Journal of Information Science and Engineering 37, 553-573 (2021)
DOI: 10.6688/JISE.202105_37(3).0004

Position Control and Production of Various Strategies for Game of Go Using Deep Learning Methods

Yuan Shi, Tianwen Fan, Wanxiang Li, Chu-Hsuan Hsueh and Kokolo Ikeda
School of Information Science
Japan Advanced Institute of Science and Technology
Nomi, Ishikawa, 9231292 Japan
E-mail: {shiyuan; fantianwen; wanxiang.li; hsuehch; kokolo}@jaist.ac.jp

Computer Go programs have surpassed top-level human players by using deep learning and reinforcement learning techniques. Other than the strength, entertaining Go AI and AI coaches are also interesting directions but have not been well investigated. Some researchers have worked on entertaining beginners or intermediate players. One topic is position control, aiming to make strong programs play close games against weak players. Under such a scenario, the naturalness of the moves is likely to influence weaker players' enjoyment. Another topic is producing various strategies (or preferences), which human players usually have. Some methods for the two topics have been proposed and evaluated for a traditional Monte-Carlo tree search (MCTS) program. However, there are some critical differences between traditional MCTS programs and recent programs based on AlphaGo Zero, such as LeelaZero and KataGo. For example, recent programs do not run random simulations to the ends of games in MCTS, making the existing method for producing various strategies not applicable. In this paper, we first summarize such differences and some resulted problems. We then adapt existing methods as well as propose new methods to solve the problems, where promising results are obtained. For position control, the modified LeelaZero can play gently against a weaker player (48% of wins against a weaker program, Ray). A human subject experiment shows that the average number of unnatural moves per game is 1.22, while that by a simple method without considering naturalness is 2.29. We also propose a new position control method specifically for endgames. Finally, for producing various strategies, two methods are introduced. In our experiments, center- and edge/corner-oriented strategies are produced by both methods, and human players can successfully identify the strategies.

Keywords: computer Go, position control, various strategies, entertainment, coaching, deep learning, AlphaGo Zero

1. INTRODUCTION

Playing the game of Go well requires an outstanding ability of decision making and has been treated as an important goal of artificial intelligence (AI) for a long time. Recently, with the development of deep learning techniques, AlphaGo [1] developed by DeepMind beat a top-level professional player for the first time. Moreover, the advanced version AlphaGo Zero [2] is much stronger than the previous one by learning on itself.

Since the strength of recent Go programs surpassed even top-level human players, it becomes even worthier to research into entertaining or coaching human players [3, 4].

Received August 9, 2020; revised October 25, 2020; accepted November 29, 2020.
Communicated by Chuan-Kang Ting.

There are many key points for entertaining and coaching, and an important one is to control the game position properly, *i.e.*, keeping a good balance between Black and White. Strong computer players can easily beat intermediate players but are undesired. Entertaining computer players should select gentle or weak moves intentionally; meanwhile, such moves should look natural. For entertainment, another possible way is to let computer players have preferences just like humans, *e.g.*, favoring center territories.

Ikeda and Viennot [6] proposed some methods for natural position control and production of various strategies, which were successfully applied to a traditional Monte-Carlo tree search (MCTS) program named Nomitan. However, we find some problems applying these methods to state-of-the-art programs because their strength and mechanisms are different from traditional MCTS programs.

In this paper, we first review related work on computer Go and methods for position control and various strategies in Section 2. Next, we summarize in Section 3 the differences between traditional and new Go programs, including some potential problems. In Section 4, we propose methods for position control with natural moves and production of various strategies for AlphaGo Zero programs such as LeelaZero [11] (abbr. Leela) and ELF OpenGo [7] (abbr. Elf). The results of the position control methods (gentleness and naturalness) are presented and discussed in Section 5. The results of producing center-oriented and edge/corner-oriented strategies (evaluations by human subjects) are shown in Section 6. Section 7 introduces a new method for position control in endgames using KataGo [14]. Finally, Section 8 concludes this paper. Note that this paper is extended from our previous work [10], where we include two new methods for producing various strategies (Section 4.4.1) and for controlling positions in endgames (Section 7).

2. RELATED WORK

2.1 Strong Go Programs

The game of Go is regarded as one of the toughest problems for AI. The progress of the strength of Go programs was slow for a long time until Monte-Carlo tree search (MCTS) made significant progress [5]. MCTS Go programs reached high amateur levels, such as KGS-6d, in 2015.

In 2016, AlphaGo [1], which combined multi-layer convolutional neural networks (CNNs) and MCTS, beat the top-level player Lee Sedol and became the first program to defeat top-level professionals. A successor of AlphaGo was AlphaGo Zero [2], having several crucial differences. Nevertheless, both programs used the selection probabilities of moves (policy) and the expected win rates (value) of game states from CNNs, trained by supervised learning or reinforcement learning.

2.2 Position Control

Putting handicap stones is a classical way to balance players' strength in Go. However, even nine stones are insufficient for intermediate players to win state-of-the-art programs. Also, too many handicap stones may harm the enjoyment. Thus, researchers have tried different approaches to weaken strong programs.

The approaches to control Go programs' strength can be roughly divided into two

types, static and dynamic. The former tries to be statically weak, *e.g.*, decreasing the number of MCTS simulations or thinking time. The latter tries to select proper moves according to the game states dynamically. For example, when a computer player has a big advantage over a weaker human player, some bad moves are intentionally played, which we call *position control*.

The two groups of approaches have their advantages and disadvantages. Static approaches have consistent strength, usually a good aspect. However, the estimation of opponents' levels should be done in advance. Further, even when two players have a similar strength, sometimes one-sided games happen. On the contrary, dynamic approaches can do position control without knowing opponents' levels. Good moves or bad moves are selected according to how advantageous computer players are. However, inconsistent strength may be problematic sometimes.

Both static and dynamic approaches select bad moves sometimes. However, too-bad moves should be avoided since they harm the enjoyment of the game. Softmax selection is such an approach for MCTS programs [9]. Each candidate move m_i is selected by a probability of $n_i^z / \sum_j n_j^z$, where n_i is move m_i's number of visits, and z is a parameter. For $z = 0$, a move is randomly selected from all candidates; for $z \to \infty$, the most-visited move is selected. Wu *et al*. [8] applied this approach to the game of Go on Elf and showed that the strength could be well controlled by z. They also proposed a new technique to remove less-visited moves from candidates to avoid selecting too-bad moves.

Ikeda and Viennot [6] explicitly considered the naturalness of moves for entertaining intermediate players. They employed a traditional MCTS program and estimated the naturalness of moves by selection probabilities from a static evaluation trained based on human game records. When the computer player had high expected win rates, it intentionally selected worse but the most natural moves.

