c + H \ln N_i) \\ &= \ln c + H \sum_i \pi_i(z) \ln N_i. \end{aligned} \quad (5)$$

In the above formula, the first item in (5) is fixed for this position, and therefore it can be removed to obtain relative ratings, say, relative to the rating where $z = \infty$ (which always choose the move with the maximum simulation counts N_1) as follows.

$$\begin{aligned} E_{rel}(z) &= E(z) - E(\infty) \\ &= H \sum_i \pi_i(z)(\ln N_i - \ln N_1) \end{aligned}$$

$$= H \sum_i \pi_i(z)(\ln R_i), \quad (6)$$

where R_i is the ratio N_i/N_1. Since all moves with the ratio less than R_{th} are filtered out, $R_i \geq R_{th}$. In addition, since N_1 is the maximum among N_i, $R_i \leq 1$ and $\ln R_i$ are therefore all non-positive. Thus, we obtain

$$E_{rel}(z) \geq H \times (\ln R_{th}). \quad (7)$$

An important implication in formula (7) is that the relative ratings of the chosen moves are at worst $H \times (\ln R_{th})$. Assume $R_{th} = 0.1$. The relative ratings of all chosen moves are at worst $H \times \ln 0.1 \cong -2.3H$, no worse than move 1 by $2.3H$. Since H is a constant under this hypothesis, this implies that the strength of any chosen move is at worst a fixed value. This ensures the quality of all chosen moves.

Now, let us consider following the above SA method to play a game g, containing a sequence of m_g moves or m_g positions to move. Let $\gamma^{(j)}(z)$ and $E^{(j)}(z)$ denote the strength and rating of the move made at the jth position (i.e. on the jth turn), which can be formulated as follows.

$$\gamma^{(j)}(z) = \prod_i \gamma_i^{\pi_i(z)}, \text{ and} \quad (8)$$

$$E^{(j)}(z) = \sum_i \pi_i^{(j)}(z) E_i^{(j)} = \ln c^{(j)} + H \sum_i \pi_i^{(j)}(z) \ln N_i^{(j)}, \quad (9)$$

where $\gamma_i^{(j)}$, $E_i^{(j)}$, $\pi_i^{(j)}$ and $N_i^{(j)}$ are respectively the strength, rating, policy and simulation count of moves i at the jth position in the game, and $c^{(j)}$ is the coefficient with respect to the position.

Furthermore, let $\gamma^g(z)$ and $E^g(z)$ denote the *averaged* strength and rating as follows.

$$\gamma^g(z) = \left(\prod_j \gamma^{(j)}(z)\right)^{1/m_g}, \text{ and} \quad (10)$$

$$E^g(z) = \left(\frac{1}{m_g}\right)\sum_j E^{(j)}(z) = \left(\frac{1}{m_g}\right)\sum_j \left(\ln c^{(j)} + H \sum_i \pi_i^{(j)}(z) \ln N_i^{(j)}\right) = \sum_j \ln c^{(j)} + \left(\frac{H}{m_g}\right)\sum_j \sum_i \pi_i^{(j)}(z) \ln N_i^{(j)}. \quad (11)$$

Note that we evaluate the averaged strength and rating as Formula (1), instead of the aggregated values in (2), simply because the number of moves in a game is not fixed.

In the above formula, the first item is fixed, and therefore can be omitted when calculating ratings relative to the one with $z = \infty$, similarly, as follows.

$$E_{rel}^g(z) = E^g(z) - E^g(\infty) = \left(\frac{H}{m_g}\right)\sum_j \sum_i \pi_i^{(j)}(z) \left(\ln R_i^{(j)}\right), \quad (12)$$

where $R_i^{(j)} = N_i^{(j)}/N_1^{(j)}$. Moreover, let the relative rating be normalized to be independent of the value H as follows.

$$E_{norm}^g(z) = E_{rel}^g(z)/H = \left(\frac{1}{m_g}\right)\sum_j \sum_i \pi_i^{(j)}(z) \left(\ln R_i^{(j)}\right) = \mathbb{E}_g\left[\sum_i \pi_i^{(j)}(z) \left(\ln R_i^{(j)}\right)\right]. \quad (13)$$

For stochastic analysis, we extend by collecting some sets of games, each of which is collected from the games under a designated threshold ratio in the above empirical experiments. We exclude extreme cases to minimize the effect of noise for our analysis. For example, the cases of $z = \infty$ and $z = -\infty$ are not included.

