Position Control and Production of Various Strategies for Deep Learning Go Programs

Tianwen Fan*, Yuan Shi*, Wanxiang Li*, and Kokolo Ikeda*
*School of Information Science, Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan
Email: {fantianwen,shiyuan,wanxiang.li,kokolo}@jaist.ac.jp

***Abstract*—Computer Go programs have exceeded top-level human players by using deep learning and reinforcement learning techniques. On the other hand, "Entertainment Go AI" or "Coaching Go AI" are also interesting directions which have not been well investigated. Several researches have been done for entertaining beginners or intermediate players. Position control or producing various strategies are important tasks, and some methods have been proposed and evaluated using a traditional Monte-Carlo tree search program. In this paper, we try to adapt the method to LeelaZero, a program based on AlphaGo Zero. There are some critical differences between the previous program and the new program. For example the new program does not use random simulations to the ends of games, then the previous method for producing various strategies cannot be used. In this paper we summarized the differences and some expected problems, and proposed several approaches to solve the problems. It was shown that the modified LeelaZero could play gently against weaker players (48% won against a program Ray). Through experiments using human subjects, it was shown that the average number of unnatural moves per game was 1.22, where that by a simple method without considering naturalness was 2.29. Also we evaluated the proposed method for training "center-oriented" and "edge/corner-oriented" players, and it was confirmed that human players could identify the produced strategy (center or edge/corner) with a probability of 71.88%.**

***Index Terms*—Computer Go, Position Control, Various Strategies, Entertainment, Coaching, Deep Learning, AlphaGo Zero**

I. Introduction

The game of Go 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], introduced by DeepMind, beat a top-level professional player for the first time. Moreover, the advanced version AlphaGo Zero [2] is much stronger than the original one, by using self-training.

Now the strength of Go programs have surpassed top-level human players, so it becomes even worthier conducting researches on entertaining and teaching human players [3], [4]. There are many requirements to entertain and/or coach human players, and one of the most important things is to control the game position, in other words, to keep a good balance between Black and White. If the computer player is strong, then intermediate players can be easily beaten. So, entertaining computer players should select gentle or weak moves intentionally. At the same time, such moves should not be too unnatural. Using handicap stones is a good way to balance the game of Go. However, on one hand, handicap stones may harm the enjoyment. On the other hand, even nine stones are not enough for intermediate players to win against the state-of-the-art programs.

A successful method was proposed for natural position control and production of various strategies [5]. However, the paper used a traditional Monte Carlo tree search (MCTS) program named Nomitan. We find some problems when applying this method to the state-of-the-art programs because their strength and mechanisms are all different from traditional MCTS programs.

In this paper, at first, we introduce existing strength control methods in Section II, and in Section III we summarize the differences between traditional and new programs, with some expected problems caused by the differences. Then, in Section IV we propose several ways for adapting the existing method to programs based on AlphaGo Zero, such as LeelaZero [6] (abbr. Leela for the rest of the paper) and ELF OpenGo [7] (abbr. Elf for the rest of the paper). Next we evaluate the performance (gentleness and naturalness), and discuss the remaining or newly raised problems in Section V. Finally, in Section VI, we propose a completely new method to produce various strategies, because the previous method cannot be used in programs based on AlphaGo Zero. The players with center-oriented or edge/corner-oriented strategy were evaluated by using human subjects.

II. Related Work

A. 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, but Monte-Carlo Tree Search (MCTS) made significant progress [8]. The level of MCTS Go programs reached high amateur levels, such as KGS-6d in 2015.

In 2016, "AlphaGo" [1], which combined multi-level convolutional neural networks and MCTS, beat a top-level player Lee Sedol and became the first program to defeat a top-level professional Go player. The successor of AlphaGo is "AlphaGo Zero" [2]. There are many differences, but both programs use the "selection probabilities" of moves (policy) and the "expected winning probability" (value) by inputting board statuses into neural networks, trained by supervised learning or reinforcement learning.

978-1-7281-4666-9/19/$31.00 ©2019 IEEE

B. Position control

Generally, there are two major kinds of approaches to control the strength of Go programs. One is trying to produce a "static" weakness, for example, by decreasing the number of simulations or thinking time. Another is a "dynamic" way, to determine the proper degree of strength control according to current statuses on the boards. For example, if the computer player has a big advantage against a weaker human player, then some bad moves are intentionally played. We call this "position control".

