Improvement of Move Naturalness for Playing Good-Quality Games with Middle-Level Players

Chu-Hsuan Hsueh[1*†] and Kokolo Ikeda[1†]

[1*]School of Information Science, Japan Advanced Institute of Science and Technology, 1-1 Asahidai, Nomi, 923-1292, Ishikawa, Japan.

*Corresponding author(s). E-mail(s): hsuehch@jaist.ac.jp;
Contributing authors: kokolo@jaist.ac.jp;
†These authors contributed equally to this work.

Abstract

In the game field, computer programs have surpassed top human players in many games. A well-known example is AlphaZero. These strong programs provide human players with opportunities to improve their skills. However, human players may not enjoy such strong opponents. To make middle-level players learn from playing good-quality games with strong programs, we have proposed to combine programs with distinct roles in our previous research. One role is a superhuman program that proposes and accurately evaluates candidate moves. The other role is a naturalness (or human likeness) evaluator. Candidate moves are evaluated by combining the two roles using a function, and the moves with the highest scores are played. This study builds upon our earlier work to further improve the naturalness of moves. First, we propose a search mechanism inspired by the sequential halving algorithm to decide candidate moves and the moves to play. Second, we propose a new score function to address several issues of the previous approach. We conduct experiments to compare the proposed approaches with several existing approaches. The results show that the move naturalness of the proposed approaches is greatly improved and that performance in other aspects is at least as good as existing approaches.

Keywords: Teaching, Strength Adjustment, Entertaining, Go, Naturalness

1 Introduction

Artificial intelligence has made significant progress in many fields, including computer games. For a long time, one of the primary focuses of the game field has been the development of strong computer programs. With substantial advancements in deep learning techniques and many other algorithms, computer programs have now outperformed top human players in a wide range of games. A well-known example is AlphaZero [1], which masters chess, shogi, and Go through self-play games without requiring domain-specific knowledge from human experts.

These strong programs present opportunities for human players to acquire new skills; however, human players may not enjoy playing against and learning from these programs. For one reason, these programs are too strong for human players to defeat. When losing constantly, human players may be frustrated and less motivated for continuous gameplay. For another reason, computer programs sometimes play moves that are beyond expectations. Amateurs or even experts may not understand the logic behind such moves, making it challenging for human players to learn from these strong programs.

If these strong programs can *teach* as human teachers do, amateurs will have an additional option to improve their skills, which is more accessible than hiring a teacher or attending classrooms in terms of expense. There are several methods for human teachers to teach their students, such as showing good examples, pointing out bad moves, or playing *teaching games* [2]. In the following, we discuss the case of playing teaching games in a board game, Go, where the teaching game culture also exists in other games such as shogi and Chinese chess. The solid-lined dialog in the background part of Fig. 1 illustrates what human teachers typically consider when playing teaching games. A teaching game can be further divided into two phases: playing a good-quality game and reviewing the game. In this paper, we focus on the former phase and leave the latter phase as future research.

Regarding good-quality games, we have discussed four critical factors in our previous work [3], where human teachers aim to achieve these factors implicitly or explicitly.

(a) playing balanced games with the students
Go teachers are usually stronger than their students, and it is easy for the teachers to defeat their students. However, the teachers avoid winning against the students always. (a1) Sometimes, the teachers win, while at other times, the teachers intentionally let the students win. Also, in each game, the teachers try to avoid winning or losing by a large margin.

(b) reflecting the goodness and badness of the students' moves in the final wins and losses
Related to (a1), the teachers sometimes intentionally let the students win. To make the students remember good and bad moves clearly with strong impressions, the teachers try to associate the final wins and losses with the students' moves' goodness and badness. For example, when a student

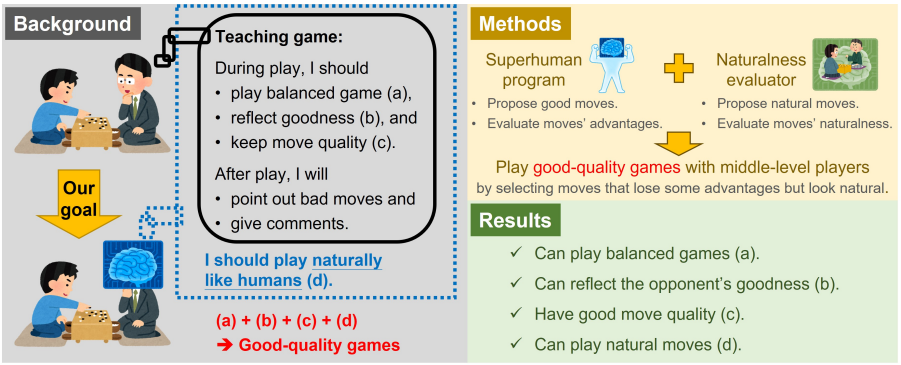


Fig. 1: An overview of this paper, where the background part illustrates teaching games by human teachers (solid-lined dialog) and strong programs (dotted-lined dialog), the method part shows the concept of our proposed approach, and the results part summarizes the findings

plays many good moves during a game, the teachers tend to let the student win as a kind of reward.

(c) keep the quality of their (teachers') own moves as good as possible
Related to (a), the teachers need to put in some effort to play balanced games with the students, which can be achieved by giving handicap stones, intentionally playing some bad moves to lose advantages, or doing both. Giving handicap stones occurs before the start of the games and is beyond the scope of our consideration. As for playing bad moves, even though it is necessary, the teachers try to keep the quality of their moves as good as possible so that the students can learn good stone shapes or good move sequences by observing the teachers' gameplay.

(d) playing natural (or say human-like) moves
Playing human-like moves is not an issue to most human teachers but is a non-trivial task for computer programs. As an illustrative example, human players tend to finish a local battle before moving to another, while strong Go programs, especially AlphaZero-based programs, do not exhibit such a tendency.[1]

The evaluation metrics of these factors will be explained in Section 5. Our goal is to create computer programs capable of playing good-quality games with middle-level players using strong programs, as depicted using the dotted-lined dialog in the background part of Fig. 1. More specifically, we target

[1]This is widely known among Go players and has been verified by us with necessary data. An average Euclidean distance for a KataGo's move to its previous move was 4.649, while that of middle-level human players was 3.426. KataGo used 1,000 simulations per move, where the program version was 20d34784703c5b4000643d3ccc43bb37d418f3b5 and the neural network version was kata1-b40c256-s9948109056-d2425397051. The middle-level human players' data is from Table 4.

Go players ranging from kgs8k to kgs2d[2], which covers approximately 80% of active players on the KGS Go Server.

Creating programs that can achieve a single factor is not difficult; however, considering all four factors at once is a challenging task. For example, to achieve (a1), a naïve method can be employed, which selects moves with win rates closest to 50%. However, this method will select a bad move when the opponent makes a mistake in order to cancel the advantage; the method plays well when the opponent plays well. Therefore, the goodness and badness of the opponent's moves are hardly reflected in the final wins and losses, failing to achieve (b). Also, moves with win rates closest to 50% may not have good shapes and good flows nor be human-like, failing to achieve (c) and (d).

In our previous work [3], we proposed playing good-quality games with middle-level players by combining programs with two different roles. The first role employed a superhuman program to generate candidate moves and accurately evaluate how advantageous each move is. The second role employed a neural network trained using strong human players' games to evaluate the naturalness (human likeness) of moves. By combining the statistics from both roles using a function, we calculated the scores of candidate moves and selected the one with the highest score.

In this paper, we aim to further improve the naturalness of the moves, where the naturalness is measured via a neural network trained using strong human players' games, which has been shown to reflect experts' evaluations [3]. In our approaches, first, we propose a search mechanism inspired by the sequential halving algorithm [4, 5]. Second, we replace the move score calculation with a smoother and situation-based function. Experiments show that the naturalness of the proposed approaches' moves is greatly improved (factor (d)) while the other factors (a)–(c) remain at the same level or have slightly improved compared with existing approaches.

The rest of this paper is organized as follows. Section 2 provides background information. Section 3 reviews our previous work [3] and discusses some issues with the approach. Section 4 presents our approaches, and the experiments are shown in Section 5. Finally, Section 6 makes concluding remarks.

2 Background

In this section, Subsection 2.1 introduces several representative works that have developed strong Go programs, and Subsection 2.2 reviews several works that tried to create human-centered opponent programs.

2.1 Strong Go Programs

Go is a two-player zero-sum perfect-information deterministic board game with a long history. When played on 19×19 boards, the game-tree complexity

[2]KGS is an online Go server for players (including humans and computer programs) to play Go, which also assigns rankings to the players, https://www.gokgs.com.

is about 10^{360} [6], which is considerably complex. Creating strong Go programs was considered one of the most challenging research topics in the field of artificial intelligence.

The first significant milestone was reached in the late 2000s when Monte-Carlo tree search (MCTS) was applied to Go [7]. Since then, most Go programs employed MCTS, and many approaches were proposed to further improve the programs' strength. For example, some researchers incorporated knowledge learned from human games into MCTS [8, 9]. Strong MCTS-based Go programs achieved the skill levels of high-ranking amateur human players.

The second significant milestone was reached in the mid-2010s when deep learning was employed. In 2016, AlphaGo [10] became the first Go program that defeated a top-level professional player. AlphaZero [1] was a successor of AlphaGo. A big difference between AlphaGo and AlphaZero was that AlphaZero did not use human games in its learning. Instead, AlphaZero learned from self-play games. Several open-source projects tried to create strong Go programs based on AlphaZero-like approaches, including ELF OpenGo [11], Leela Zero [12], and KataGo [13].

2.2 Player-Centered Programs

In addition to creating strong game-playing programs, considerable effort has been put into developing human-player-centered programs for teaching and/or entertaining human players. For this purpose, an important direction is to make the games have proper difficulty or make the opponent programs have proper strength. Playing too-difficult games or playing against too-strong programs is frustrating, while playing too-easy games or playing against too-weak programs is boring; both make human players lose motivation to play the games. Making games have proper difficulty (often referred to as *dynamic difficulty adjustment*) has been widely studied in various games, such as horror game [14] and vocabulary learning game [15].