For simplicity of analysis, let us illustrate the case for $D_{0.1}$, denoting the set of games with threshold ratio 0.1, which contains about 2000 games. The expected relative rating under the set $D_{0.1}$ is

$$E_{norm}^{D_{0.1}}(z) = \mathbb{E}_{g\sim D_{0.1}}\left[E_{norm}^g(z)\right] = \mathbb{E}_{g\sim D_{0.1}}\left[\mathbb{E}_g\left[\sum_i \pi_i^{(j)}(z) \left(\ln R_i^{(j)}\right)\right]\right] = \mathbb{E}_{j\sim D_{0.1}}\left[\sum_i \pi_i^{(j)}(z) \left(\ln R_i^{(j)}\right)\right]. \quad (14)$$

Figure 4 (below) depicts the solid curve of $E_{norm}^{D_{0.1}}(z)$ calculated from the set $D_{0.1}$ according to formula (14). The left y axis indicates the value of $E_{norm}^{D_{0.1}}(z)$. The curve resembles a logistic function. Now, let $E^{D_{0.1}}(z)$ denote the expected rating, and $H^{D_{0.1}}$ be the value H, under the set of games, $D_{0.1}$. Thus, we have

$$E^{D_{0.1}}(z) = E^{D_{0.1}}(\infty) + H^{D_{0.1}}\, E_{norm}^{D_{0.1}}(z). \quad (15)$$

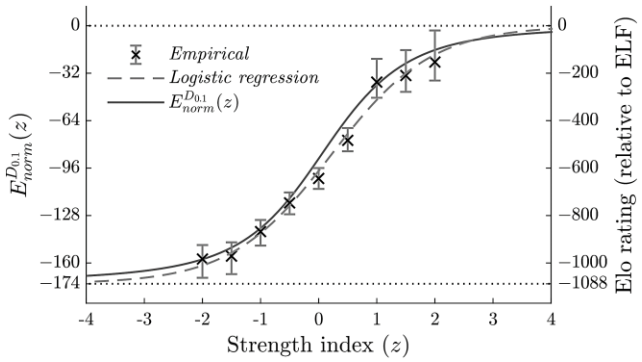

Figure 4. The curve of $E_{norm}^{D_{0.1}}(z)$ and the empirical data.

Then, we can derive that

$$E^{D_{0.1}}(-\infty) = E^{D_{0.1}}(\infty) + H^{D_{0.1}}\, E_{norm}^{D_{0.1}}(-\infty), \text{ and} \quad (16)$$

$$H^{D_{0.1}} = \frac{E^{D_{0.1}}(-\infty) - E^{D_{0.1}}(\infty)}{E_{norm}^{D_{0.1}}(-\infty)}. \quad (17)$$

Since the values $E^{D_{0.1}}(\infty)$ and $E^{D_{0.1}}(-\infty)$ are supposed to approximate the strength in the empirical experiments, they can be replaced by the empirical strengths at $z = \infty$ and $z = -\infty$, whose relative Elo ratings are 0 and -1088 as shown in Table 1. Thus, the value $H^{D_{0.1}}$ is derived to be

6.243 according to the above formula. The right y axis in Figure 4 follows the y axis in Figure 1. The regression error to the empirical strengths for z between -2 and 2 is about 40.45 Elo, and the regression error to a logistic regression curve is about 10.51 Elo ($\beta_0 = -0.05$, $\beta_1 = 1.17$). These low errors justify the hypothesis.

Data set	$D_{0.02}$	$D_{0.05}$	$D_{0.1}$	$D_{0.25}$	$D_{0.5}$
H	4.385	5.369	6.243	6.244	3.292

Table 2. The conjectured strength indices estimated in different data sets.

In our experiments, we also derived the value H for other sets of games, as shown in Table 2. From the table, $H^{D_{0.25}}$ is almost the same as $H^{D_{0.1}}$, while $H^{D_{0.5}}$, $H^{D_{0.05}}$ and $H^{D_{0.02}}$ are lower. For $H^{D_{0.05}}$ and $H^{D_{0.02}}$, our conjecture is that the noise incurred from having a low threshold ratio are high as the following illustration. In the case of $R_{th} = 0.02$, since the average number of simulation counts for the best move is about 259.4 with the one second time limit, it is highly likely to include the moves with very low simulation counts (the threshold is about $259.4 \times 0.02 \cong 5.2$). Since many of these simulations may be generated simply because of the exploration bias, these simulations may introduce noise and therefore affect the verification of our hypothesis.