2.3 Playing Various Strategies

Human Go players have widely varied strategies or preferences, such as favoring edge/corner territories, favoring center territories, pessimistic, optimistic, offensive, and defensive. Such variety is a seed of entertainment, and it is valuable to equip computer Go programs with different strategies so that they look like to have characteristics. Ikeda and Viennot [6] employed a simple trick and partly succeeded in letting human subjects identify each strategy, with only slightly losing the strength. More specifically, they changed the definitions of wins and losses in MCTS simulations. For example, players favoring center territories counted a center territory as 1.2 points and a corner/edge territory as 0.8 points, while both should be 1 point under the standard rules.

3. DIFFERENCES IN NEW GO PROGRAMS

The existing methods for position control and producing various strategies were proposed for traditional MCTS programs [6]. Recent programs based on AlphaGo Zero are much stronger than traditional ones and have different mechanisms. Thus, the methods may be unable to apply directly or may cause poor performance. This section summarizes the differences and the related problems, either well-known or of our expectation.

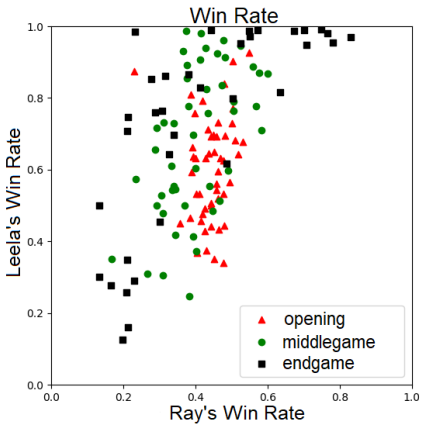

Fig. 1. Predicted win rates by Ray (x-axis) and Leela (y-axis).

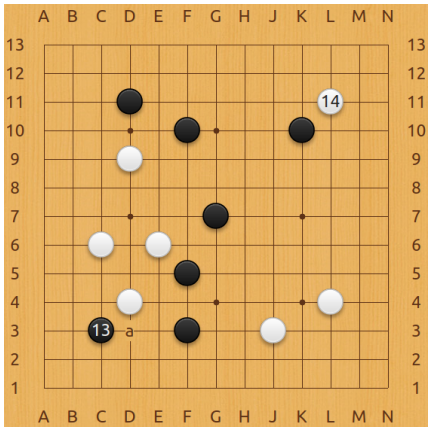

Fig. 2. A move far from the opponent's.

Two open-sourced programs are used for the discussions: Leela [11] as the recent one (AlphaGo Zero) and Ray [13] as the traditional one (MCTS).

The main difference lies in the state evaluation mechanism. To evaluate leaf nodes, traditional MCTS programs run some (biased) random simulations to the ends of games, deciding win or loss by Go rules. In contrast, recent programs employ value networks without random simulations. The previous method for producing various strategies (described in Section 2.3) cannot be used since it relies on modifying the definitions of wins and losses at the ends of games.

Also, it is well known that win rates predicted by recent programs are somewhat sharp or drastic. Fig. 1 shows win rates of moves predicted by Ray and Leela in 50 testing games, where red triangles are win rates at the 30th move (opening), green circles at the 60th (middlegame), and black squares at the 100th (endgame). The heights are greater than the widths for both the red-triangle and green-circle distributions, meaning small dis/advantages for Ray may be judged big by Leela. The disagreements mainly come from that Ray employs random simulations while Leela not. Long random simulations, especially in openings, may introduce noise that averages out true dis/advantages from win rates due to good/bad luck. Since the previous methods are based on win rates and their differences among candidate moves, some tunings are needed for new programs.

Another difference is whether human game records are used, which recent programs not. The moves with high selection probabilities from policy networks are sometimes different from human players' senses. Notably, recent programs tend to play far from the opponent's moves more frequently. For example, when Leela played against Ray, the average Euclidean distance between Leela's moves and Ray's last moves (100 games, the 2nd-60th moves) was 3.16 ± 0.10. This is significantly greater than that between Ray's moves and Leela's last moves, 2.65 ± 0.08. Usually, it seems natural to react directly when the opponent tries to attack or invade. For beginners, such a tendency is especially strong. Thus, playing far from the last move is sometimes risky for entertainment, where beginners may think, "my move was ignored, and it is a strange play." Fig. 2 is an example where Black played at C3 (13), aiming to invade the territories of the White corner. For beginners,

it is natural to play at D3 (a), while Leela selected L11 (14). In fact, this move is not bad but may look strange from beginners' views.

4. APPROACHES

4.1 Position Control

For position control, we generally follow the procedure proposed by Ikeda and Viennot [6]. Some methods are newly introduced to deal with the problems described in Section 3. For simplicity of discussion, let moves m_i be sorted by their win rates, $m_1(w_1, p_1)$, $m_2(w_2, p_2)$, ..., where w_i is m_i's win rate, and p_i the selection probability[1]. The procedure to select moves is summarized as follows; (1) When w_1 is much higher than w_2, play the best move m_1; (2) When w_1 is low, *i.e.*, the program is losing, play the best move m_1; (3) When w_1 is moderate (around 50%), select the most natural move among moves whose win rates are not low; (4) When w_1 is high, *i.e.*, the program is winning, intentionally select a worse move such as m_2 or m_3 considering the balance between selection probabilities and win rates.

When applying the method directly to a recent program, Leela [11], three problems occur: (a) some candidate moves have only a few visits, so their win rates are not reliable, (b) Leela tends to play far from the opponent's last moves, and (c) naturalness is often sacrificed for decreasing win rates.

To solve problem (a), we need to search bad moves more (*i.e.*, to have more visits) to get more reliable win rates since they may be intentionally selected for position control, though it is reasonable for strong play to ignore them after a few visits. Therefore, we employ a high exploration coefficient of 10 in the PUCT algorithm [1, 2]. Also, we exclude moves with visits less than 100 from candidates. We call these two modifications Method A. Problems (b) and (c) are discussed in the following subsections.

4.2 Method-B: Decreasing Distance from Opponent's Last Moves

In many traditional MCTS programs, the distances to the opponent's last moves were explicitly considered in the selection probabilities [5]. Namely, candidate moves nearer the opponent's last move had higher selection probabilities.

Thus, we propose to modify the policy networks' outputs according to the distances from the opponent's last moves. Let m_i be a candidate move, p_i be the original selection probability, and d_i be the Euclidean distance from the opponent's last move to m_i. The new selection probability p'_i is calculated by $p'_i = p_i \times \omega_i$, where ω_i is the weight assigned according to d_i. More specifically, we divide d_i into seven segments and set ω_i to 1.50 for $d_i \leq 2$, 1.25 for $2 < d_i \leq 3$, 1.00 for $3 < d_i \leq 4$, 0.75 for $4 < d_i \leq 5$, 0.50 for $5 < d_i \leq 6$, 0.25 for $6 < d_i \leq 7$, and 0.10 for $d_i > 7$.

One exception is that the program has no stones around the opponent's last move (Euclidean distance $\leq$3), which means that the opponent's move does not intend to attack or invade. In such a case, selection probabilities are unchanged.