To create opponent programs with proper strength, some researchers intentionally weakened strong programs. Specifically, Sephton et al. [16] investigated several ways of move selection in MCTS, among which one selected moves based on a softmax policy: select move i with a probability of $(n_i)^z / \Sigma_j (n_j)^z$, where n_i is move i's visit count and z a strength-controlling parameter. However, a problem with this approach was that very bad moves had chances to be selected as long as their visit counts were not 0. To address this problem, Liu et al. [17] introduced a threshold for visit counts. Assuming that the move with the highest visit count was visited n_{max} times during MCTS, moves with $n_i < n_{max} \times R_{th}$ were excluded from consideration.

In addition to having proper strength, some researchers further utilized human games in their programs to achieve human likeness. For example, Nakamichi and Ito [18] used shogi amateur games to train evaluation functions. They then replaced the evaluation functions in a strong program with the trained ones to create weaker but human-like programs. Rosemarin and

Rosenfeld [19] proposed a variant of MCTS by introducing several parameters. They then used chess amateur games with different rating ranges to tune the introduced parameters so that the programs achieved the corresponding strength of the given rating range.

As another way to use human players' games, McIlroy-Young et al. [20] trained neural networks to predict chess amateur moves. They separated chess amateur games according to the players' ratings and trained a neural network for each of the rating ranges [1100, 1199], [1200, 1299], and so on. When predicting chess amateurs' moves in a specific rating range, their experiments showed that the neural network trained using the corresponding rating range performed the best. They also analyzed moves played by AlphaZero-based programs and concluded that the moves were quite different from human players. To further improve the prediction accuracy, Jacob et al. [21] and Baier et al. [22] found that incorporating neural networks into MCTS was an effective way, while the programs' strength was also improved.

Some researchers employed game-specific knowledge instead of human games to achieve human likeness. For example, Shi et al. [23] employed the Euclidean distance between moves and the previous moves in their Go program. Moon et al. [24] in their work on fighting game estimated players' affective states by machine learning models and incorporated the estimations into MCTS.

In our previous work [3], we proposed to combine programs with different roles, which will be presented in more detail in the next section.

3 Previous Work and Its Issues

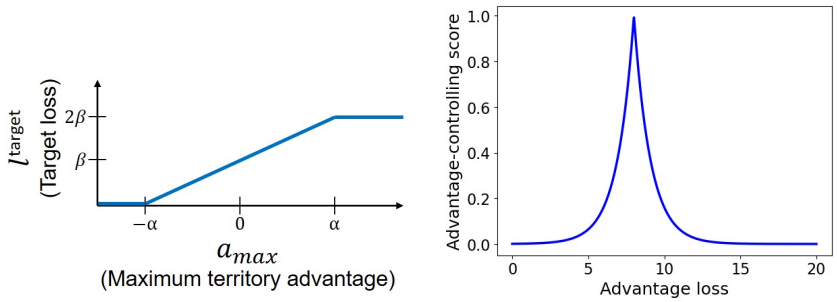
In this section, Subsection 3.1 reviews our previous work [3], and then Subsection 3.2 discusses the issues of the approach.

3.1 Combining Programs with Different Roles

To play good-quality games with middle-level players, as described in Section 1, we proposed to combine programs with different roles [3]. In more detail, we employed two programs, (1) a superhuman program that proposed candidate moves and accurately evaluated each move's degree of advantage and (2) a naturalness evaluator that informed us how natural each candidate move looks.

In the implementation for Go, we employed the AlphaZero-based superhuman program KataGo [13] as the first role and a neural network trained using strong human players' games[3] as the second role. In the rest of this paper, the policy (probability distribution of moves) from the neural network is denoted as π_{human}. We consider that both roles are necessary for the following reasons.

[3] https://sjeng.org/zero/best_v1.txt.zip, released along with the Leela Zero project [12]. The input of the network contains 17 binary planes of size 19×19, where 16=8×2 planes represent the black and white stones (×2) for the 8 most recent board states and the remaining 1 plane to indicate which player is going to play. The output of the network contains a 362-dimensional policy for possible moves (19×19 intersections + PASS) and a 1-dimensional value to predict the degree of advantage of the player to move. The network was trained using strong human players' games that were openly available, referring to https://github.com/leela-zero/leela-zero/issues/628.



(a) (b)

Fig. 2: (a) Visualization of Eq. (1) showing the relation between a_{max} and l^{target} with two parameters α and β and (b) the advantage-controlling score in Eq. (2) (i.e., the second term)

First, good moves from KataGo's view may look strange to human players. McIlroy-Young et al. [20] also showed that an AlphaZero-based chess program played quite differently from humans. Second, programs trained using human games may not accurately evaluate board states or moves because they are biased by human data. Therefore, it is reasonable to separate different roles into different programs.[4]

Algorithm 1 shows the move decision procedure. The target loss l^{target} in step 3 was defined as follows[5]:

$$l^{\text{target}} = \begin{cases} 0, & \text{if } a_{max} \leq -\alpha \\ (1 + \frac{a_{max}}{\alpha}) \times \beta, & \text{if } -\alpha < a_{max} \leq \alpha \\ 2 \times \beta, & \text{if } \alpha < a_{max} \end{cases} \tag{1}$$

where α is a positive number deciding the range to tune l^{target} finer (a higher α makes the target loss depend less on the current situation), and β decides the degree of the loss (a higher β tries to reduce the strength more), which is depicted in Fig. 2a. When the program has an advantage, the target loss l^{target} is high, making the program more likely to select moves with high losses. On the contrary, when the program has a disadvantage, it prefers moves that lose few or no territories.

Based on l^{target}, the score s_i considering both strength and naturalness of move i was defined as follows:

$$s_i = (p'_i + \epsilon) \times \gamma^{e_i} \tag{2}$$

[4]For games other than Go, we consider that the idea of separating roles is also applicable. Taking chess as an example, one may use Stockfish [25] as the superhuman role and Maia [20] as the naturalness evaluator.

[5]The term and notation used in our previous work were *ideal loss* and l^*. However, the word "ideal" was misleading, and "target" is more proper.

where p'_i is move i's probability from π_{human}, ϵ a small number (say 10^{-3}) to prevent the first term from being 0, e_i move i's error of loss compared to the target loss (i.e., $|(a_{max} - a_i) - l^{\text{target}}|$), and γ a coefficient (say 0.4) deciding the importance of e_i.

The first term (p'_i) is the naturalness score (for factor (d)), and the second term is the advantage-controlling score (for factors (a)–(c)). Assuming that l^{target} is 8, the curve in Fig. 2b depicts the advantage-controlling score in Eq. (2). Moves with advantage losses close to the target loss receive higher advantage-controlling scores, where the target loss is decided by the current territory advantage. While trying to play balanced games with the opponent (factor (a)), the program tries to avoid moves that lose big advantages (factor (c)). When the program has an advantage, it does not try to immediately reduce the advantage to 0. Instead, it tries to play natural moves with proper advantage losses. By doing so, the advantage control is done in a long term during the game, making it possible to reflect the opponent's moves' goodness (factor (b)).

Algorithm 1 The move decision procedure in our previous work [3]

Require: A board state, #simulation N_{sim}, parameters α, β, γ and ϵ for Eqs. (1) and (2)
Ensure: A move to play

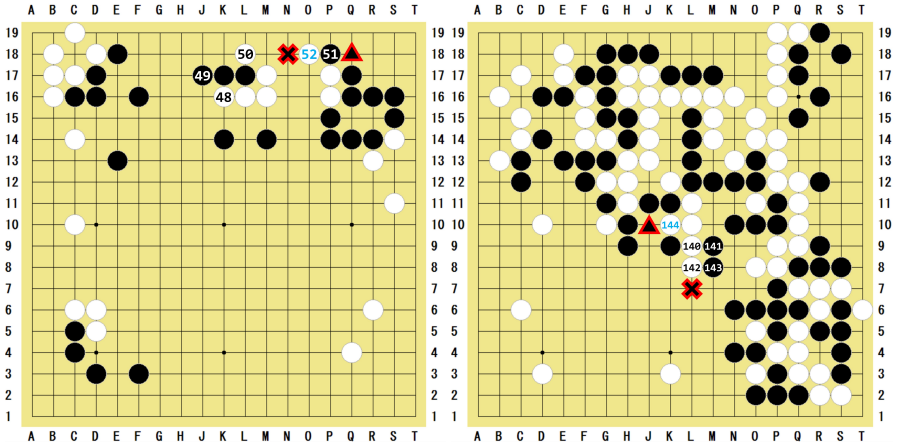
// Step 1: do KataGo's MCTS
Let KataGo search using N_{sim} simulations to obtain a set of candidate moves M
Let n_i and a_i denote move i's visit count and territory advantage, respectively

// Step 2: filter out obviously bad moves
Delete move i from M where $n_i < 10$ // to have reliable statistics
$a_{max} = \max_{i \in \{j | n_j > 10\}} a_i$ // i.e., maximum territory advantage
Delete move i from M where $a_{max} - a_i > 20$

// Step 3: calculate target advantage loss l^{target}
Calculate l^{target} using Eq. (1)

// Step 4: calculate each move's score s_i
Calculate s_i using Eq. (2)

// Step 5: play the move with the highest s_i
return $\text{argmax}_{i \in M} s_i$



(a) (b)

Fig. 3: Examples showing that natural-looking moves are ignored by KataGo's MCTS, where triangles indicate the natural-looking moves and crosses the played moves

The experiments targeted three middle-level programs, GNU Go[6] level 10 (˜kgs8k) and Pachi[7] with settings of kgs3k and kgs2d. Several approaches (e.g., [17, 20]) were compared, and the results showed that Algorithm 1 was the most promising to play good-quality games with these middle-level players.