As for $H^{D_{0.5}}$, we observe that the average number of candidates is 1.4 from Figure 3. Since the number is relatively low, in many cases, the policy chooses only from a single candidate move. Therefore, the distribution is insufficient to justify our hypothesis. As an example, the most extreme case is where the threshold ratio is 1, and only the moves with the highest simulation counts are chosen, as in the original MCTS. The value of H in this case does not affect the policy at all, since there is only one choice.

Dynamic Strength Adjustment

As stated in the previous sections, this paper presents a flexible strength adjustment method simply by altering the value z with an appropriate R_{th}, say 0.1. Moreover, the strength ratings are approximately linear with respect to z in the interval [-2, 2]. This allows us to fit the programs strength to its opponents' dynamically, provided the opponents' strengths are within [-983, -153], corresponding to the range of z in [-2, 2]. This section introduces two types of dynamic strength adjustment, *inter-game* and *intra-game strength adjustment*. For the former, strengths are adjusted based on previous game results, while for the latter strengths are adjusted within each game. We present two methods of dynamic strength adjustment (DSA) here only to showcase how we can predict opponent strengths and adjust accordingly with relative ease; the presented methods are by no means a comprehensive review of all available methods.

Inter-game Strength Adjustment

Inter-game strength adjustment is relatively easy. Namely, the strength index z of a game is adjusted based on the previous game results and the index remains unchanged within the game.

In this section, a simple adjustment method is presented and demonstrated to predict the opponent's strength. The prediction can then be used to set z accordingly. The strength index z is decreased for every win and increased for every loss, both by a small amount Δz. The initial value of z is set to 0. The value Δz is initialized to Δz_{init} and decreased by a discount factor r for each game, capped by a lower bound Δz_{low}.

In our experiments for the method, Δz_{init} is 0.375, approximately equivalent to 100 in Elo rating based on the linear regression in Figure 1, then decreased by a factor of $r = 0.95$ for each game, with $\Delta z_{low} = 0.03$, equivalent to 8 in Elo rating. In the experiment, 100 games are played against each of the five opponents whose strength indices are $z = 2, 1, 0, -1, -2$ for a total of 500 games. The experiment is repeated five times and the following experiment results are based on the average values of the five times.

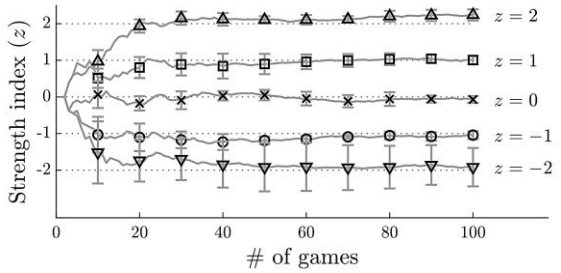

Figure 5. Strength index estimation for inter-game SA.

Opponent	$z = 2$	$z = 1$	$z = 0$	$z = -1$	$z = -2$
w/o DSA WR	5.6% (±2.9%)	8.8% (±3.6%)	50.0%	78.4% (±5.2%)	87.6% (±4.2%)
Inter-game WR	43.4% (±4.4%)	46.6% (±4.5%)	52.0% (±4.5%)	50.0% (±4.5%)	54.8% (±4.5%)
Avg. z	1.93	0.88	-0.04	-1.06	-1.73

Table 3. Win rate (WR) and average dynamic strength index (Avg. z) against different opponents using inter-game SA.

In Figure 5 each of the five lines indicates the predicted z for each opponent. The result shows that our method can approximately predict opponents' strengths and clearly distinguish five opponents. Table 3 also shows that the aver-

aged win rate for each opponent is within 43-54% and the averaged predicted z is very close the opponent's.

Intra-game Strength Adjustment

Intra-game strength adjustment is relatively challenging, given that the algorithm only has one game to predict the opponent's approximate level of play. Players often play inconsistently within the same game, mixing good and bad moves. On the one hand, adjusting by large amounts leads to high variance of program strength. On the other hand, if strengths are adjusted by a small amount, the effects may not be sufficiently obvious.