In fact, the idea of modifying the selection probabilities by assigning weights is applicable to various circumstances to achieve specific effects. Playing moves near the op-

[1] We assume p_i can somewhat approximate naturalness, even learned without human games.

ponent's last moves is an example. Another example will be shown in Section 4.4.1 to produce center- or edge/corner-oriented strategies. Designing the criteria for assigning weights and finding proper weights to achieve the desired effects can be considered a kind of optimization problem. In this paper, we determine the criteria and weights by some preliminary experiments and basic Go knowledge. It is also promising to employ other optimization or machine learning methods to select the criteria or tune the weights, which is left as future research.

4.3 Method C: Avoiding Big Sacrifice on Naturalness

In the previous method [6], when the best win rate w_1 was higher than a threshold of 0.55, a bad move was intentionally selected. At first, each candidate m_i was tested whether it satisfied one of the following four conditions sequentially: (i) $w_1 - w_i < 0.03c$ and $p_i > 0.05$, (ii) $w_1 - w_i < 0.04c$ and $p_i > 0.10$, (iii) $w_1 - w_i < 0.06c$ and $p_i > 0.20$, and (iv) $w_1 - w_i < 0.08c$ and $p_i > 0.40$, where c is a parameter to control the degree of gentleness. Higher c enabled worse moves to be selected and usually sacrificed the naturalness. Let M^+ be a set including m_1 and moves satisfying one of the conditions, *i.e.*, candidates not too bad and relatively natural. From M^+, the move that had the lowest w_i was played to decrease the advantage.

Fig. 2 also serves as a typical example of this problem. After Leela's search, the win rate of D3 was 69.7%, and the selection probability was 0.400. Meanwhile, the win rate of L11 was 69.0%, with a selection probability of 0.139. The previous method selected L11 to decrease the win rate, sacrificing the naturalness. This selection seemed unreasonable since the win rate did not decrease too much, but naturalness was almost sacrificed.

Thus, we propose to consider not only w_i but also p_i when selecting the move from candidates M^+. More specifically, the move that has the lowest $w'_i = w_i - \alpha \times p_i$ is selected, where α is a positive value and is tunable. For the above example, w' of D3 is 65.7% for $\alpha = 0.1$, and that of L11 is 67.6%, so D3 is selected.

4.4 Playing Various Strategies

Ikeda and Viennot [6] also proposed a method for producing various strategies. They changed the definitions of wins and losses in random simulations of MCTS, which needs to play games to the ends. The same method cannot be applied to recent programs such as Leela and Elf since they do not do random simulations until games end. In this paper, we propose two different methods for strategy production. The *online* method modifies the selection probabilities from the policy networks. The *offline* method trains special networks for specific strategies.

4.4.1 Online: modifying selection probabilities

In the online method, we modify policy networks' outputs based on the moves' distances to the borders to produce center- and edge/corner-oriented strategies. For simplicity of discussions, let the first line be the border lines, the second line be one grid closer to the center, and so on. For move m_i on the l_i-th line, we assign a new selection probability p''_i by $p''_i = p'_i \times \omega'_i$, where p'_i is the modified selection probability by Method B in Section 4.2 and ω'_i designed based on the board size and the desired strategy. p'_i is incorporated to reduce the unnaturalness of moves.

For a center-oriented strategy on 19×19 boards, we assign ω_i' to 0.25 for $l_i \leq 2$, 0.50 for $l_i = 3$, 1.50 for $l_i = 4$, 1.75 for $l_i = 5$, and 2.00 for $l_i > 5$. Since 13×13 boards are smaller, we assign ω_i' to 0.50 for $l_i \leq 3$ and 2.00 for $l_i > 3$. Similarly, an edge/corner-oriented strategy can be produced by assigning to ω_i' 2.00, 1.50, 0.75, 0.50, and 0.25 on 19×19 boards and 2.00 and 0.50 on 13×13 boards, with the same criteria on l_i.

4.4.2 Offline: training new networks

In the offline method, we propose to (1) train new networks that can play specific strategies and (2) combine the results of such networks with normal ones to maintain the strength. More specifically, we redefine wins and losses of the self-play games for AlphaGo Zero programs and train new policy and value networks from scratch. The idea is similar to the previous method [6]. Since training AlphaGo Zero networks is computationally costly, in this paper, we target on 13×13 boards. These can be achieved by making minor modifications to open-sourced programs such as Leela [11]. Namely, we can customize the training with our win/loss definitions and board sizes.

To train center-oriented networks, we give higher weights to center territories and determine wins and losses by comparing Black and White's weighted territories. Komi is also considered as the standard Go rules. We assign weights according to the distances to the borders. Each point on the 4th line is counted as 1, on 1st-3rd lines (*i.e.*, corner or edge) as $1-\beta$, and within the 5th line as $1+\beta$, where β is a positive value determining how center territories are preferred compared to edge/corner ones. By assigning a negative value to β, we can train edge/corner-oriented networks.

The idea of modifying the definitions of wins and losses is applicable to train networks with other preferences, such as optimistic, pessimistic, offensive, and defensive. For example, counting each captured stone as –2 points may train defensive networks. We remain other preferences as future research.

If such networks are used solely, they are highly likely not to play the standard games of Go well. Thus, we further propose the *mixed approach* to combine the results of biased networks with normal ones for balancing the strength and strategy preference.

The approach contains two parts, search and mix. For the former, searches using the normal network and the biased network are done separately to obtain two lists of candidate moves, M_N and M_B. For the mix part, first, some moves m are removed from M_B: (i) if $w_N(m) < (\max_i\{w_N(i)\} - w_{diff})$, where $w_N(m)$ is m's win rate from the search with the normal network and w_{diff} a tunable parameter determining the acceptable difference of win rates from the most promising move, or (ii) if $p_B(m) < p_{th}$, where $p_B(m)$ is m's selection probability from the biased network and p_{th} a tunable threshold for selection probabilities. If M_B still has candidates, the most visited one is selected; otherwise, the most visited move in M_N is selected. In other words, the approach selects the best move to show the preference (*e.g.*, center- or edge/corner-oriented) only when the move is also acceptable from the view of normal play.

5. EXPERIMENTS: POSITION CONTROL METHODS

In this section, we compare the previous and the proposed position control methods when implemented in a recent program, Leela. We conducted experiments on 13×13

boards. Leela's policy network and value network were trained from scratch for two weeks on a server with two Titan-X GPUs. The traditional MCTS program Ray served as a weaker player. The numbers of simulations per move for Leela and Ray were 16,000 and 60,000, respectively. Leela always played as White.

5.1 Evaluation of Position Control Ability

We first evaluated the strength of our trained Leela. Normal Leela without position control was much stronger than Ray, who won none of the 30 testing games against the normal Leela. Next, we evaluated Leela with the previous position control method [6], setting all parameters to the recommended values. Especially, the important parameter c was set to 1.5. The only difference was to apply Method A (shown in Section 4.1). We denote this setting by $\text{Leela}_{\text{A15}}$, where Ray won 183 out of 500 games. The results showed that position control was performed, but the degree was insufficient.