The two kinds of approaches have their own advantages and disadvantages. The static approach has consistent strength, this is usually a good aspect. However, the estimation of opponent levels should be done before the games. Further, even when the strength of the two players is similar, sometimes one-sided games happened. On the contrary, the dynamic approach can control the game position (advantage) without knowing opponent levels. Good moves or bad moves are selected according to the degree of advantage of the computer player, so sometimes the inconsistent strength may be problematic.

In the both cases of static/dynamic approaches, bad moves were sometimes selected. However, too bad moves should be avoided because such moves harm the enjoyment. Softmax selection is such an approach for MCTS programs [9]. Each candidate move m_i is selected with a probability of $n_i^z / \sum_j n_j^z$, where n_i is the number of visits of move m_i, and z is a parameter. When $z = 0$, all candidates are selected randomly, while the most visited move is selected when $z \to \infty$. Wu *et.al.* [10] applied this approach to the game of Go on Elf, and showed the strength can be well controlled by z. Also they proposed a new technique to remove less visited moves from candidates, to avoid selecting too bad moves.

In our previous paper [5], naturalness of moves was explicitly considered to entertain intermediate players, with using a traditional MCTS program. The degree of naturalness of a move was calculated by "selection probability", which was trained from human game records. When the expected winning probability (of computer player) was too high, a bad but not too unnatural move is intentionally selected.

C. Playing various strategies

Human Go players have widely varied strategies or preferences, such as favoring edge/corner territory, favoring center territory, pessimistic, optimistic, offensive, or defensive. Such variety is a seed of entertainment, so it is valuable to implement such strategies in computer Go programs. In the previous paper [5], a simple trick was employed and partly succeeded to let human subjects identify each strategy, while with only a slight loss of strength. Simply speaking, the method changed the definition of wins/losses in MCTS simulations. For example, for players favoring center territory, a center territory was counted as 1.2 points, and a corner/edge territory was counted as 0.8 points.

III. Differences in New Programs

The existing methods of position control and producing various strategies [5] were proposed for traditional MCTS programs. Recent programs based on AlphaGo Zero are much stronger than traditional ones, and have different mechanisms. Then, some of the existing methods cannot be directly applied, or may produce bad performance. In this section, we summarize the differences and several expected or known problems. In this paper, we employ two open-sourced programs, Leela [6] as a recent program, and Ray [11] as a traditional program, and 13×13 board is used for experiments.

The main difference lies in the state evaluation mechanism. To evaluate a leaf node, traditional MCTS programs run some (biased) random simulations to the ends of games. Win or loss is judged by the rules of Go. On the contrary, recent programs use value networks to evaluate leaf nodes. As a result, our previous method for producing various strategies (described in Section II-C) cannot be used, because the trick modifies the definition of wins/losses at the ends of games.

Also it is well known that winning probabilities predicted by recent programs are somewhat sharp or drastic. Fig. 1 shows winning probabilities of moves, predicted by Ray and Leela, in 50 testing games. Red triangular points were sampled at the 30th move, green circular at the 60th, and black square at the 100th. The height of red triangle/green circle distribution is greater than the width, which means small dis/advantage for Ray may be judged big by Leela. In traditional MCTS, long random simulations are done especially in early stages of a game, so small dis/advantage may not be well reflected to winning ratios. Our previous method is based on winning probabilities and their differences among candidate moves, so some tuning will be needed.

Another difference is whether human game records are used. Recent programs do not involve human game records, then their trained policy networks are sometimes different from human players' senses. Especially, recent programs tend to play far from the opponent moves more frequently. For example, when Leela and Ray play against each other, the average Euclidean distance between a Leela's move and the last Ray's move (100 games, from the second to the 60th moves) is 3.16 ± 0.10. This is significantly bigger than that between Ray's moves and the last Leela's moves, 2.65 ± 0.08. Usually it seems natural to react directly when the opponent tries to attack or invade. This tendency is strong especially for beginners. So, playing far from the last move is sometimes risky for entertainment, because beginners may think "my move was ignored, it's a strange play". Fig. 2 is an example. Black played C3, which aims to invade the territories of the White corner. For beginners, it is natural to play at D3 (A). However, here Leela selected L11 (1). In fact, this move is not bad, but may seem to be strange from beginners' viewpoints.