3.2 Remaining Issues

Despite the generally good results, several issues remained, and two major issues are discussed as follows. First, KataGo's MCTS usually ignores acceptable bad moves that are natural-looking. Fig. 3 shows two examples. In Fig. 3a, it is the black player's turn, and playing at Q18 (the triangle mark) to connect black stones is natural. π_{human} gave a probability of 54.8% to play this move, which is relatively high. However, during KataGo's MCTS with $N_{sim} = 1000$, the move Q18 was visited only 3 times and was deleted in step 2. From KataGo's view, playing at R9 would obtain a territory advantage of +8.74 points, better than Q18 whose advantage was +6.16 points. With an advantage loss of 2.58 points, spending too many simulations on move Q18 was wasteful in terms of being a strong player. Thus, it was reasonable that the visit count of move Q18 was low. Among moves with visit counts higher than or equal to 10, our previous approach played at N18 (the cross mark), with an advantage of +7.43 points from KataGo and a probability of 7.2% from π_{human}. Move N18 is an advanced move that high-level players might play but may be difficult for middle-level players.

[6] https://www.gnu.org/software/gnugo/
[7] https://github.com/pasky/pachi

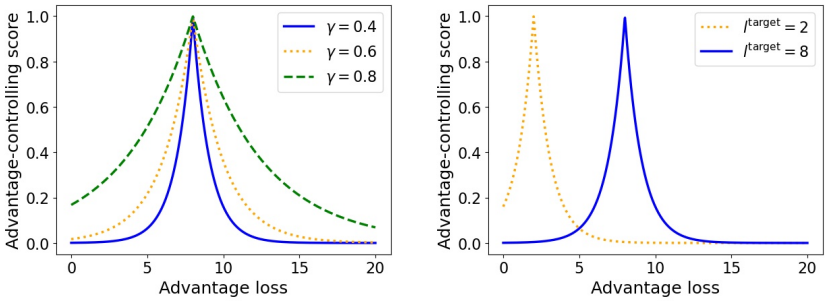
In Fig. 3b, it is the black player's turn, and it is natural to play at J10 (the triangle mark) to save the two black stones at J11 and K11. π_{human} gave a probability of 88.2% to play this move, which is considerably high. From KataGo's view, playing at M7 seemed the best, where the territory advantage was +15.88 points. In contrast, playing at J10 obtained +12.13 points, which was visited only 8 times during MCTS. Among moves with visit counts higher than or equal to 10, our previous approach played at L7 (the cross mark), with an advantage of +13.49 points from KataGo and a probability of 2.7% from π_{human}. Move L7 gives the black player a better shape in the bottom-right corner, which is worth giving up the two black stones at J11 and K11. However, for middle-level players, it may look more natural to save the two stones.

To achieve a superhuman level, it is not problematic to ignore bad moves even when they are natural-looking, such as Q18 in Fig. 3a and J10 in Fig. 3b. However, to play good-quality games with middle-level players, these moves are worth considering. If such natural-looking but acceptable bad moves can be included in the candidates, we expect the approach to play more natural moves and be easier to play balanced games.

The second issue lies in the advantage-controlling score in Eq. (2) (i.e., the second term), which can be discussed from two aspects:

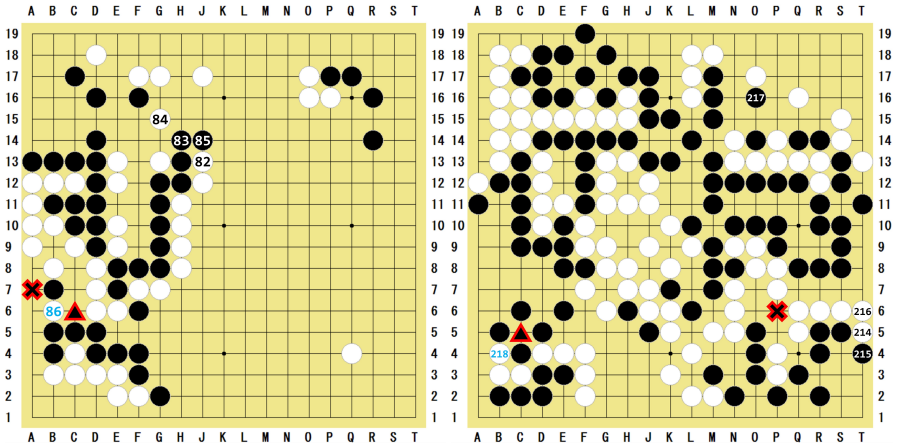
- As shown in Fig. 2b, the curve is sharply centered at the target loss l^{target}, meaning that the score drops drastically as the advantage loss gets away from the target loss. Although increasing the γ value can alleviate the problem, it introduces another problem that the width of the mountain shape becomes too wide, as shown in Fig. 4a (e.g., $\gamma = 0.8$). With wider mountain shapes, moves with high advantage losses receive lighter penalties and are more likely to be played, which should be avoided.
- Regardless of the value of the target loss l^{target}, the mountain width is the same as long as γ is the same, as shown in Fig. 4b. However, according to different situations, the same error $|(a_{max} - a_i) - l^{\text{target}}|$ has totally different effects. For example, an error of 2 is big for $l^{\text{target}} = 2$ (i.e., 2±2) but is moderate for $l^{\text{target}} = 8$ (i.e., 8±2). The mountain width should vary according to the target loss, and lower target losses should have narrower mountain widths.

The second issue makes the approach likely to select moves with losses close to the target loss and ignore naturalness, where Fig. 5 shows two examples. In Fig. 5a, it is the black player's turn, and it is natural to play at C6 (the triangle mark) to push through the white stones at B6 and D6. π_{human} gave a probability of 64.2% to play this move, which is fairly high. This board state's territory advantage was +27.93, the same as move C6's. Namely, the advantage loss of move C6 was 0. Assuming $\alpha = 25$ and $\beta = 3$, the target loss l^{target} was $2 \times \beta = 6$. With $\gamma = 0.4$, the score of move C6 was $(0.642 + 10^{-3}) \times 0.4^{|0-6|} = 2.6 \times 10^{-3}$. Another candidate move, A7, obtained +26.40 points of advantage according to KataGo. π_{human} gave a probability



(a) (b)

Fig. 4: The advantage-controlling score in Eq. (2) with different γ and l^{target}: (a) $\gamma \in \{0.4, 0.6, 0.8\}$ with $l^{\text{target}} = 8$ and (b) $\gamma = 0.4$ with $l^{\text{target}} \in \{2, 8\}$



(a) (b)

Fig. 5: Examples showing that naturalness is sacrificed to make advantage loss close to the target loss, where triangles indicate the natural-looking moves and crosses the played moves

of 19.8% to play this move, which looks less natural than C6. The difference in territory advantages between these two moves was 1.54. At a board state with an advantage of +27.93, losing only 1.54 points is minor, which is not worth sacrificing naturalness too much. However, move A7 had a score of $(0.198 + 10^{-3}) \times 0.4^{|1.54-6|} = 3.3 \times 10^{-3}$ and was played.

In Fig. 5b, it is the black player's turn, and it is natural to play at C5 (the triangle mark) to connect the neighboring black stones. π_{human} gave a probability of 95.2% to play this move, which is considerably high. This board state's territory advantage was +25.18 points by playing at P17 to

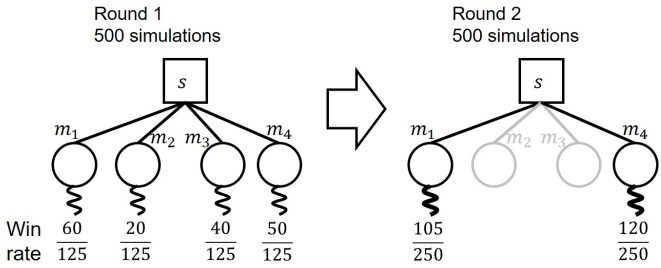


Fig. 6: An example of the sequential halving algorithm

invade the top-right corner. Meanwhile, the territory advantage of move C5 was +7.79 points, meaning that it had an advantage loss of 17.39 points. Since the advantage loss was big, the move score was extremely low, which was $(0.952 + 10^{-3}) \times 0.4^{|17.39-6|} = 2.8 \times 10^{-5}$. Another candidate move, P6, obtained +21.60 points of advantage according to KataGo (i.e., an advantage loss of 3.58). π_{human} gave a probability of 0.007% to play this move, which looks unnatural. Also, playing at P6 wastes a ko threat, which is meaningless and should be avoided when playing teaching games. However, move P6 had a score of $(0.00007 + 10^{-3}) \times 0.4^{|3.58-6|} = 1.2 \times 10^{-4}$ and was played.

4 Approaches

To solve the issues mentioned in Subsection 3.2, we propose two solutions. First, we replace KataGo's MCTS (Step 1 in Algorithm 1) with a search mechanism inspired by the sequential halving algorithm [4, 5], which will be presented in Subsection 4.1. Second, we replace the calculation of the advantage-controlling score in Eq. (2) with a situation-based smoother function, which will be presented in Subsection 4.2.

4.1 Sequential Halving for Playing Good-Quality Games

Because our goal is to play good-quality games with middle-level players, it may be improper to employ KataGo's MCTS to propose candidate moves. KataGo's MCTS tries to find optimal moves; thus, it is reasonable to spend no or only a few simulations on moves with small territory advantages. However, such moves may be suitable for playing with middle-level players. With too few simulations, these moves have unreliable statistics and are deleted from candidates (Algorithm 1 line 5), making these moves have no chance of being selected.

The sequential halving algorithm [4] divides a given number of simulations into several rounds and searches moves with the same number of simulations in each round. The goal is to identify the best move to play using the given simulation budget. Fig. 6 illustrates a simple example, where board state s has 4 moves m_1–m_4. As the name implies, the worse half of the moves are deleted in

each round, so the search is divided into $R = \log_2 4 = 2$ rounds. Assuming that the simulation budget N_{sim} is 1,000, each round uses $N_{sim}/R = 1000/2 = 500$ simulations. In round 1, the 500 simulations are evenly distributed to the 4 moves m_1–m_4. Based on the win rates, the worse half of the moves (i.e., moves m_2 and m_3) are deleted. Similarly, in round 2, the remaining 500 simulations are distributed to the remaining 2 moves, and the worse move (i.e., m_1) is deleted. Finally, the remaining move (i.e., m_4) is played.