To weaken Leela, c should be increased, but we expected the naturalness to be sacrificed. Thus, we tried 2.5 for c and included methods-B and -C introduced in Sections 4.2 and 4.3 for compensating the sacrificed naturalness. We set α to 0.25 for Method C and denote this setting by $\text{Leela}_{\text{ABC25}}$, where Ray won 238 out of 500 games. $\text{Leela}_{\text{ABC25}}$ was significantly weaker than $\text{Leela}_{\text{A15}}$. When c was 3.5, Ray's win rate went up to 73%.

5.2 Evaluation of Naturalness

By ignoring moves' naturalness, it is generally easy for stronger human/computer players to lose intentionally. They can play meaningless or even suicidal moves to lose their advantages. However, weaker players do prefer position control in more natural ways. To evaluate the naturalness, we asked human players to count the unnatural moves in the provided game records. Ikeda and Viennot [6] did a similar experiment, where Black was the weaker player, and White was position control methods. They reported that 5.2 White's moves per game seemed unnatural for a native method and only 1.9 for their proposed method. However, since the human subjects and their strength, as well as the programs, differed from ours, it is hard to compare their results in the two papers.

In our experiments, we prepared three versions of position control Leela: $\text{Leela}_{\text{A15}}$, $\text{Leela}_{\text{ABC25}}$, and $\text{Leela}_{\text{naive}}$, where $\text{Leela}_{\text{naive}}$ collected candidate moves by Method A and selected the one with a win rate nearest 0.5 without explicitly handling moves' naturalness. We asked nine human subjects, ranked from 8k to 8d, to review games played by the three versions. Totally, 15 games were collected from the three versions, where each played 5 games against Ray. All the 15 games were given in random order and a blind manner. Each person was given two hours to review the 1st-60th moves in the records.

As for employing the program Ray as the opponent (a weaker player), the results might be different if intermediate human players were employed. Ray sometimes plays unnatural and too-aggressive moves, while it is generally easier to play natural and gentle moves against natural and defensive moves. However, since human players have varied characteristics, we consider that unnatural and too-aggressive moves may also be played. The difference between employing weaker programs and intermediate human players as opponents of the position control methods remains an open question.

Table 1 shows the average numbers of unnatural moves per game for the three prepared Leela versions. The differences between $\text{Leela}_{\text{naive}}$ and our two methods were sta-

Table 1. Four Leela versions' position control results with 95% confidence intervals.

Leela version	Ray's win rate	Number of unnatural moves/game
Normal Leela	0.00	–
$\text{Leela}_{\text{naive}}$	–	2.29 ± 0.53
$\text{Leela}_{\text{A15}}$	0.37 ± 0.04	1.27 ± 0.41
$\text{Leela}_{\text{ABC25}}$	0.48 ± 0.04	1.22 ± 0.25

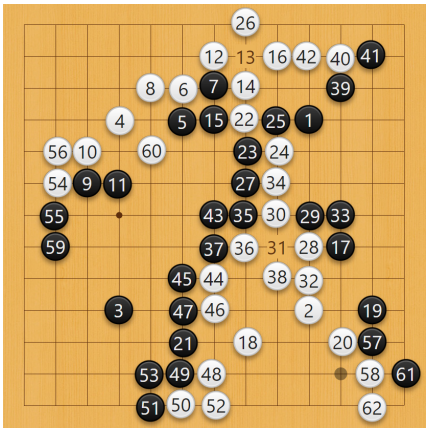

Fig. 3. Ray (Black) vs. $\text{Leela}_{\text{ABC25}}$ (White).

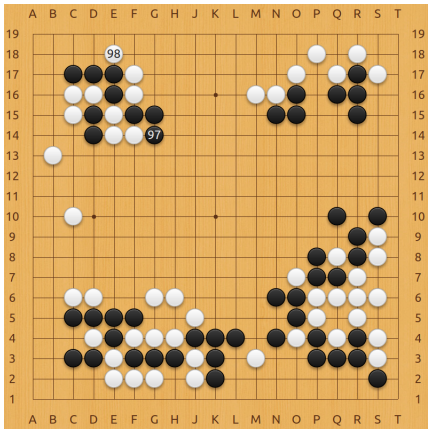

Fig. 4. An example of a ladder (White 98).

tistically significant. The results demonstrated the effectiveness of the position control methods for AlphaGo Zero programs. Furthermore, $\text{Leela}_{\text{ABC25}}$ had almost the same naturalness as $\text{Leela}_{\text{A15}}$, while the strength was significantly weakened. This suggested that Methods B and C were effective.

Besides, the average Euclidean distance with the 95% confidence interval was 2.33 ± 0.06 from $\text{Leela}_{\text{ABC25}}$'s moves to Ray's last moves (100 games, the 2nd-60th moves). The value was significantly lower than those from Ray to its two opponents (2.65 ± 0.08) and from $\text{Leela}_{\text{A15}}$ to Ray (3.16 ± 0.10), as we expected. The degree can be controlled by tuning parameters of Method B according to players' preferences. Fig. 3 shows a game between Ray (Black) and $\text{Leela}_{\text{ABC25}}$ (White). White played gently without obviously bad moves, and finally, Black won 0.5 points.

5.3 Remaining Problems

We have shown that the number of unnatural moves from human players' views was low. Still, there are some remaining problems.

5.3.1 Ladder

A ladder [15] is a well-known move sequence in Go where an attacker tries to capture a group of the opponent's stones in a zigzag pattern. If there are no intervening stones, the group will hit the border and be captured. Fig. 4 is a game between a human player (KGS-3d, Black) and Elf (White). Black played at G14 (97) and threatened three white stones by a ladder. However, Elf ignored the risk and played at E18 (98). The white stones were

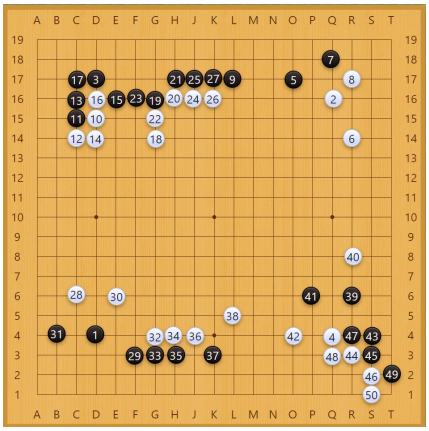

Fig. 5. $\text{Leela}_{\text{online\_edge}}$ (Black) vs. $\text{Leela}_{\text{online\_center}}$ (White) on the 19×19 board.

then captured, and Elf resigned soon. Our position control methods cannot avoid such unnatural moves since they are issues of the original programs.

5.3.2 Openings

As mentioned in Section 3, recent programs may predict very high (*e.g.*, over 80%) or low win rates even in opening games. For example, Leela often judges the advantage to be big and tries to lose by strange moves. Human players are easily aware of strange moves, especially in openings. To alleviate this problem, it may be useful to lower the level of position control in openings.