IV. Approaches

A. Position control

About the position control, basically the original procedure [5] is employed, but some new methods are newly introduced

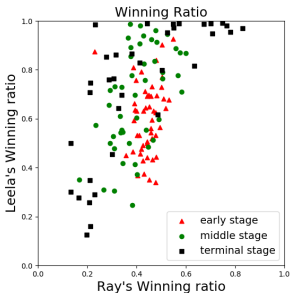

Fig. 1. Predicted winning ratio by Ray (x-axis) and Leela (y-axis)

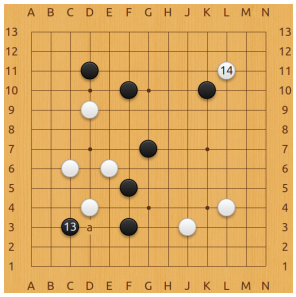

Fig. 2. A move far from the opponent move

to deal with problems introduced in Section III.

The detailed procedure is described in the previous paper [5]. For simplicity of discussion, assume that moves m_i are sorted by their winning ratios, $m_1(w_1, p_1)$, $m_2(w_2, p_2)$, ..., where w_i represents the winning ratio, and p_i shows the selection probability (we assume this value can approximate naturalness to a certain degree, even without human game records). Briefly speaking, (1) if w_1 is much higher than w_2, the best move m_1 is played. (2) When w_1 is low, i.e., the program is losing, the best move m_1 is played. (3) When w_1 is moderate (around 50%), the most natural move is selected among moves whose winning ratios are not low. (4) When w_1 is high, i.e., the program is winning, then a worse move such as m_2 or m_3 is intentionally selected, with considering the balance of selection probability and winning ratio.

There are three expected/observed problems when using Leela, (a) some candidate moves have few visits, then their winning probabilities are not reliable, (b) Leela tends to move far from the last opponent moves, and (c) naturalness is often sacrificed for decreasing winning probabilities.

About problem (a), it is reasonable to ignore bad moves after few visits for strong play. However, since we intentionally select bad moves for position control, they should also be well searched. Thus, we employ a high exploration coefficient, 10. Also, we exclude moves with visits less than 100 from candidates. We call this modification method-A. Problems (b) and (c) are discussed in the following subsections.

B. Method-B: decreasing distance from last opponent moves

In many traditional MCTS programs, the distances to the last opponent moves are explicitly considered to calculate the selection probabilities [8]. Namely, if a candidate move is near the last opponent move, its selection probability is increased.

So we propose to modify the policy network output according to the distances from the last opponent moves. Let m_i be a candidate move, p_i be the original selection probability, and d_i be the Euclidean distance from the last opponent move to m_i. New selection probability p'_i is calculated as follows:

- if $d_i \leq 2$, $p'_i = p_i \times 1.50$
- if $2 < d_i \leq 3$, $p'_i = p_i \times 1.25$
- if $3 < d_i \leq 4$, $p'_i = p_i \times 1.00$
- if $4 < d_i \leq 5$, $p'_i = p_i \times 0.75$
- if $5 < d_i \leq 6$, $p'_i = p_i \times 0.50$
- if $6 < d_i \leq 7$, $p'_i = p_i \times 0.25$
- otherwise, $p'_i = p_i \times 0.10$

One exception is the case that the program has no stones around the last opponent move (Euclidean distance ≤ 3), which means that the opponent move does not intend to attack or invade. In such case, selection probabilities are not changed.

C. Method-C: avoiding big sacrifice

In our original method, when the best winning probability w_1 is higher than a threshold 0.55, a bad move is intentionally selected. At first, each candidate m_i is tested whether it satisfies one of four conditions: (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. When c is high, worse moves can be played, and then naturalness is usually sacrificed. Let M^+ be a set including m_1 and moves satisfying one of the conditions, which can be considered as candidates "not so bad, and relatively natural". From M^+, the move which has the lowest w_i is selected to decrease the advantage.