Our method is as follows. In principle, we still attempt to maintain all moves so that the overall win rate is around 50%. For each move, we first estimate the current win rate W, by using the MCTS win rate of the move with the most simulation counts. The index z is decreased when $W > 50\%$ and increased when $W < 50\%$, both by Δz. The program chooses moves based on the softmax policy proposed earlier.

However, for stability, Δz is set to be relatively small when W is within a range (50%$-\epsilon$, 50%$+\epsilon$), where ϵ is a user defined value, say 10%. Namely,

$$\Delta z = \begin{cases} \Delta Z, & \text{if } |W - 50\%| \geq \epsilon \quad (18) \\ \Delta Z \times \dfrac{|W - 50\%|}{\epsilon}, & \text{if } |W - 50\%| < \epsilon \quad (19) \end{cases}$$

In addition, ΔZ decreases linearly from an initial value, say 0.1, to 0 after a number of moves, say 150 moves. The purpose is to cool down the amplitude of changes as games progress, since the program should have an idea of its opponent's strength by that stage in the game. This cool down mechanism is important because without it, the program will make unreasonable concessions when it is in the lead, or ramp up in strength indefinitely when it is behind.

In the experiment, the above method is used to play against the five opponents with strength indices $z = 2, 1, 0, -1,$ and -2. For each opponent, consider two cases of ΔZ, 0.2 and 0.1, where for each case, 100 games are played.

Table 4 presents the experiment results. The results show that the average z values over 100 games are close to the opponents' strength indices z, especially when $\Delta Z = 0.2$. Note that the predicted z for each game is the value z at the end of the game. The standard deviation is high as expected.

The results in Table 4 also show that while all the win rates except for $z = 0$ are not around 50%, when compared to the baseline win rates without DSA (as shown in the second row), the overall win rates are closer to 50%. This shows that intra-game DSA can predict opponents' strengths and even out the games. The reason why the win rates are not balanced around 50%, despite the predicted z value to be more or less accurate, is that the early moves in a game influence the outcome significantly, but the program has yet to observe its opponents' strengths sufficiently at that point.

Opponent		z = 2	z = 1	z = 0	z = -1	z = -2
w/o DSA		5.6%	8.8%	50.0%	78.4%	87.6%
$\Delta z_{init} = 0.2$	WR.	29.0%	38.0%	43.0%	64.0%	71.0%
	Avg. z	1.85	0.92	-0.10	-1.39	-1.57
	Std. z	1.82	1.81	1.41	1.73	1.56
$\Delta z_{init} = 0.1$	WR.	15.0%	32.0%	49.0%	73.0%	72.0%
	Avg. z	1.02	0.75	-0.21	-0.66	-0.96
	Std. z	0.94	0.90	0.81	0.84	0.85

Table 4. Win rate (WR), average z (Avg. z) and standard deviation of z (Std. z) against different opponents using intra-game SA.

Conclusion

In this paper, our major contributions are:

1. We propose an approach to strength adjustment for MCTS-based game-playing programs. In this approach, we follow a softmax policy (Sephton, Cowling, and Slaven 2015) with a strength index z to choose moves. Most importantly, this approach uses a threshold ratio R_{th} to filter out low-quality moves i whose simulation counts in MCTS are $N_i \leq N_1 \times R_{th}$.
2. We apply the approach to the Go program ELF, and demonstrate that we can easily adjust the strength. The empirical results show the strength covers a range of about 830 Elo ratings with a low linear regression error of 47.95 Elo, with respect to z in the range [-2, 2]. To our knowledge, this result is state-of-the-art in terms of the range of strengths in Elo rating while maintaining a controllable relationship between the strength and a strength index. Another advantage is that the program is still able to play diverse moves despite its adjusted, weaker strength.
3. We present an in-depth strength analysis for the above empirical results. First, we make the hypothesis that given a position, the strength of move i is proportional to N_i^H. From this hypothesis, the strength ratings of chosen moves are shown to be at worst a fixed value, $H \times \ln R_{th}$, lower than the best move. This justifies that the move quality is under control, avoiding exceptionally bad moves. In addition, the analysis also shows that the derived strengths are also close to the empirical strengths with regression error 40.45 Elo, and to a logistic function with regression error 10.51 Elo.
4. With the ease of strength adjustment using z, we introduce two methods to adjust strength dynamically, including inter-game and intra-game strength adjustment. The experiment results show that these methods

are able to predict the opponents' expected strengths, though the variances can be high.