5.3.3 Endgames

The problem in endgames is a bit more complicated, explained as follows. When the territories' difference is small, human coaches often try to lose by 0.5 or 1.5 points. It is easy for human players but not for Leela since +0.5 points and –0.5 points may almost mean win rates of 100% and 0%. When there is a move with +0.5 points, Leela is almost impossible to select another with –0.5 since the win rate difference is too big. We will propose a new method to solve this problem in Section 7.

6. EXPERIMENTS: PLAYING VARIOUS STRATEGIES

This section presents the results of the two methods for producing various strategies introduced in Section 4.4. The online method modifies the selection probabilities from normal networks, while the offline method requires additional training for biased networks. Sections 6.1 and 6.2 show some games played by the methods, and the playing strength of the offline method is further analyzed. Section 6.3 then describes the experiments on human subjects' evaluations of the methods.

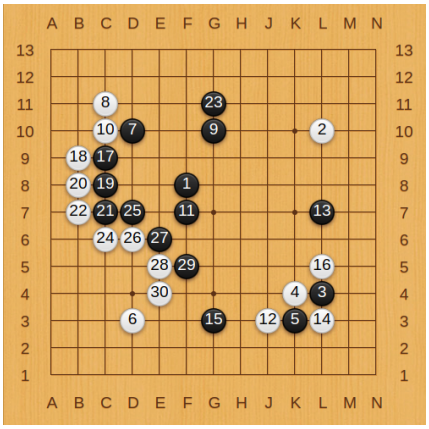

Fig. 6. $\text{Net}_{\text{center}}$ (Black) vs. normal Leela.

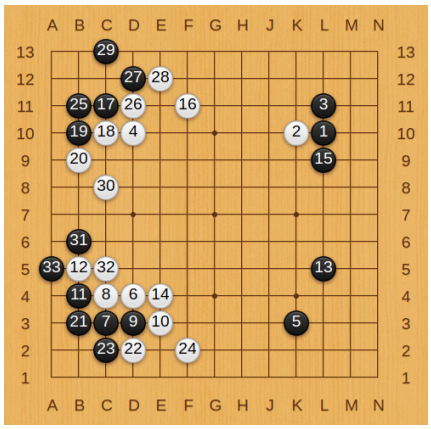

Fig. 7. Net_{edge} (Black) vs. normal Leela.

6.1 Online Method

We applied the online method to 19×19 boards using Leela [11] with the network weights trained from human games [12] and 6,000 simulations per move. Fig. 5 shows a game between the edge/corner- (Black) and center-oriented (White) players. It can be seen clearly that the methods produced corresponding strategies well. For example, White 10 to 14 are typical moves when White prefers center territories and Black prefers edge/corner. Also, White 18, 30, 32, and 38 are all typical moves for center-oriented players.

6.2 Offline Method

We trained a center-oriented Leela network ($\text{Net}_{\text{center}}$) and an edge/corner-oriented one (Net_{edge}) for about two weeks from scratch on 13×13 boards. The parameter β was set to +0.2 for $\text{Net}_{\text{center}}$ and –0.2 for Net_{edge}.

6.2.1 Pure Strategies

As discussed in Section 4.4.2, $\text{Net}_{\text{center}}$ and Net_{edge} were trained not for playing solely. Both won none of the 60 games against the normal Leela for checking their behaviors. Fig. 6 shows a game between $\text{Net}_{\text{center}}$ (Black) and the normal Leela (White). Moves 1, 9, and 11 look very strange for the standard Go rules but do fit $\text{Net}_{\text{center}}$'s goal. Fig. 7 shows a game between Net_{edge} (Black) and the normal Leela (White). Moves 13 and 15 are far from the hot area, and finally, the left-bottom black stones are in danger. The results confirmed that $\text{Net}_{\text{center}}$ and Net_{edge} had clear preferences for the center and the edge/corner territories, respectively, though they were too weak to play the standard Go.

6.2.2 Mixed approach

Next, we applied the mixed approach in Section 4.4.2 with $\text{Net}_{\text{center}}$ and Net_{edge}, referred to as $\text{Leela}_{\text{offline\_center}}$ and $\text{Leela}_{\text{offline\_edge}}$, where w_{diff} were set to 0.05 and p_{th} to 0.1. All programs used 6,000 simulations per move. Figs. 8 and 9 show games by $\text{Leela}_{\text{offline\_center}}$ and $\text{Leela}_{\text{offline\_edge}}$ (Black) against the normal Leela (White), respectively. While there are still several unexpected moves, the preferences can be identified.

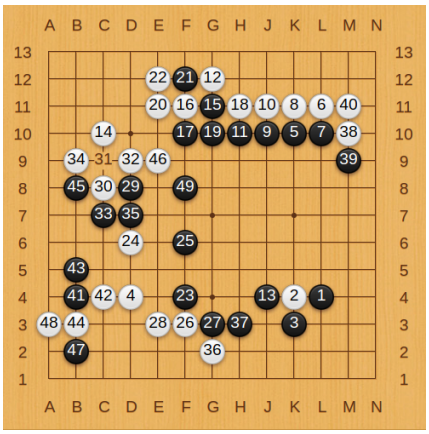

Fig. 8. $\text{Leela}_{\text{offline\_center}}$ (Black) vs. normal Leela.

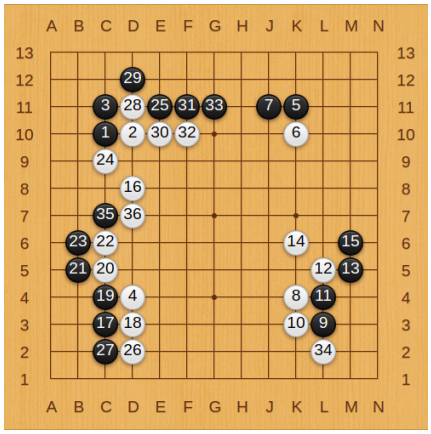

Fig. 9. $\text{Leela}_{\text{offline\_edge}}$ (Black) vs. normal Leela.

We also conducted experiments to evaluate the strength of the mixed versions. Each of $\text{Leela}_{\text{offline\_center}}$ and $\text{Leela}_{\text{offline\_edge}}$ played 300 games against the normal Leela. The numbers of wins were 135 and 129, respectively. The mixed versions were not weak because they only selected acceptable moves from the normal Leela's view.

6.3 How Well Strategies Can Be Identified

We conducted two experiments to confirm how well human players could identify the strategies. The first one was for the offline method on 13×13 boards. The next experiment compared the two methods on 13×13 boards and applied the online method to 19×19. All programs used 6,000 simulations per move to play games.

6.3.1 Strategy identification of the offline method

We first experimented on the offline method on 13×13 boards employing human subjects to evaluate how the strategies can be identified from their views. We tested three versions of Leela, (A) the normal Leela, (B) $\text{Leela}_{\text{offline\_center}}$, and (C) $\text{Leela}_{\text{offline\_edge}}$. Four groups of game records were generated, A vs. A, A vs. B, A vs. C, and B vs. C. In each group, five games were played. Totally, 20 game records, containing only the 1st-40th moves, were given to eight human subjects (ranked 6k to 8d) in random order and a blind manner. Each person was given one hour to review the game records.