Fig. 2 is also a good example. After a search by Leela, the winning probability of D3 was 0.697 and the selection probability was 0.400. Meanwhile, the winning probability of L11 was 0.690 and the selection probability was 0.139. Our original method selected L11 to decrease the winning ratio, but naturalness was sacrificed. This selection seems to be unreasonable, since the winning ratio does not decrease too much but naturalness is almost sacrificed.

So, we propose to consider not only w_i but also p_i when selecting the move from candidates M^+. More specifically, the move which has the lowest $w'_i = w_i - \alpha \times p_i$ is selected, where α is a positive parameter. For the above example, w' of D3 is 0.657 when $\alpha = 0.1$, and that of L11 is 0.676, thus D3 is selected.

D. Playing various strategies

Production of various strategies was also proposed in the previous paper [5]. It changed the definition of wins/losses for random simulations of MCTS, which needs to play games to the ends. However, since random simulations to the ends of games are not done in programs such as Leela or Elf, the same method cannot be applied. So, we propose to train new networks to produce various strategies in an offline manner, while the previous method is done in an online manner.

Since Leela open-sourced the codes, we can customize the training of new Leela networks, e.g., with our own definition of wins/losses, or in different board sizes. Usually, when training value networks, the pairs (game state, result) are saved as training data. The result is determined by the rules of Go, by comparing the territories of Black and White (+Komi).

In this paper, we tried to make a center-oriented network, by giving higher weights to center territories:

- From the first line to the third line (i.e., corner or edge), the weight is $1 - \beta$.
- On line 4, the weight is 1.

- Starting from line 5 (i.e., center), the weight is $1 + \beta$.

When β is positive, center-oriented network are trained. With negative β, which gives lower weights to center territories, edge/corner-oriented networks are trained. Note that if such networks are used solely, they may not play well for standard games of Go.

Simply by modifying the definition of wins/losses, networks with different preference can be trained, such as optimistic, pessimistic, offensive or defensive. For example, if one captured stone is counted as -2 points, defensive networks may be trained. We remain such experiments as future work.

V. Experiments on Position Control Methods

In this section, we compare the original and the modified position control methods when implemented in a recent program, Leela. Our experiments were done in a board size of 13×13, Leela's policy network and value network were trained from zero for two weeks, using a server with two Titan-X GPUs. On the other hand, Ray, a traditional MCTS program 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 White side.

A. Evaluation of position control ability

At first, we evaluated the strength of the trained Leela. Pure Leela was much stronger than Ray. Ray could win no game against Pure Leela in 30 games. Next, we evaluated Leela with original position control method. All parameters were set to the recommended values as the original paper, especially, the important parameter c was set to 1.5. The only difference was to apply method-A (shown in Section IV-A). We call this setting $Leela_{A15}$. Ray won 183 out of 500 games against $Leela_{A15}$, which means position control was performed, but the degree was insufficient.

To make Leela weaker, c should be increased, but it is expected that naturalness is sacrificed. So, we tried $c = 2.5$, and included methods-B and -C, shown in Sections IV-B and IV-C, to compensate the sacrificed naturalness. $\alpha = 0.25$ was used for method-C. We call this setting $Leela_{ABC25}$. Ray won 238 out of 500 games against $Leela_{ABC25}$, so $Leela_{ABC25}$ was significantly weaker than $Leela_{A15}$. When $c = 3.5$, Ray won with a probability of 73%.

B. Evaluation of naturalness

Generally, it is easy for stronger human/computer players to lose intentionally, if regardless of the naturalness of moves. Valueless moves or even suicidal moves can be played to lose the advantage. But weaker players do not want such unnatural position control. So, it is very important to know how many moves seem to be unnatural from the viewpoints of human players. The previous paper [5] reported that 5.2 White moves seemed to be unnatural per game when using naive control method, and only 1.9 White moves when using the proposed method. The Black player was a weaker player and its unnatural moves were not counted.