Fabiano and Cazenave [5] presented a more general form of sequential halving. In each round, a fraction $\lambda \in (0, 1)$ of moves remains, and the rest $(1 - \lambda)$ are deleted. Namely, the original sequential halving is a special case with $\lambda = 1/2$. In addition, they empirically compared *restart* and *stockpile*. Restart means that statistics from previous rounds are not brought to later rounds. An example has been shown in Fig. 6, where moves m_1 and m_4's statistics in round 1 are ignored in round 2. On the contrary, to gather more accurate statistics, stockpile includes the statistics obtained from the previous rounds when calculating win rates. With stockpile, the win rates of moves m_1 and m_4 in round 2 become (60+105)/(125+250) and (50+120)/(125+250), respectively. Their experiments showed that stockpile obtained better results than restart.

Inspired by sequential halving, we propose a new search mechanism for playing good-quality games with middle-level Go players, as presented in Algorithm 2.

First, we need to decide a set of candidates moves to search (lines 1–2). Straightforwardly, we can search all legal moves. However, since each board state in Go has many legal moves (≈250 [10]), it is time-consuming to search all. To address this, we consider only moves that satisfy one of the following three conditions:

- When sorting moves according to π_{human} (probabilities of moves predicted by a neural network trained using human games), the move is within the top 25.[8]
- When sorting moves according to π_{KataGo} (probabilities of moves predicted by KataGo's policy network), the move is within the top 5.
- The move is a pass (i.e., not placing stones on the board).

π_{human} provides us natural-looking moves, and π_{KataGo} provides us good moves. Since moves from π_{KataGo} often overlap with those from π_{human}, we have approximately 26 candidate moves. The necessity of including moves from π_{KataGo} will be explained in Subsection 4.1.1.

With the set of candidate moves M_0, round 1 (lines 3–10) uses $N_{sim}/2$ simulations to investigate these moves and decides 4 moves to remain. More specifically, the 4 moves that have the top-4 move scores remain. Different from sequential halving, we select moves according to the move scores instead of the

[8] We confirmed that moves within the top 25 generally contained those possibly considered by human players. We also confirmed that moves outside the top 25 generally need not be considered, though in some rare cases, some critical moves are not included in the top 25, as shown in Subsection 4.1.1. The number 25 resulted in a good balance between strength adjustment and thinking time.

Algorithm 2 Our proposed search mechanism

Require: A board state, #simulation N_{sim}, parameters α, β, γ, and ϵ for Eqs. (1) and (2)

Ensure: A move to play

1: // Initialization: Decide candidate moves M_0 to search
2: $M_0 \leftarrow \{\text{top 25 from } \pi_{\text{human}}\} \cup \{\text{top 5 from } \pi_{\text{KataGo}}\} \cup \{\text{PASS}\}$

3: // Round 1: Determine 4 moves to remain
4: // - Use approximately $N_{sim}/2$ simulations to investigate $\forall i \in M_0$
5: $N_1 \leftarrow \lfloor (N_{sim}/2)/|M_0| \rfloor$
6: Do KataGo's MCTS (N_1 simulations) on each move $i \in M_0$
7: Assume that N_{sim} is high enough s.t. $N_1 \geq 10$
8: Calculate move scores using Algorithm 3 with $M_{\text{considering}}$ being M_0
9: // - Select 4 moves with the top-4 scores
10: $M_1 \leftarrow \{i \in M_0 \mid s_i \text{ is within top 4}\}$

11: // Round 2: Investigate M_1 further
12: // - Use approximately $N_{sim}/2$ simulations to investigate $\forall i \in M_1$
13: $N_2 \leftarrow \lfloor (N_{sim} - N_1 \times |M_0|)/|M_1| \rfloor$
14: Do KataGo's MCTS (N_2 simulations) on each move $i \in M_1$
15: Update the 4 moves' statistics based on the new results
16: Calculate move scores using Algorithm 3 with $M_{\text{considering}}$ being M_1
17: Note 1: the maximum territory advantage may change because of the search in round 2
18: Note 2: the maximum territory advantage is obtained from M_0 instead of M_1

19: // Special situation: do extra simulations
20: **if** $\max_{i \in \{M_1\}} s_i < \max_{i \in M_0 \setminus M_1} s_i$ **then**
21: // i.e., moves in M_1 turn out to have lower scores than deleted moves
22: Do N_2 simulations on each move $i \in M_0 \setminus M_1$
23: Update the moves' statistics based on the new results
24: Calculate move scores using Algorithm 3 with $M_{\text{considering}}$ being M_0
25: **end if**

26: // Decision: play the move with the highest s_i
27: **return** $\text{argmax}_{i \in M_0} s_i$

win rates. The move score calculation on line 3 of Algorithm 3 can be replaced with other functions, such as the one that will be presented in Subsection 4.2.

With the shrunk set of candidate moves M_1 ($|M_1| = 4$), round 2 (lines 11–18) uses the remaining simulation budget to investigate the remaining 4 moves. The move statistics (e.g., territory advantage) from round 2 are merged with those from round 1, weighted by the visit counts. Reusing the results in round 1

Algorithm 3 Score calculation in our proposed search mechanism

Require: A set of moves considering $M_{\text{considering}}$, parameters α, β, γ and ϵ for Eqs. (1) and (2)

1: $a_{max} = \max_{i \in \{M_0\}} a_i$ // i.e., maximum territory advantage
2: Calculate l^{target} using Eq. (1)
3: Calculate each move $i \in M_{\text{considering}}$'s score s_i using Eq. (2)
4: But move i losing more than 20 points is not considered ($s_i = -\infty$) as Algorithm 1

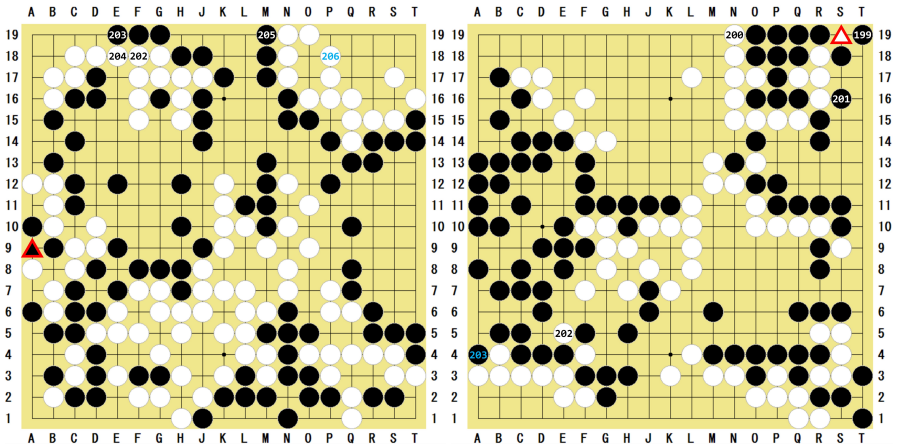
is similar to stockpile. Move scores of $i \in M_1$ are then calculated similarly to round 1. Note that when calculating the maximum territory advantage, we check all candidates $i \in M_0$'s territory advantages instead of only checking M_1. The move with the maximum territory advantage may not be included in M_1 because it is unpromising for playing good-quality games with middle-level players. However, including such moves is crucial for accurately evaluating the board states' degrees of advantage and other moves' advantage losses.

The following explains a minor problem that may happen at the end of round 2 and a solution to the problem (lines 19–25). For some complicated board states, the territory advantages may be overestimated when the number of simulations is low (round 1). If a move leading to such a board state remains in round 2 and receives more simulations, it turns out that this move has a very low score due to a big territory loss. The problem is that when all 4 moves in round 2 turn out to have scores lower than the deleted moves' scores, it is strange to select a move from the 4 in round 2. For example, it is possible that all 4 moves in M_1 turn out to have territory losses higher than 20 points and have $s_i = -\infty$.

In such a situation, we consider the given board state unstable and propose to do the same number of simulations (i.e., N_2) on the deleted moves $M_0 \setminus M_1$. The concept of searching more when unstable is similar to quiescence search [26]. Although the extra simulations increase the time cost, we argue that such situations rarely happen, approximately 0.2-0.3 times per game, and that the overhead can be neglected.

4.1.1 Including Moves from KataGo' Policy Network

It is necessary to include moves from π_{KataGo}, where Fig. 7 shows two examples. In Fig. 7a, it is the black player's turn, and playing at A9 is the only move that can capture white stones in the middle-left area. The territory advantage of this move was +11.91, and others' were lower than −13. When playing teaching games, move A9 should be played to let students understand that the white stones would be captured. However, π_{human} gave a probability of 0.1% to play this move, which was ranked as the 26th highest. In contrast, π_{KataGo} gave a probability of 89.9% to play this move and helped the approach avoid overlooking move A9.



(a) (b)

Fig. 7: Examples showing that including the top-5 moves from π_{KataGo} is necessary, where triangles indicate the only moves that should be played

In Fig. 7b, it is the white player's turn, and playing at S19 is the only move that has a chance of winning the game. Capturing the black stones in the top-right corner also looks natural for human players. The territory advantage of this move was +21.22, and others' were lower than −1. π_{human} gave a probability of 0.05% to play this move, which was ranked as the 27th highest. π_{KataGo} gave a probability of 99.5% to play this move and helped the approach avoid overlooking move S19.

4.2 Situation-Based Smoother Advantage-Controlling Scores

As discussed in Subsection 3.2, the advantage-controlling score in Eq. (2) has two problems. First, the score drops drastically as the advantage loss gets away from the target loss. Moves with advantage losses around the target loss should receive higher scores. Second, the mountain shape has the same width no matter how large the target loss is. The mountain shape should become narrower for lower target losses. Examples in Fig. 5 have shown that the advantage-controlling score in Eq. (2) influences the overall score s_i a lot, which tends to sacrifice moves' naturalness.