Human subjects were asked to judge which strategy can be identified for Black and White, respectively, in each game by –2 for center-oriented, –1 for slightly center-oriented, 0 for none, +1 for slightly edge/corner-oriented, and +2 for edge/corner-oriented.

Table 2. Evaluations of $\text{Leela}_{\text{offline\_center}}$, $\text{Leela}_{\text{offline\_edge}}$, and normal Leela.

Evaluation score	–2	–1	0	+1	+2	Total game	Average
$\text{Leela}_{\text{offline\_center}}$	36	19	20	3	2	80	–1.05
$\text{Leela}_{\text{offline\_edge}}$	3	4	13	14	46	80	+1.20
Normal Leela	37	22	48	25	28	160	–0.09

Table 2 lists the numbers of each score and the average evaluations. Both center- and edge/corner-oriented strategies could be produced and identified with a high portion. In 115/160 answers, the programs' strategies were correctly identified, and only 12 answers were opposite. The average evaluations also showed clear differences. Interestingly, when the normal Leela played against the center- (or edge/corner-) oriented Leela, the normal Leela tended to be judged as edge/corner- (or center-) oriented.

6.3.2 Comparison of the Online and the Offline Methods

In this experiment, the online method was applied to both 13×13 and 19×19 boards. The offline method was applied to 13×13 boards only. We prepared a total of 45 games, 15 for each group of (a) the offline method on 13×13 boards, (b) online on 13×13, and (c) online on 19×19. In each of the 15 games, 5 were played by center-oriented vs. edge/corner-oriented, another 5 by center-oriented vs. the normal Leela, and the other 5 by edge/corner-oriented vs. the normal Leela. With different combinations of methods, board sizes, and strategy preferences, we had eight different programs.

A total of ten human subjects (ranked 1k to 6d) participated in the experiment. Each human subject was asked to review 9 games, 3 for each of the three groups, and give –2 to 2 for center-oriented to edge/corner-oriented, as described in Section 6.3.1. Game records of 13×13 boards contained the 1st-40th moves and 19×19 the 1st-60th. Each game was evaluated by two human subjects, and thus, we collected 20 evaluations for each program, except that the normal Leela on 13×13 boards received 40.

Table 3. Evaluations of different programs of board sizes, methods, and strategies.

	Center-oriented	*Normal*	Edge/corner-oriented
13×13 offline	–0.40	–0.10	+1.05
13×13 online	–0.50		+0.70
19×19 online	-1.25	+0.65	+0.85

Table 3 shows the average evaluation scores of each program. For both the online and the offline methods and both 13×13 and 19×19 boards, the edge/corner-oriented strategies had the highest scores, and the center-oriented ones the lowest. The results supported that the two strategies could be well-identified. However, on 19×19 boards, +0.65 for the normal Leela was close to +0.85 for edge/corner-oriented. We suspected that the normal Leela was already slightly edge/corner-oriented. Such a preference was confirmed by several strong Go players that we asked to review some games played by the normal Leela. We considered that the weights should be re-designed to make the edge/corner-oriented Leela's preference clearer, *e.g.*, decreasing the 0.75 for the fourth line.

For the online method, strategies on 19×19 boards were easier to be identified than 13×13. We suspected that different strategies are easier to be identified in opening games, and the lengths of openings are longer for 19×19 boards. Besides, for 13×13 boards, the offline method was slightly better than the online method. To sum up, the online method effectively produced desired strategies and was easier to employ (no extra training was required). The offline method could show clearer preferences, and we expected that it would also produce promising results for 19×19 boards.

We further generated 40 games for each strategy and counted the number of moves

Table 4. Average numbers of moves played at the center area by different programs of board sizes, methods, and strategies with 95% confidence intervals.

	Center-oriented	*Normal*	Edge/corner-oriented
13×13 offline	9.13±0.82	8.73±0.71	7.63±0.89
13×13 online	10.80±0.72		3.88±0.75
19×19 online	18.68±1.01	13.43±1.24	11.33±1.09

played at the center area (4th line or more inner) in openings (1st-40th moves for 13×13 and 1st-60th moves for 19×19). More specifically, for each method, each strategy played 20 games against the other two strategies. Increasing the game number was to reduce the effect of randomness compared to the above experiment, which had 10 games for each strategy. The average center-move numbers are listed in Table 4. The overall tendency was similar to that of Table 3: center-oriented players played more moves at the center area while edge/corner-oriented players fewer. The differences between the three strategies were statistically significant ($p < 0.015$) except for the 13×13 offline case.

7. POSITION CONTROL IN ENDGAMES

For position control, we have introduced new methods in Sections 4.1 to 4.3, confirmed their effectiveness through experiments in Sections 5.1 and 5.2, and discussed some remaining problems in Section 5.3. This section proposes an approach to solve the third problem, the difficulty of position control in endgames (or Yose in Japanese).

Both the previous and the proposed position control methods (*i.e.*, [6] and Section 4) use expected *win rates* to evaluate dis/advantages of given states or moves. In the openings or middlegames, we can assume that if a move's win rate is worse than the best one by a certain degree, say 30%, the move is terrible and should not be played even for position control. However, in endgames, this assumption does not hold frequently. As explained in Section 5.3.3, +0.5 points (of territory score) can mean almost a win rate of 100% for strong computer players and –0.5 almost 0%. Thus, a move losing 1 point may be judged as very bad and prohibited from being played by the existing methods.

7.1 Core Concept

We propose to control position in endgames by using expected *territory advantages* instead of win rates. Human players often consider territory advantages, especially in endgames, *e.g.*, "Black will win about 5 points," or "this move gained 2 points, but there was another that could gain 4." If computer players can also calculate such numbers, it is possible to develop more effective position control methods and coaching methods.

Since Leela and Elf do not have such a function, at least in the versions we employed, we switch to another open-sourced program, KataGo [14]. In addition to common functions such as obtaining the prior selection probabilities from the policy network and doing searches to get each move's win rate and the number of visits, KataGo outputs many other values such as (a) each move's expected territory advantage, called *scoreLead* in the program, (b) the standard deviation of territory advantage, *scoreStdev*, and (c) the prediction of each point to be finally occupied by Black, White, or shared, *ownership*.

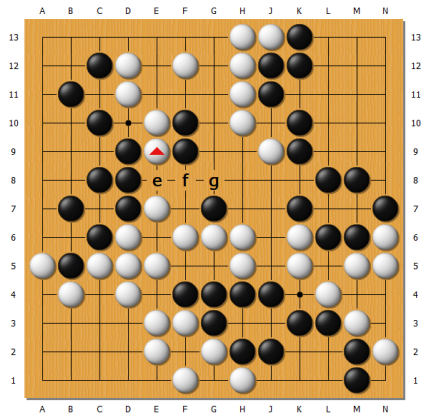

Fig. 10. A typical example of position control in an endgame, where $\text{Leela}_{\text{ABC25}}$ plays at **f** or **g** while the new approach at **e**.