TABLE I
Position control performance by four versions of Leela

Leela version	Winning ratio of Ray	number of unnatural moves
Pure Leela	0.00	-
$Leela_{naive}$	-	2.29 ± 0.53
$Leela_{A15}$	0.37 ± 0.04	1.27 ± 0.41
$Leela_{ABC25}$	0.48 ± 0.04	1.22 ± 0.25

So, we conducted a similar experiment, though the human subjects and their strength were different. We prepared three versions of Leela, $Leela_{A15}$, $Leela_{ABC25}$, and $Leela_{naive}$. For $Leela_{naive}$, candidate moves were collected using method-A, and the move with a winning probability nearest to 0.5 was selected, without explicitly dealing with the naturalness of moves.

In the experiments, we asked nine human subject, ranks from 8k to 8d, to review games played by the three versions. A total of 15 games were generated by the three versions, which played against Ray five games for each. 15 games were given in random order and in a blind manner. Each person was given two hours to review the records from the first moves to the 60th moves.

The average numbers of unnatural moves per game is listed in Table I. One may note that the numbers of unnatural moves by gentle Leela programs ($Leela_{A15}$ and $Leela_{ABC25}$) were smaller than the that of our previous paper. However, the experiment settings (programs and human subjects) were totally different, thus, it is hard to compare. The difference between $Leela_{naive}$ and our two methods was statistically significant. The results showed that position control methods were also effective for programs based on AlphaGo Zero. Furthermore, $Leela_{ABC25}$ had almost the same naturalness as $Leela_{A15}$, while the strength was significantly weakened. The results suggested that method-B and method-C are effective.

In addition, the average Euclidean distance between $Leela_{ABC25}$'s moves and the last Ray's moves (100 games, from the second to the 60th moves) was 2.33 ± 0.06. The value was significantly lower than those of Ray (2.65 ± 0.08) and $Leela_{A15}$ (3.16 ± 0.10). This tendency is as we expected, and the degree can be controlled by tuning parameters of method-B, according to players' preference. Fig. 3 shows a game between Ray (Black) and $Leela_{ABC25}$ (White). White played gently without obviously bad moves. Finally, Black won 0.5 points.

C. Remaining problems

It was shown that the number of unnatural moves from the viewpoints of human players was low, still there are some remaining problems.

1) Ladder: Ladder is a well-known sequence of moves in which an attacker pursues a group in a zigzag pattern across the board. If there are no intervening stones, the group will hit the edge of the board and be captured [12]. Fig. 4 is a game between Elf (White) and a human player (KGS-3d, Black). Black played G14 (97), and threatened three white stones by ladder. However, Elf ignored this risk and played E18 (98),

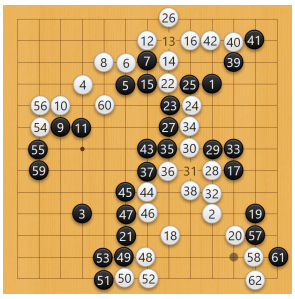

Fig. 3. A game between Ray (Black) and $Leela_{ABC25}$ (White)

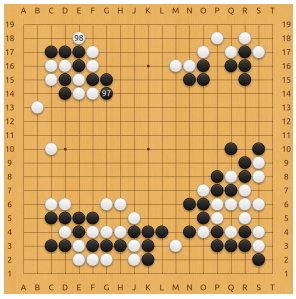

Fig. 4. Example of "Ladder" (White 98)

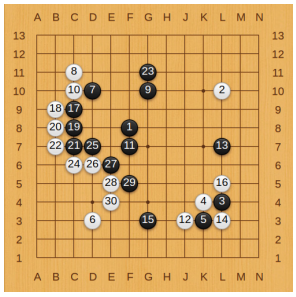

Fig. 5. Net_{center} (Black) vs standard Leela

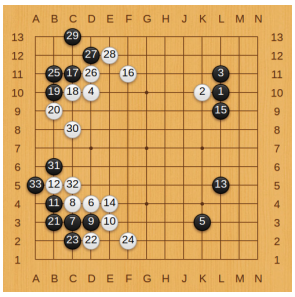

Fig. 6. Net_{edge} (Black) vs standard Leela

then the white stones were captured and Elf resigned soon. Such "unnatural" moves cannot be avoided by our position control method, since it is an issue of the original program.

2) Early stage: As mentioned in Section III, very high (such as over 80%) or low winning probability can be easily observed even in early stages when using recent programs. For example, Leela often thinks the advantage is big and tries to lose by strange moves. Usually, strange moves in early stages are easily detected by human players, which should better to be avoided. For this problem, it may be valuable to disable strong position control in early stages.