To improve the move naturalness by addressing the two mentioned problems, we propose the following equation to calculate move i's score s_i^{new}:

$$s_i^{new} = (p_i' + \epsilon) \times \frac{1}{\left(\frac{(e_i)^2}{l^{\text{target}}+1} + 1\right)^{\tau}} \tag{3}$$

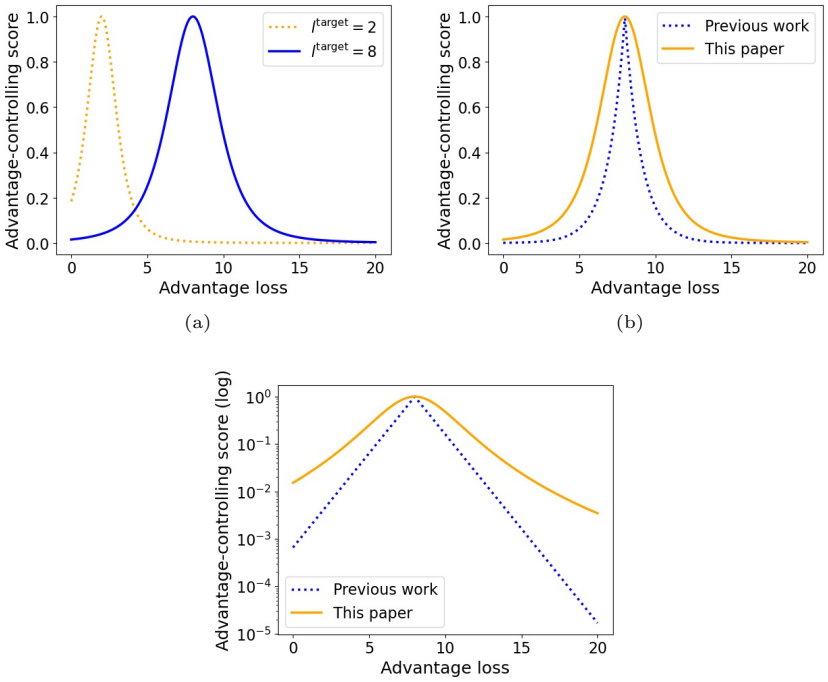


Fig. 8: The advantage-controlling score in Eq. (3): (a) comparing the cases of different target losses $l^{\text{target}} \in \{2, 8\}$ and comparing with Eq. (2) (blue dotted curve) where the y-axis is in (b) the normal scale and (c) the log scale

where p'_i, ϵ, and e_i are the same as Eq. (2) (i.e., p'_i is move i's probability from π_{human}, $\epsilon = 10^{-3}$ a small number to prevent the first term from being 0, and $e_i = |(a_{max} - a_i) - l^{\text{target}}|$ move i's error of loss compared to the target loss l^{target}), and τ is a coefficient (say 2) deciding the importance of e_i. With a higher τ, the score drops more drastically as e_i increases. The target loss l^{target} is involved in the score calculation in a way that lower target losses have a narrower mountain shape, as shown in Fig. 8a.

Figs. 8b and 8c depict the advantage-controlling scores in Eq. (2) and Eq. (3), assuming that the target loss l^{target} is 8. From Fig. 8b, we observe that the advantage-controlling score function in Eq. (3) is smoother than Eq. (2). From Fig. 8c (log scale), we can clearly observe that when the advantage loss goes far away from the target loss l^{target}, Eq. (3) gives much higher values than Eq. (2).

When considering the examples in Fig. 5 again, Eq. (3) successfully selects natural moves. For the board state in Fig. 5a, moves C6 ($p' = 0.642$ and loss= 0) and A7 ($p' = 0.198$ and loss=1.54) were discussed in Subsection 3.2. With a target loss l^{target} of 6, the two moves' scores s_i^{new} are

$(0.642 + 10^{-3})/((0-6)^2/(6+1)+1)^2 = 1.7 \times 10^{-2}$ and $(0.198 + 10^{-3})/((1.54-6)^2/(6+1)+1)^2 = 1.3 \times 10^{-2}$, respectively. The relatively natural move C6 is selected.

For the board state in Fig. 5b, moves C5 ($p' = 0.952$ and loss= 17.39) and P6 ($p' = 0.00007$ and loss= 3.58) were discussed. With a target loss l^{target} of 6, the two moves' scores s_i^{new} are $(0.952+10^{-3})/((17.39-6)^2/(6+1)+1)^2 = 2.5 \times 10^{-3}$ and $(0.00007 + 10^{-3})/((3.58-6)^2/(6+1)+1)^2 = 3.2 \times 10^{-4}$, respectively. The relatively natural move C5 is selected.

5 Experiments

In the experiments, we compared the proposed approaches in Section 4 with several existing approaches. Subsection 5.1 presents the experiment settings, Subsections 5.2 to 5.5 show the experiment results with respect to different teaching game factors, and Subsection 5.6 makes a brief summary.

5.1 Experiment Settings

Following our previous work [3], we targeted middle-level programs and settings with well-known strength: GNU Go level 10 (˜kgs8k) and Pachi with settings of kgs3k and kgs2d. The reason for employing programs instead of human players as opponents was to collect numerous games for analysis. For existing approaches included in the comparison, we let them play against the above-mentioned middle-level programs. Each pair played 500 games, and the programs alternatively played black and white. Japanese rules with a komi of 6.5 were applied. The board size was 19×19.

As comparison baselines, we used human players' game records and 4 kinds of approaches. The human players' games were collected in the same way as our previous work [3]. We employed the Fox Go dataset[9] and identified players who played most frequently at the ranks of 5k, 1d, and 5d (corresponding to kgs8k, kgs3k, and kgs2d) to extract approximately 30 games played between those players for each rank. The 4 kinds of approaches are as follows.

- P1 (middle-level programs themselves, e.g., $\text{Pachi}_{\text{kgs3k}}$ versus $\text{Pachi}_{\text{kgs3k}}$),
- P2 (Maia-like supervised learning neural network [20])
- P3 (the softmax method [17]), and
- P4 (Algorithm 1, i.e., [3]).

For the proposed approaches in this paper, we compared

- P5 (halving with Eq. (2)) and
- P6 (halving with Eq. (3)).

P3–P6 were based on KataGo[10] with an N_{sim} of 1,000. P5 and P6 differed from P4 in the search mechanism: P4 used KataGo's MCTS to propose candidate

[9] https://github.com/featurecat/go-dataset

[10] The program version was 20d34784703c5b4000643d3ccc43bb37d418f3b5 and the neural network was kata1-b40c256-s9948109056-d2425397051.

moves while P5 and P6 used sequential halving as presented in Subsection 4.1. P6 differed from P5 in the move score function: P5 used Eq. (2) and P6 used Eq. (3).

After the games were played or collected, we employed another KataGo program to review these games to evaluate the played moves and determine the final winners. In the cases that some player resigned, we respected the decision and counted a loss for that player. Otherwise, the player with a positive territory advantage was the winner of the game.

We set a prerequisite to the six approaches so that they must have win rates of approximately 50% when playing against the middle-level programs (i.e., achieving factor (a1)). P1 needed no parameter tuning. For P2, our previous work [3] reported win rates of 19.2±3.5% and 65.8±4.2% against $\text{Pachi}_{\text{kgs2d}}$ and $\text{Pachi}_{\text{kgs3k}}$, respectively, when selecting the moves with the highest probabilities from π_{human}. Namely, P2 in our previous work was considerably weaker than $\text{Pachi}_{\text{kgs2d}}$ and slightly stronger than $\text{Pachi}_{\text{kgs3k}}$. With π_{human} alone, it is hard to create a stronger player that is suitable to $\text{Pachi}_{\text{kgs2d}}$; nevertheless, it is possible to create weaker players to fit $\text{Pachi}_{\text{kgs3k}}$ or GNU Go (˜kgs8k). More specifically, in this paper, we controlled the players' strength by selecting moves in proportion to $(p'_i)^\rho$, where p'_i is move i's probability from π_{human} and ρ a tunable parameter.[11] For P3, we used z to control the strength and set R_{th} to 0.1 following Liu et al. [17]. For P4–P6, we used β in Eq. (1) to control the strength and fixed other parameters. In more detail, we set α in Eq. (1) to 25.0, ϵ in Eqs. (2) and (3) to 0.001 for P4–P6, γ in Eq. (2) to 0.4 for P4–P5, and τ in Eq. (3) to 2 for P6.

In practice, we further introduced a trick called *optimistic komi* for P3 and P4 because they hardly lost to the middle-level programs[12]. Other approaches, including the proposed P5 and P6, did not require this trick.

Tables 1 to 3 show the parameter settings and P1–P6's win rates against GNU Go, $\text{Pachi}_{\text{kgs3k}}$, and $\text{Pachi}_{\text{kgs2d}}$, respectively, where the 95% confidence intervals are also presented. We confirmed that P2–P6 could achieve win rates close to 50% against a wide range of opponent levels (from kgs8k to kgs2d), as long as the parameters were properly set, except for P2 against $\text{Pachi}_{\text{kgs2d}}$. In the rest of this paper, we used the ρ, z, and β settings shown in Tables 1 to 3. To keep the paper simple and easy to understand, we only include the results of playing against $\text{Pachi}_{\text{kgs3k}}$ because those of GNU Go and $\text{Pachi}_{\text{kgs2d}}$ generally had similar tendencies.

[11] Another way to create weaker players is to do supervised learning on weaker players' games, as McIlroy-Young et al. [20] did. However, it is much more expensive to obtain neural networks with desired strength.