In this approach, we mainly utilized expected territory advantages. More specifically, we try to select moves that do not lose too many territory advantages, making the territory advantages in an adequate range. In contrast, previous methods consider win rates. Meanwhile, all methods aim to play moves as naturally as possible.

7.2 Procedure

Algorithm 1 shows the concrete procedure, applied only when the game is ending (discussed more in Section 7.4). For each legal move m_i in the move list M, we utilize its selection probability p_i (called *prior* in KataGo) and its expected territory advantage δ_i (*scoreLead*). Note that we do not consider expected win rates and numbers of visits in this approach. All values in Algorithm 1, including γ, can be parameterized and tuned according to the employed programs and the target players.

Algorithm 1 : Position control in endgames using KataGo.

1: Sort legal moves by p_i in decreasing order and collect the first 20 moves as M.
2: If $p_1 > 0.9$, play m_1 directly.
3: Remove $\{m_i | p_i < 0.01\}$ from M. // *never play very unnatural moves*
4: Calculate δ_i for each $m_i \in M$ and let δ^* denote the maximum.
5: Remove $\{m_i | \delta_i < \delta^* - 5\}$ from M. // *never play moves that lose too much territory*
6: Calculate the preference score s_i and play the move with the highest s_i:
if $\delta_i < -10$, $s_i = p_i / \gamma^{-10-\delta_i}$ // *too disadv., discount by the degree of disadv.*
if $-10 \leq \delta_i \leq -4$, $s_i = p_i$ // *an adequate range, follow the selection probability*
if $-4 < \delta_i$, $s_i = p_i / \gamma^{4+\delta_i}$ // *less disadv. or even adv., discount by the degree of adv.*

7.3 Example

Fig. 10 shows a typical example that adequate position control is needed, where White just played at E9 (marked by a triangle). Black likely has a small advantage as Leela judged

Black's win rate to be 58.2% with the best move of G8 (marked by g). Besides, KataGo judged Black's win rate to be 83.6% with the best move of F8 (marked by f).

Table 5. Candidate moves of the original method by Leela$_{ABC25}$, where moves with visits less than 100 are excluded.

Move	Win rate	Visits	Modified selection probability
G8 (g)	0.582	871	0.0095
F8 (f)	0.572	1188	0.0445
L5	0.384	133	0.0043
E8 (e)	0.352	3429	1.1176

Table 5 shows the search results by Leela$_{ABC25}$ in Section 5. Black's win rate was 58.2% with the best move G8, while E8 was much more natural (a higher selection probability). However, E8 would not be selected since its win rate was much lower than G8.

Table 6. Candidate moves of the new approach by KataGo, where only moves with selection probabilities higher than 0.1 are shown.

Move	(Win rate)	Territory advantage	Selection probability	Preference score
E8 (e)	0.235	–1.234	0.4755	0.0699
G8 (g)	0.516	+0.527	0.1410	0.0061
J8	0.538	+0.605	0.1307	0.0053
F8 (f)	0.836	+1.862	0.1094	0.0018

Table 6 shows the moves' statistics collected from KataGo with $\gamma = 2$, including the win rates after a search, though the search is unnecessary in this approach. KataGo judged the best move to be F8, where Black's win rate was 83.6%, and the territory advantage was +1.862. E8 was more natural, with an expected territory advantage of –1.234, closer to the adequate range than F8 or G8. Thus, E8 had the highest preference score s_i and was played, regardless of its significant loss of win rate.

7.4 Experiments

We further did an experiment to evaluate the new approach. First, we prepared several board states that just entered endgames and that both players had roughly even chances to win. Next, Leela$_{ABC25}$ (abbr. Leela+) continued to play from the prepared board states against Ray. The new approach using KataGo (abbr. KataGo+) also played against Ray in the same way. We then analyzed and discussed the position control ability and the naturalness based on these games.

In fact, it is an interesting and challenging topic to find the start points of endgames. In this experiment, we tried to collect board states from game records satisfying all the following conditions: (i) the best territory advantage's absolute value lower than 5 for 13×13 boards and 12 for 19×19, which we consider the state roughly even; (ii) the estimated standard deviation of territory advantage (*scoreStdev*) lower than 10, which we consider the state relatively peaceful or static; and (iii) the territory advantage loss of the pass move (*i.e.*, the best move's *scoreLead* minus the pass move's) lower than 7 for 13×13 boards and 5 for 19×19, which we consider to have no big-gain moves anymore.

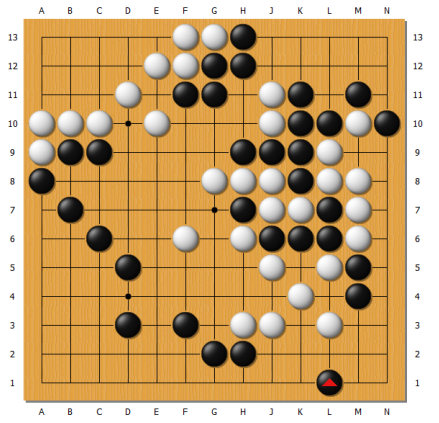

Fig. 11. An example of a selected 13×13 board state just entering endgames.

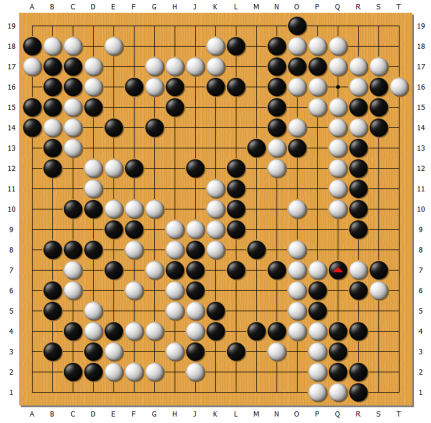

Fig. 12. An example of a selected 19×19 board state just entering endgames.

Fig. 11 is an example of such states on the 13×13 board played between two professionals. Fig. 12 shows a game played between two 3-dan players on the 19×19 board. Both states are peaceful and almost just entering endgames.

Table 7. Position control results of Leela+ and KataGo+, where p_{Leela} and p_{KataGo} are the played move's selection probability from Leela's and KataGo's networks, respectively.

Program	Leela+	KataGo+
vs Ray	63 wins (63%)	37 wins (37%)
p_{Leela} arithmetic mean	0.3551	0.3553
p_{KataGo} arithmetic mean	0.3742	0.4565
p_{Leela} geometric mean	0.2398	0.2473
p_{KataGo} geometric mean	0.1538	0.3008
$p_{Leela} < 0.05$ move	223 (9.70%)	164 (9.34%)
$p_{KataGo} < 0.05$ move	552 (24.15%)	154 (8.50%)

We collected ten 13×13 states and let each of Leela+ and KataGo+ (with $\gamma = 5$) play ten times from each state against Ray. Namely, each program played 100 games. The results are summarized in Table 7. Leela+ won 63% of the games against Ray, while KataGo+ with the new approach only won 37%. For coaching games, the results of KataGo+ should be more favorable. Also, note that the base program KataGo was actually much stronger than the Leela trained by us from scratch for 13×13 boards. In a brief experiment, KataGo won 20 games in a row without handicap and 14 out of 20 games even when two handicap stones and 7.5 Komi were given to our Leela. The results confirmed that the position control ability of the new approach was effective.