3) Terminal stage: The problem in terminal stages may be a bit more complex, which Leela often wins by 0.5 or 1.5 points. When the difference of territories 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, because $+0.5$ points can mean almost a winning ratio of 100%, and -0.5 points almost a winning ratio of 0%. When there is a move with 0.5 points, Leela is almost impossible to select a move with -0.5 points, because difference of winning ratio is too big. For this problem, it may be valuable to consider expected points of territories instead of winning ratios. It was easy when using traditional MCTS programs, but additional learning may be needed when using recent programs.

VI. Experiments of Playing Various Strategies

In Section IV-D, a new approach for producing various strategies was proposed. For each strategy or preference, some period of self-training is needed. We trained center-oriented Leela network (Net_{center}) and edge/corner-oriented Leela network (Net_{edge}) about two weeks from scratch. The parameter β was set to $+0.2$ for Net_{center}, and -0.2 for Net_{edge} respectively.

A. Pure strategies

Net_{center} and Net_{edge} were trained not for playing solely. Both won no game against the standard Leela in 60 games for checking their behaviors. Fig. 5 shows a game between Net_{center} (Black) and the standard Leela (White). Moves 1, 9, and 11 are very strange from the viewpoint of the standard rules of Go, but does fit the goal of Net_{center}. Fig. 6 shows a game between Net_{edge} (Black) and the standard Leela (White). Moves 13 and 15 are far from hot area, and finally the left-bottom black stones are in danger. The results confirmed that Net_{center} and Net_{edge} had clear preference to the center and the edge/corner territories respectively.

B. Mixed approach

Both center- and edge/corner-oriented network had clear preference, but too weak when used solely. In this section, we introduce a way to combine the trained network, Net_{center} or Net_{edge}, with the original Leela, in order to balance the strength and strategy preference. The decision making procedure is as follows:

- *Search*. The current board status is given to both networks (the original Leela and the biased one), and two lists of candidate moves, M_o and M_b, are obtained. Let $w_o(m)$ be the winning ratio of move m from the original Leela, and $p_b(m)$ be selection probability of the biased Leela .
- *Mix*. From the move list M_b, some moves are removed: (1) when its winning ratio $w_o(m)$ is less than $\max_i\{w_o(i)\} - param_{gap}$, or (2) when its selection probability $p_b(m)$ is less than $param_{sp}$. After that, if M_b is not empty, the most visited move in M_b is selected. Otherwise, the most visited move in M_o is selected.

This mix procedure means that when and only when there are several acceptable moves from the viewpoint of the original Leela, the best one from the viewpoint of the biased (center- or edge/corner-oriented) Leela is selected. $param_{gap}$ was set to 0.05, and $param_{sp}$ to 0.1. We call the mixed players $Leela_{center}$ and $Leela_{edge}$ respectively. Fig. 7 shows a game between $Leela_{center}$ (Black) and the original Leela (White). Fig. 8 shows a game between $Leela_{edge}$ (Black) and the original Leela (White). While there are still several unexpected moves, we can recognize the preference.

We conducted experiments to evaluate the strength of the mixed versions. Each of $Leela_{center}$ and $Leela_{edge}$ played 300 games against the original Leela. The number of wins were 135 and 129 respectively. We can say that the mixed versions were not weak, because they only selected acceptable moves from the viewpoint of the original Leela.

C. How well strategies can be identified

We mentioned that mixed Leela seems to have a strategy or preference, and not so weak. Finally, we conducted an

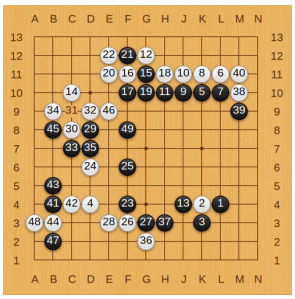

Fig. 7. $Leela_{center}$ (Black) vs standard Leela

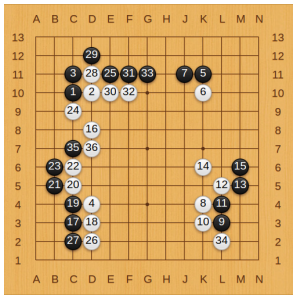

Fig. 8. $Leela_{edge}$ (Black) vs standard Leela

TABLE II
EVALUATION RESULT OF CENTER, EDGE/CORNER, AND STANDARD LEELA

Actual Evaluated	a	b	c	d	e	Total	Average
$Leela_{center}$	36	19	20	3	2	80	-1.05
$Leela_{edge}$	3	4	13	14	46	80	1.20
Original	37	22	48	25	28	160	-0.09

experiment using human subjects, to evaluate how the strategy can be identified from the viewpoints of human players.