[12] With an optimistic komi of k, KataGo evaluated the advantages with k more points. Assume that we have a board state close to terminal games with two candidate moves, one leading to a win of +0.5 points and the other to a loss of −0.5 points. When doing MCTS with the normal komi, the former move's win rate is close to 100% while the latter move's close to 0%; thus, the latter move is rarely visited and is deleted due to the thresholds of visit counts (i.e., $n_{max} \times R_{th}$ for P3 and 10 for P4 (line 5 in Algorithm 1)). In the experiments, k was set to 4. For the same example, KataGo evaluates the former move as a win of +4.5 and the latter move as a win of +3.5, both with win rates close to 100%; thus, the latter move has a chance of being selected.

Table 1: Approaches P1–P6's win rates along with the 95% confidence intervals when playing against GNU Go (kgs8k) with the listed parameter settings.

	Parameter	Win Rate
P1 (GNU Go)	–	52.0±4.4%
P2 (Maia-like [20])	$\rho = 0.87$	49.0±4.4%
P3 (Softmax [17])	$z = 0.65$	49.4±4.4%
P4 (Previous work [3])	$\beta = 6.00$	53.6±4.4%
P5 (Halving with Eq. (2))	$\beta = 3.65$	53.2±4.4%
P6 (Halving with Eq. (3))	$\beta = 5.05$	54.4±4.4%

Table 2: Approaches P1–P6's win rates along with the 95% confidence intervals when playing against $\text{Pachi}_{\text{kgs3k}}$ with the listed parameter settings.

	Parameter	Win Rate
P1 ($\text{Pachi}_{\text{kgs3k}}$)	–	47.8±4.4%
P2 (Maia-like [20])	$\rho = 3.0$	52.0±4.4%
P3 (Softmax [17])	$z = -0.5$	47.4±4.4%
P4 (Previous work [3])	$\beta = 3.00$	48.4±4.4%
P5 (Halving with Eq. (2))	$\beta = 2.95$	50.4±4.4%
P6 (Halving with Eq. (3))	$\beta = 3.20$	49.2±4.4%

Table 3: Approaches P1–P6's win rates along with the 95% confidence intervals when playing against $\text{Pachi}_{\text{kgs2d}}$ with the listed parameter settings.

	Parameter	Win Rate
P1 ($\text{Pachi}_{\text{kgs2d}}$)	–	50.6±4.4%
P2 (Maia-like [20])	–	–
P3 (Softmax [17])	$z = 1.00$	50.4±4.4%
P4 (Previous work [3])	$\beta = 2.00$	50.4±4.4%
P5 (Halving with Eq. (2))	$\beta = 2.30$	52.4±4.4%
P6 (Halving with Eq. (3))	$\beta = 2.30$	50.8±4.4%

5.2 Results of Move Naturalness

Move naturalness (factor (d)) can be assessed from two perspectives: One is from teachers who can judge whether the teaching games are properly conducted, and the other is from students who will be the targets of these approaches. Following our previous work [3], we focused on evaluating the move naturalness from teachers' perspectives, leaving the consideration of students' perspectives to future research. In our previous work, we asked Go experts to evaluate the move naturalness of P1, P3, P4, and human games (Fox Go) to obtain qualitative evaluations. However, since Go experts' evaluations are not

easily available, in this work, we only employed quantitative evaluation metrics. We considered the following four metrics as appropriate because these metrics generally aligned with the experts' evaluations:

1. The Euclidean distance between a move and the previous move.
   As discussed in Section 1, human players tend to finish a local battle before going to another, which we considered can be reflected by the Euclidean distance.
2. The arithmetic average of p'_i from π_{human} for each played move i.
   Let M_{p} be the set of moves played by some approach. The arithmetic average of p'_i is $\frac{1}{|M_{\text{p}}|} \times \sum_{i \in M_{\text{p}}} p'_i$. In many cases, moves with high p'_i look natural for human players, and *vice versa*.
3. The geometric average of p'_i from π_{human} for each played move i.
   This was extended from the second metric, where we calculated the geometric average, i.e., $(\prod_{i \in M_{\text{p}}} p'_i)^{\frac{1}{|M_{\text{p}}|}}$. The reason for additionally employing the geometric average is that the arithmetic average may not well reflect moves with low p', explained as follows. Consider two groups of moves, (i) A and B and (ii) C and D, where p'_A, p'_B, p'_C, and p'_D are 0.9, 0.001, 0.4, and 0.1, respectively. Group (i)'s arithmetic average of 0.4505 is higher than group (ii)'s 0.25, which well reflects that moves likely natural to humans (i.e., A) are selected. However, the relatively high value of 0.4505 fails to reflect the selection of possibly unnatural moves (i.e., B). In contrast, the geometric average addresses this limitation, where groups (i) and (ii)'s geometric averages are 0.03 and 0.2, respectively. We consider both the arithmetic and geometric averages to be worth referring to.
4. The ratio of moves when p' higher than 0.9 were not played.
   This metric was also based on π_{human}. Moves with p' higher than 0.9 are those that middle- and high-level human players are highly likely to play. If the approaches ignored such moves and played other moves, human players might feel strange; thus, ignoring such moves should be avoided.

Table 4 shows the results of P1–P6's move naturalness. The results of the expert evaluation from our previous work [3] are in the rightmost column, showing that the quantitative metrics were generally reliable. The results of the Euclidean distance between a move and the previous move showed that P3 was notably unnatural compared to other approaches. Among the others, P6 had the highest naturalness, P2 was the second, and P5 was the third.

P2 obtained the highest values of the arithmetic and geometric averages of p', though the results were unsurprising in that π_{human} was directly employed to select moves. Excluding P2, P4–P6 were identified as the most natural. While the differences among P4–P6 were limited in the arithmetic average of p', clear distinctions emerged when considering the geometric averages. The improvement of P5 and P6 came from the fact that moves with very low p''s were played less.

To sum up, we concluded that the search mechanism inspired by sequential halving (Algorithm 2) indeed helped improve the naturalness (i.e., P5 versus

Table 4: Approaches P1–P6's move naturalness when playing against $\text{Pachi}_{\text{kgs3k}}$.

	Euclidean distance	Arithmetic average of p'	Geometric average of p'	(%) Neglecting moves with $p' > 0.9$	Expert evaluation (1 to 5) from [3]
Interpretation: In which terms is better	Closer to humans	Higher		Lower	Higher
Fox Go	3.426	0.364	0.162	0.589%	3.60
P1 ($\text{Pachi}_{\text{kgs3k}}$)	3.627	0.381	0.150	0.857%	2.90
P2 (Maia-like [20])	3.352	0.551	0.471	0.003%	–
P3 (Softmax [17])	5.860	0.253	0.041	4.429%	1.80
P4 (Previous work [3])	3.711	0.468	0.322	0.588%	3.35
P5 (Halving with Eq. (2))	3.586	0.475	0.340	0.335%	–
P6 (Halving with Eq. (3))	3.435	0.487	0.364	0.152%	–

P4). Also, the new score function Eq. (3) further improved the naturalness (i.e., P6 versus P5).

5.3 Results of Move Quality

Different from move naturalness in Subsection 5.2 which can be considered from both teachers' and students' perspectives, move quality (factor (c)) is solely considered from teachers' perspectives. In our previous work [3], we asked Go experts to evaluate the move quality of P1, P3, P4, and human games (Fox Go) to obtain qualitative evaluations. Similar to Subsection 5.2, we only employed quantitative metrics in this work since it is not easy to obtain expert evaluations. We considered the following two metrics to be reasonable, where the first one was used in our previous work and the other was newly proposed.

1. The frequency of playing moves losing more than 15 points, i.e., $a_{max} - a_i > 15.0$.
   Although Fig. 5b has shown that moves losing more than 15 points are not necessarily bad from experts' views, we kept this metric because moves losing many points are generally bad.
2. The frequency that the played moves received probabilities less than 1% from both π_{KataGo} and π_{human}.
   A move receiving a probability less than 1% from π_{KataGo} implies that the move's quality is low from a superhuman player's view. A move receiving a probability less than 1% from π_{human} implies that middle- or high-level human players prefer not to play this move. When both conditions are satisfied, the move is likely to have low quality.

Table 5 shows P1–P6's move quality evaluations. When looking at the frequency of playing moves losing more than 15 points, four distinct groups emerged: (i) P2 was obviously the worst, where two examples will be discussed

Table 5: Approaches P1–P6's move quality when playing against $\text{Pachi}_{\text{kgs3k}}$.

	(%) Moves losing >15 points	(%) Moves receiving <1% from both π_{KataGo} and π_{human}	Expert evaluation (1 to 5) from [3]
Interpretation: In which terms is better	Lower		Higher
Fox Go	7.35%	7.88%	3.30
P1 ($\text{Pachi}_{\text{kgs3k}}$)	1.87%	9.52%	3.00
P2 (Maia-like [20])	3.32%	0.005%	–
P3 (Softmax [17])	0.22%	22.72%	2.40
P4 (Previous work [3])	0.28%	0.51%	3.50
P5 (Halving with Eq. (2))	0.09%	0.29%	–
P6 (Halving with Eq. (3))	0.21%	0.07%	–

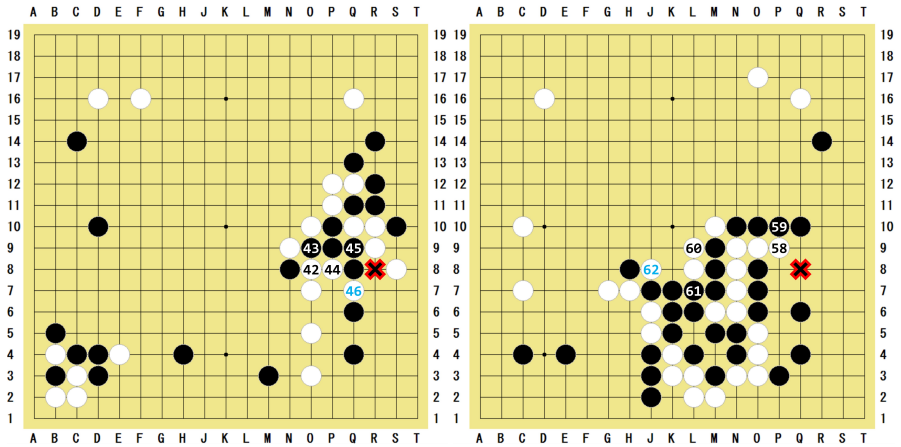
in Subsection 5.3.1. (ii) P1 was the next worst, as expected. (iii) The next ones were P3, P4, and P6. (iv) P5 played such moves the least frequently.