Next, Leela+ and KataGo+'s move naturalness was compared. The move naturalness was evaluated by the selection probabilities from policy networks, where both Leela's and KataGo's policy networks were considered. The arithmetic and geometric means of the selection probabilities were calculated. Table 7 shows that KataGo+'s moves were more natural than Leela+'s moves on average, not only from KataGo's view but also from

Leela's view. Besides, we counted the number of moves whose selection probabilities were lower than 0.05. Such unnatural moves should be avoided but sometimes needed for position control. The number (and the proportion) of such moves of KataGo+ was lower than Leela+, also from both KataGo and Leela's views.

We also collected three 19×19 states, and KataGo+ played against Ray ten games from each state. Leela+ was not compared since it required further parameter tunings for 19×19 boards. Instead, we compared $\gamma = 3$ and $\gamma = 5$ for KataGo+[2]. The γ parameter decides how greatly the selection probability is sacrificed to control position. For $\gamma = 3$, KataGo+ won 29 out of 30 games, *i.e.*, failing to lose. In contrast, KataGo+ won only 17 games with $\gamma = 5$. As for naturalness, the numbers of unnatural moves ($p_{KataGo} < 0.05$) were 66 for $\gamma = 3$ and 117 for $\gamma = 5$. The results showed a tradeoff between position control ability and naturalness. Also, the γ parameter was sensitive and should be tuned according to opponent players' levels. Investigating how to tune such parameters and how they affect the naturalness from human players' views is important for future work.

To sum up, this section proposes a new position control approach specific for endgames using KataGo's features. The new approach successfully lost more games than the original method, and its naturalness was also likely improved. Although the conducted experiment was limited, we claim that the new approach is promising.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we adapt existing methods for position control and producing various strategies to Go programs based on AlphaGo Zero. There are several differences between such programs and traditional MCTS programs. We summarize the differences and propose methods to solve some related problems.

First, we introduce three methods for natural position control in general, *e.g.*, avoiding moves far from the opponent's last moves. From experiments, it is confirmed that a very strong program, Leela, can play gently against a weaker traditional MCTS program, Ray. Experiments employing human subjects show that the proposed method can reduce the number of unnatural moves compared to a naive method that ignores naturalness. Besides, we propose another approach specifically to endgames, which considers territory advantages instead of win rates of moves.

Also, we introduce two methods to produce various strategies. The online method modifies the selection probabilities from policy networks. The offline method trains biased networks by modifying the definitions of wins and losses in self-play games. The biased networks are used together with normal ones to maintain the programs' strength. In our experiments, center- and edge/corner-oriented strategies are produced. Experiments on human subjects show that the strategies can be successfully identified.

Some promising future research includes (1) applying the offline method for strategy production on 19×19 boards, (2) implementing other strategies such as optimistic, pessimistic, offensive, and defensive, and (3) solving remaining naturalness problems.

[2]Some other parameters were slightly different from those in Section 7.2.

ACKNOWLEDGMENT

This research is financially supported by Japan Society for the Promotion of Science (JSPS) under contract numbers 17K00506 and 20K12121.

REFERENCES

1. D. Silver, A. Huang, *et al*., "Mastering the game of Go with deep neural networks and tree search," *Nature*, 2016, Vol. 529, pp. 484-489.
2. D. Silver, J. Schrittwieser, *et al*., "Mastering the game of Go without human knowledge," *Nature*, 2017, Vol. 550, pp. 354-359.
3. H. Iida, K. I Handa, and J. Uiterwijk, "Tutoring strategies in game-tree search," *ICGA Journal*, 1995, Vol. 18, pp. 191-204.
4. K. Ikeda, S. Viennot, and N. Sato, "Detection and labeling of bad moves for coaching Go," in *Proceedings of IEEE Conference on Computational Intelligence and Games*, 2016, pp. 1-8.
5. R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search,," in *Proceedings of the 5th International Conference on Computers and Games*, 2006, pp. 72-83.
6. K. Ikeda and S. Viennot, "Production of various strategies and position control for Monte-Carlo Go-Entertaining human players,," in *Proceedings of IEEE Conference on Computational Intelligence in Games*, 2013, pp. 145-152.
7. Y. Tian, Q. Gong, W. Shang, Y. Wu, and C. L. Zitnick, "ELF: An extensive, lightweight and flexible research platform for real-time strategy games," in *Proceedings of the 31st Conference on Neural Information Processing Systems*, 2017, pp. 2656-2666.
8. I. C. Wu, T. R. Wu, A. J. Liu, H. Guei, and T. Wei, "On strength adjustment for MCTS-based programs,," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Vol. 33, 2019, pp. 1222-1229.
9. N. Sephton, P. I. Cowling, and N. H. Slaven, "An experimental study of action selection mechanisms to create an entertaining opponent," in *Proceedings of IEEE Conference on Computational Intelligence and Games*, 2015, pp. 122-129.
10. T. Fan, Y. Shi, W. Li, and K. Ikeda, "Position control and production of various strategies for deep learning Go programs," in *Proceedings of International Conference on Technologies and Applications of Artificial Intelligence*, 2019, pp. 1-6.
11. Leela-Zero, https://github.com/leela-zero/leela-zero, 2020.
12. Leela's network trained from human games, https://sjeng.org/zero/best_v1.txt.zip, 2020.
13. Ray – Computer Go Program, http://computer-go-ray.com, 2020.
14. KataGo, https://github.com/lightvector/KataGo/tree/v1.3.3, 2020.
15. Ladder (Go), https://en.wikipedia.org/wiki/Ladder_(Go), 2020.

Yuan Shi received his B.E. in Jiangxi Normal University, Nanchang, Jiangxi, China, in 2015. He is currently a master's student at Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan. His research interests include artificial intelligence and computer games.

Tianwen Fan received his B.S. from Civil Aviation University of China, Tianjin, China, in 2015, and his M.S. from Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan, in 2020. His research interests include artificial intelligence and computer games.

Wanxiang Li received his B.E. in Wuhan Institute of Technology, Wuhan, Hubei, China, in 2018. He is currently a master's student at Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan. His research interests include artificial intelligence and game informatics.

Chu-Hsuan Hsueh received her B.S. and Ph.D. in Computer Science from National Chiao Tung University, Hsinchu, Taiwan, in 2014 and 2019, respectively. She is currently an Assistant Professor at Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan. Her research interests include artificial intelligence, machine learning, and computer games.

Kokolo Ikeda received from the University of Tokyo the B.S. degree in 1999, from the Tokyo Institute of Technology the M.E. degree in 2000, and D.E. degree in 2003. He is currently an Associate Professor at Japan Advanced Institute of Science and Technology. His research interests include optimization and game informatics.