We tested three versions of Leela, (A) the original Leela, (B) $Leela_{center}$, and (C) $Leela_{edge}$. Four groups of game records were generated, A vs A, A vs B, A vs C, and B vs C. In each group, five game were played. Totally, 20 game records were given to eight human subjects (ranks 6k to 8d) in random order and in a blind manner. Each person was given one hour to review the records.

Human subjects were asked to judge which preference can be identified for Black player and White player respectively in each game. Options were (a) center-oriented, (b) slightly center-oriented, (c) nothing, (d) slightly edge/corner-oriented, and (e) edge/corner-oriented. For averaging the results, we assigned -2 to (a), -1 to (b), 0 to (c), $+1$ to (d), and $+2$ to (e). The number of answers and average values are listed in Table II.

We can claim that both center- and edge/corner-oriented strategies could be produced and recognized with a high probability. In 115/160 answers, the strategies of programs were correctly judged, and there were only 12 opposite answers. Average scores were also close to expected, and indeed different from each other. It is interesting that when the original Leela played against center- (or edge/corner-) oriented Leela, the original Leela tended to be judged as edge/corner- (or center-) oriented.

In this paper, the training and experiments were done on 13×13 board. The early stage is shorter than 19×19 board case, and thus human players had less chances to identify strategies. So, we expected it to be easier to produce various strategies in 19×19 board case, while the required training cost will be much higher.

VII. Conclusions and Future Work

In this paper, we adapted 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 programs. We summarized the differences and proposed methods to solve the raised problems.

Next we employed three methods for natural position control, for example avoiding moves far from the last opponent moves. From experiments, it was confirmed that we can control the winning ratio of a very strong program, Leela, against a traditional MCTS program, Ray. Through experiments using human subjects, it was shown that the proposed method can reduce the numbers of unnatural moves, compared to a naive method regardless to naturalness.

Also we proposed a way to produce various strategies. By self-training with using a modified definition of wins/losses, it is possible to obtain center-oriented or edge/corner-oriented network. Experiments on human subjects successfully showed that the mixture of such biased networks and the original networks was effective to produce specific strategies while keeping the strength.

Some promising future researches include (1) training on 19×19 board and comparing to other existing work, (2) implementing optimistic/pessimistic/offensive/defensive strategies, and (3) solving remaining problems about naturalness.

Acknowledgment

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

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[3] H. Iida, K.-i. Handa, and J. Uiterwijk, "Tutoring strategies in game-tree search," *ICGA Journal*, vol. 18, no. 4, pp. 191–204, 1995.

[4] K. Ikeda, S. Viennot, and N. Sato, "Detection and labeling of bad moves for coaching go," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016, pp. 1–8.

[5] K. Ikeda and S. Viennot, "Production of various strategies and position control for monte-carlo goentertaining human players," in *2013 IEEE Conference on Computational Inteligence in Games (CIG)*. IEEE, 2013, pp. 1–8.

[6] "LeelaZero," https://github.com/leela-zero/leela-zero, last accessed: 2019-02-10.

[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 *Advances in Neural Information Processing Systems*, 2017, pp. 2656–2666.

[8] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *International conference on computers and games*. Springer, 2006, pp. 72–83.

[9] N. Sephton, P. I. Cowling, and N. H. Slaven, "An experimental study of action selection mechanisms to create an entertaining opponent," in *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2015, pp. 122–129.

[10] 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 AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1222–1229.

[11] "Ray," http://computer-go-ray.com, last accessed: 2019-02-10.

[12] "Ladder of go," https://en.wikipedia.org/wiki/Ladder_(Go), last accessed: 2018-07-27.