In practice, we have applied our method to ELF OpenGo and three versions of the Go program CGI (Wu et al. 2018), which can cover a strength range over 3000 Elo ratings from beginners to ELF, which is much stronger than human champions. Our estimation is that ELF roughly covers the range of Elo ratings [3300, 4300] and the other three covers [2600, 3700], [1800, 2800] and [900, 2000]. The four versions have been tested on online Go websites against human players. Namely, the last two versions were tested to play against amateur players on a Go playing online site Tygem (Tongyang Online 2018), while the first two were used to play against professionals in HaiFong Go Association (HaiFong Go Association 2018).

As mentioned in the introduction, the AlphaZero algorithm has also been successfully applied to other games such as chess and shogi, reaching a strength level much higher than human champions and other top programs (Silver et al. 2017a). With our approach, we expect to be able to provide a wide range of strength levels for each of these games. We expect our approach to not only impact the Go community, but also the games community at large.

References

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4(1):1–43.

Cauwet, M.-L.; Teytaud, O.; Cazenave, T.; Saffidine, A.; Liang, H.-M.; Yen, S.-J.; Lin, H.-H.; and Wu, I-C. 2015. Depth, Balancing, and Limits of the Elo Model. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, 376–382. IEEE.

Coulom, R. 2006. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *International Conference on Computers and Games*, 72–83. Springer.

Coulom, R. 2007. Computing Elo Ratings of Move Patterns in the Game of Go. *ICGA Journal* 30(4):198–208.

Coulom, R. 2008. Whole-History Rating: A Bayesian Rating System for Players of Time-Varying Strength. In *International Conference on Computers and Games*, 113–124. Springer.

Demediuk, S.; Tamassia, M.; Raffe, W. L.; Zambetta, F.; Li, X.; and Mueller, F. 2017. Monte Carlo Tree Search Based Algorithms for Dynamic Difficulty Adjustment. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, 53–59. IEEE.

HaiFong Go Association. 2018. The Homepage of HaiFong Go Association. Retrieved from http://www.haifong.org/.

Hollosi, A., and Pahle, M. 2018. Sensei's Library. Retrieved from https://senseis.xmp.net/?RankWorldwideComparison.

Hosmer Jr, D. W.; Lemeshow, S.; and Sturdivant, R. X. 2013. *Applied Logistic Regression*, volume 398. John Wiley & Sons.

Hunicke, R., and Chapman, V. 2004. AI for Dynamic Difficulty Adjustment in Games. In *Proceedings of the Challenges in Game AI Workshop, Nineteenth National Conference on Artificial Intelligence*, 91─96. San Jose, California: AAAI Press.

Hunter, D. R. 2004. MM Algorithms for Generalized Bradley-Terry Models. *The Annals of Statistics* 32(1):384–406.

Kocsis, L., and Szepesvári, C. 2006. Bandit Based Monte-Carlo Planning. In *European Conference on Machine Learning*, 282-293. Springer.

Pascutto, G.-C. 2018. Leela-zero GitHub Repository. Retrieved from https://github.com/gcp/leela-zero.

Paulsen, P., and Fürnkranz, J. 2010. A Moderately Successful Attempt to Train Chess Evaluation Functions of Different Strengths. In *Proceedings of the ICML-10 Workshop on Machine Learning and Games*, Haifa, Israel.

Sephton, N.; Cowling, P. I.; and Slaven, N. H. 2015. An Experimental Study of Action Selection Mechanisms to Create an Entertaining Opponent. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, 122–129. IEEE.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529(7587):484–489.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2017a. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv preprint arXiv:1712.01815*.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017b. Mastering the Game of Go Without Human Knowledge. *Nature* 550(7676):354.

Tencent AI Lab. 2018. Fine Art Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Fine_Art_(software).

Tian, Y.; Ma, J.; Gong, Q.; Sengupta, S.; Chen, Z.; and Zitnick, C. L. 2018. ELF OpenGo GitHub Repository. Retrieved from https://github.com/pytorch/ELF.

Tongyang Online. 2018. The Homepage of Tygem. Retrieved from http://www.tygemgo.com/.

Wu, T.-R.; Wu, I-C.; Chen, G.-W.; Wei, T.-h.; Wu, H.-C.; Lai, T.-Y.; and Lan, L.-C. 2018. Multi-Labelled Value Networks for Computer Go. *IEEE Transactions on Games*, 10(4):378–389.