The reason why P5 played fewer moves with big losses than P4 is explained as follows. P5 differed from P4 in the search mechanism (i.e., sequential halving versus MCTS). When KataGo found that some move seemed bad in MCTS, it would then spend only a few or no more simulations on this move. With only a few simulations, the move's advantage might be overestimated, making the advantage loss smaller than what it should be[13]. For example, a move that should have an advantage loss of 18 might be erroneously evaluated to have a loss of 8. Assuming the target loss is 6, P4 is likely to select this move. In contrast, as illustrated in Algorithm 2, P5 conducted N_1 simulations on each candidate move (line 5) and then N_2 simulations on 4 moves in round 2 (line 13), which reduced the possibility of overestimating moves' advantages. With the sharp score function Eq. (2), moves with big losses usually had very low scores. Thus, P5 could avoid playing such moves.

Regarding why P5 played fewer moves with big losses than P6, with the score function Eq. (3) (i.e., P6), moves with big losses had much higher scores than Eq. (2) (i.e., P5), as shown in Fig. 8c. Thus, the results were unsurprising.

In Table 5, the results of expert evaluations from our previous work [3] are in the rightmost column, aligning well with the frequency of receiving probabilities less than 1% from both π_{KataGo} and π_{human}. Namely, approaches that played such moves more often had lower move quality from the experts' views. Among P1–P4, P3 was clearly the worst, and P2 was the best. P5 and P6 further improved over P4, especially P6. From the two quantitative metrics, it was hard to tell whether P5 or P6 was better. Nevertheless, we concluded that P5 and P6 had better move quality than the existing approaches P1–P4.

[13] Conversely, moves' advantages might also be underestimated with only a few simulations, and the new search mechanism of P5 could also alleviate this problem. However, this issue does not relate to playing moves with big losses and is beyond the scope of this discussion.



(a) (b)

Fig. 9: Examples showing bad moves made by P2 (Maia-like), where crosses indicate the played moves

5.3.1 Bad Moves Played by the Maia-Like Approach

Bad moves played by P2 (Maia-like approach in this paper) generally fall into two types: (i) π_{human} assigned high probabilities to bad moves, and (ii) bad moves were occasionally selected because moves were selected in proportion to $(p')^{\rho}$. Fig. 9 shows an example of each type.

Fig. 9a shows an example of bad moves of type (i), where it is the black player's turn. After the white player played at O8 as the 42nd move, π_{human} gave a probability of 85.0% to play at O9. The black stone at O9 cut the white stones at N9 and O8, which had a good local shape when considering only the very limited area; thus, the probability of 85.0% was understandable. However, playing at O9 was a bad move since it made the black stones at P9, P10, and Q8 dead. The territory loss of this move was 31.91 points. After the white player's 46th move at Q7, the black stones at O9, P9, P10, Q8, and Q9 had no way to live (i.e., going to be captured). However, π_{human} assigned a probability of 76.4% to play at R8 as the 47th move. The advantage loss of this move was 15.71 points. Even amateur players would recognize that those black stones were dead and would play elsewhere instead of R8.

Fig. 9b shows an example of bad moves of type (ii), where it is the black player's turn. π_{human} assigned a probability of 4.7% to play at Q8 as the 63rd move. Right after the 62nd move, the move with the highest probability from π_{human} was Q9 (25.4%). The black and white stones in that area were in a capture race. Playing at Q9 could capture 7 white stones, which was the best move according to KataGo's evaluation. In contrast, playing at Q8 would cause more than 10 black stones to be captured if the white player played at K8 as the 64th move. According to KataGo's evaluation, playing at Q8 as the 63rd move

had a territory loss of 84.76 points. In this case, the bad move was occasionally selected. In either case, our approach does not play these bad moves because both the loss and naturalness of candidate moves are considered.

5.4 Results of Reflecting Goodness of the Opponent's Moves

In our previous work [3], we proposed to use the Matthews correlation coefficient (MCC)[14] of a classification task to measure whether the approaches could well reflect the goodness and badness of the opponent's moves to the final wins and losses (factor (b)). To eliminate ambiguity, we assume that the opponent (the student) is $\text{Pachi}_{\text{kgs3k}}$. MCC is commonly used to evaluate the quality of binary classifications. It ranges from -1 to 1, with 1 indicating perfect predictions.

For this metric, we first defined the *badness* of $\text{Pachi}_{\text{kgs3k}}$'s moves. To know how bad a move was, we referred to the evaluator KataGo's analysis as mentioned in the 3rd paragraph of Subsection 5.1. For a given board state in the game records, we obtained the maximum territory advantage a_{max} as Algorithm 1 (line 6). We then calculated the loss of $\text{Pachi}_{\text{kgs3k}}$'s move i using $a_{max} - a_i$. The badness of $\text{Pachi}_{\text{kgs3k}}$'s moves in a game was then defined as the average loss of its moves.

Using the badness values obtained from the 500 games played, we did simple supervised learning to predict $\text{Pachi}_{\text{kgs3k}}$'s final wins and losses in the games. Given game g's badness b_g, we predicted $\text{Pachi}_{\text{kgs3k}}$ to win if b_g was lower than some threshold $b_{th} \in \{0, 0.01, 0.02, ..., 5.98, 5.99\}$ (i.e., played well); otherwise, we predicted $\text{Pachi}_{\text{kgs3k}}$ to lose. Among the threshold values in b_{th}, we selected the one that led to the highest MCC and used the MCC value to measure whether the approaches could reflect the badness of $\text{Pachi}_{\text{kgs3k}}$'s moves to the final wins and losses. A higher MCC value indicated a better reflection of $\text{Pachi}_{\text{kgs3k}}$'s badness.

Table 6 shows P1–P6's highest MCC values. P1 and P2 were clearly the worst ones to reflect their opponent's badness in the final wins and losses. This deficiency could be attributed to P1 and P2 also making mistakes and playing very bad moves. In contrast, P3–P6 had relatively high MCC values, showing that they could well reflect the goodness and badness of the opponent (i.e., $\text{Pachi}_{\text{kgs3k}}$) to the final wins and losses. Among P3–P6, P3 and P6 were slightly better than P4 and P5.

5.5 Results of Advantage Control

For playing balanced games (factor (a)), we already set a prerequisite that the approaches should have win rates close to 50% (factor (a1)). To measure whether the approaches could play close games against middle-level players (factor (a2)), we employed the following two metrics, where the first one was used in our previous work and the other was newly proposed.

[14] https://en.wikipedia.org/wiki/Phicoefficient

Table 6: Approaches P1–P6's degrees of reflecting the goodness and badness of $\text{Pachi}_{\text{kgs3k}}$'s moves to the final wins and losses.

	MCC of predicting wins/losses from badness
Interpretation: In which terms is better	Higher
Fox Go	– (due to the small number of games)
P1 ($\text{Pachi}_{\text{kgs3k}}$)	0.240
P2 (Maia-like [20])	0.189
P3 (Softmax [17])	0.720
P4 (Previous work [3])	0.645
P5 (Halving with Eq. (2))	0.647
P6 (Halving with Eq. (3))	0.700

Table 7: Approaches P1–P6's ratios of playing close games with $\text{Pachi}_{\text{kgs3k}}$, where the 95% confidence intervals are also shown.

	(%) Games with territory difference in $[-10, 10]$	
	the 100th move	the final
Interpretation: In which terms is better	Higher	
Fox Go	17.9±14.2%	7.1±9.5%
P1 ($\text{Pachi}_{\text{kgs3k}}$)	38.0±4.3%	20.4±3.5%
P2 (Maia-like [20])	12.4±2.9%	22.0±3.6%
P3 (Softmax [17])	57.6±4.3%	53.4±4.4%
P4 (Previous work [3])	52.8±4.4%	59.4±4.3%
P5 (Halving with Eq. (2))	57.0±4.3%	54.8±4.4%
P6 (Halving with Eq. (3))	52.0±4.4%	56.6±4.3%

1. The ratios of games where the final territory differences were within -10 to 10.
2. The ratios of games where the territory differences at the 100th move were within -10 to 10.
   This metric was an extension of the first. We considered that the territory differences should be small during the whole game instead of being small only at the final. In the opening phase (say until the 50th moves), the differences are usually small; in the endgame phase (say after 200 moves), the differences are usually similar to those at the end. Therefore, we selected the middle-game phase at the 100th move.

Table 7 shows P1–P6's ratios of playing close games with $\text{Pachi}_{\text{kgs3k}}$. In both the middle and final games, P1 and P2 clearly performed the worst, while P3–P6 played approximately 50–60% close games. For P5 and P6, although the ratios did not improve compared to the existing approaches P3 and P4, they neither got worse too much.

5.6 Summary

For the proposed approaches P5 and P6, we have confirmed the following:

- the strength could be adjusted using the β parameter, where win rates close to 50% against middle-level players could be achieved (factor (a1), Subsection 5.1),
- the move naturalness was greatly improved compared to existing approaches P1, P3, and P4 (factor (d), Subsection 5.2),
- the move quality was better than existing approaches (factor (c), Subsection 5.3)
- the goodness and badness of the middle-level players' moves could be well reflected in the final wins and losses at a level comparable to existing approaches P3 and P4 (factor (b), Subsection 5.4), and
- the ratios of playing close games with middle-level players were approximately at the same level as existing approaches P3 and P4 (factor (a2), Subsection 5.5).

Regarding P2, although it generally achieved superior results of move naturalness than P5 and P6, this superiority could be primarily attributed to P2 directly using π_{human} to select moves, where three out of four metrics were based on π_{human}. Considering the remaining three factors, i.e., playing balanced games (factor (a)), reflecting the opponents' goodness and badness (factor (b)), and having good quality of moves (factor (c)), P2 was unsuitable to play teaching games compared to P5 and P6.

6 Conclusions and Future Work

As computer programs have surpassed top human players in playing skills, a promising research direction is to investigate how these strong programs can be used to help improve human players' skills. Our goal is to utilize these strong programs to play teaching games with middle-level players (analog to students), as shown in the background part of Fig. 1. In the initial phase, we focus on developing programs capable of playing good-quality games that emphasize four important factors: (a) playing balanced games with the students, (b) reflecting the goodness and badness of the students' moves in the final wins and losses, (c) keeping the quality of the played moves as good as possible, and (d) playing natural or human-like moves.

For this goal, our previous work combined two programs with different roles. The first role employed a superhuman program, KataGo, to generate candidate moves and evaluate each move's degree of advantage. The second role

employed a neural network to evaluate candidate moves' naturalness, where the neural network π_{human} was trained using strong human players' games. Candidate moves' statistics obtained from the two roles were combined using a function that considers both naturalness and advantage control. The moves with the highest scores among the candidates were then selected.

In this paper, we tried to further improve the move naturalness using two approaches. First, we proposed a search mechanism inspired by sequential halving. We decided a set of initial candidate moves according to π_{human} and π_{KataGo} (KataGo's policy network) and divided search budgets into two rounds, where the first round investigated all moves in the set of initial candidates and the second round focused on investigating four promising moves. Second, we proposed a new score function in this paper, which is smoother than that in the previous work. The new score function also considered the current situation (target advantage loss). We conducted thorough experiments to compare the proposed approaches with several existing approaches. The results showed that the move naturalness (factor (d)) was greatly improved using our proposed approaches, while the other factors (a)–(c) were at the same level or slightly improved compared to existing approaches.

Despite the generally positive results, several issues remain to be addressed. For example, when reviewing the moves played by the proposed approaches, we found that some moves had relatively high scores compared to other candidates, but almost no human teachers would play these moves. As a future research direction, we will review more game records to identify potential problems and propose new approaches. In addition, we plan to deploy our approaches on online Go servers such as KGS to play against human players for verifying whether our approaches yield good results. Another future research direction involves adapting the proposed approaches to other games such as chess and shogi.

Acknowledgments. This work was supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant Numbers JP20K12121, JP23K11381, and JP23K17021.

Declarations

Competing interests. The authors declare that they have no competing interests.

Authors contribution statement. Conceptualization: Kokolo Ikeda and Chu-Hsuan Hsueh; Methodology: Kokolo Ikeda and Chu-Hsuan Hsueh; Software: Chu-Hsuan Hsueh and Kokolo Ikeda; Validation: Chu-Hsuan Hsueh and Kokolo Ikeda; Formal analysis: Kokolo Ikeda and Chu-Hsuan Hsueh; Investigation: Chu-Hsuan Hsueh and Kokolo Ikeda; Resources: Kokolo Ikeda and Chu-Hsuan Hsueh; Data curation: Kokolo Ikeda and Chu-Hsuan Hsueh; Writing—original draft preparation: Chu-Hsuan Hsueh; Writing—review and editing: Kokolo Ikeda; Visualization: Chu-Hsuan Hsueh and Kokolo Ikeda;

Supervision: Kokolo Ikeda; Project administration: Kokolo Ikeda; Funding acquisition: Kokolo Ikeda and Chu-Hsuan Hsueh. All authors read and approved the final manuscript.

Ethical and informed consent for data used. Not applicable.

Availability of data and materials. The game records generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

References

[1] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T.P., Simonyan, K., Hassabis, D.: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science **352**(6419), 1140–1144 (2018). https://doi.org/10.1126/science.aar6404

[2] Hollosi, A., Pahle, M.: Teaching Game at Sensei's Library. https://senseis.xmp.net/?TeachingGame. Accessed: Dec. 16, 2022 (2018)

[3] Hsueh, C.-H., Ikeda, K.: Playing good-quality games with weak players by combining programs with different roles. In: 2022 IEEE Conf. on Games (CoG), pp. 612–615 (2022). https://doi.org/10.1109/CoG51982.2022.9893698

[4] Karnin, Z., Koren, T., Somekh, O.: Almost optimal exploration in multi-armed bandits. In: Proceedings of the 30th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 28, pp. 1238–1246 (2013). https://proceedings.mlr.press/v28/karnin13.html

[5] Fabiano, N., Cazenave, T.: Sequential halving using scores. In: Lecture Notes in Computer Science, pp. 41–52 (2022). https://doi.org/10.1007/978-3-031-11488-5_4

[6] van den Herik, H.J., Uiterwijk, J.W.H.M., van Rijswijck, J.: Games solved: Now and in the future. Artificial Intelligence **134**(1-2), 277–311 (2002). https://doi.org/10.1016/S0004-3702(01)00152-7

[7] Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo tree search. In: Computers and Games, pp. 72–83 (2007). https://doi.org/10.1007/978-3-540-75538-8_7

[8] Chaslot, G.M.J.B., Winands, M.H.M., van den Herik, H.J., Uiterwijk, J.W.H.M., Bouzy, B.: Progressive strategies for Monte Carlo tree search. New Mathematics and Natural Computation **4**(3), 343–357 (2008). https://doi.org/10.1142/S1793005708001094

[9] Ikeda, K., Viennot, S.: Efficiency of static knowledge bias in Monte-Carlo tree search. In: The 8th International Conference on Computers and Games (CG 2013), pp. 26–38 (2013). https://doi.org/10.1007/978-3-319-09165-5_3

[10] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T.P., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016). https://doi.org/10.1038/nature16961

[11] Tian, Y., Ma, J., Gong, Q., Sengupta, S., Chen, Z., Pinkerton, J., Zitnick, L.: ELF OpenGo: An analysis and open reimplementation of AlphaZero. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning (ICML 2019). Proceedings of Machine Learning Research, vol. 97, pp. 6244–6253 (2019). https://proceedings.mlr.press/v97/tian19a.html

[12] Leela Zero: GitHub - leela-zero/leela-zero: Go engine with no human-provided knowledge, modeled after the AlphaGo Zero paper. Accessed: Oct. 31, 2022 (2017). https://github.com/leela-zero

[13] Wu, D.J.: Accelerating self-play learning in Go. In: The 34th AAAI Conf. Artif. Intell. (AAAI-20). Workshop on Reinforcement Learning in Games (2020). https://arxiv.org/abs/1902.10565

[14] Moschovitis, P., Denisova, A.: Keep calm and aim for the head: Biofeedback-controlled dynamic difficulty adjustment in a horror game. IEEE Trans. on Games **15**(3), 368–377 (2023). https://doi.org/10.1109/tg.2022.3179842

[15] Shohieb, S.M., Doenyas, C., Elhady, A.M.: Dynamic difficulty adjustment technique-based mobile vocabulary learning game for children with autism spectrum disorder. Entertainment Computing **42**, 100495 (2022). https://doi.org/10.1016/j.entcom.2022.100495

[16] Sephton, N., Cowling, P.I., Slaven, N.H.: An experimental study of action selection mechanisms to create an entertaining opponent. In: 2015 IEEE Conf. on Comput. Intell. and Games (CIG), pp. 122–129 (2015). https://doi.org/10.1109/CIG.2015.7317939

[17] Liu, A.-J., Wu, T.-R., Wu, I.-C., Guei, H., Wei, T.-H.: Strength adjustment and assessment for MCTS-based programs [research frontier]. IEEE Comput. Intell. Mag. **15**(3), 60–73 (2020). https://doi.org/10.1109/mci.2020.2998315

[18] Nakamichi, T., Ito, T.: Adjusting the evaluation function for weakening the competency level of a computer shogi program. ICGA J. **40**(1), 15–31 (2018). https://doi.org/10.3233/ICG-180042

[19] Rosemarin, H., Rosenfeld, A.: Playing chess at a human desired level and style. In: the 7th Int. Conf. on Human-Agent Interact., pp. 76–80 (2019). https://doi.org/10.1145/3349537.3351904

[20] McIlroy-Young, R., Sen, S., Kleinberg, J., Anderson, A.: Aligning superhuman AI with human behavior. In: the 26th ACM SIGKDD Int. Conf. on Knowl. Discovery & Data Mining, pp. 1677–1687 (2020). https://doi.org/10.1145/3394486.3403219

[21] Jacob, A.P., Wu, D.J., Farina, G., Lerer, A., Hu, H., Bakhtin, A., Andreas, J., Brown, N.: Modeling strong and human-like gameplay with KL-regularized search. In: Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162, pp. 9695–9728 (2022). https://proceedings.mlr.press/v162/jacob22a.html

[22] Baier, H., Sattaur, A., Powley, E.J., Devlin, S., Rollason, J., Cowling, P.I.: Emulating human play in a leading mobile card game. IEEE Trans. on Games **11**(4), 386–395 (2019). https://doi.org/10.1109/TG.2018.2835764

[23] Shi, Y., Fan, T., Li, W., Hsueh, C.-H., Ikeda, K.: Position control and production of various strategies for game of Go using deep learning methods. J. of Inf. Sci. and Eng. **37**(3), 553–573 (2021). https://doi.org/10.6688/JISE.202105_37(3).0004

[24] Moon, J., Choi, Y., Park, T., Choi, J., Hong, J.-H., Kim, K.-J.: Diversifying dynamic difficulty adjustment agent by integrating player state models into Monte-Carlo tree search. Expert Systems with Applications **205**, 117677 (2022). https://doi.org/10.1016/j.eswa.2022.117677

[25] official-stockfish: GitHub - official-stockfish/Stockfish: UCI chess engine. https://github.com/official-stockfish/Stockfish. Accessed: Dec. 18, 2022 (2008)

[26] Beal, D.F.: A generalised quiescence search algorithm. Artif. Intell. **43**(1), 85–98 (1990). https://doi.org/10.1016/0004-3702(90)90072-8