JAIST Repository
https://dspace.jaist.ac.jp/

Title	Playing Good-Quality Games with Weak Players by Combining Programs with Different Roles
Author(s)	Hsueh, Chu-Hsuan; Ikeda, Kokolo
Citation	2022 IEEE Conference on Games (CoG 2022), 2022
Issue Date	2022-09-20
Type	Conference Paper
Text version	author
URL	http://hdl.handle.net/10119/18240
Rights	This is the author's version of the work. Copyright (C) 2022 IEEE. 2022 IEEE Conference on Games (CoG). DOI: 10.1109/CoG51982.2022.9893698. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Description	2022 IEEE Conference on Games (CoG 2022), 23rd August 2022

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Playing Good-Quality Games with Weak Players by Combining Programs with Different Roles

Chu-Hsuan Hsueh and Kokolo Ikeda
School of Information Science
Japan Advanced Institute of Science and Technology
Nomi, Ishikawa, Japan
{hsuehch, kokolo}@jaist.ac.jp

***Abstract*—Computer programs have become stronger than top-rated human players in several games. However, weak players may not enjoy playing against these strong programs. In this study, we propose combining two programs with different roles to create programs suitable for weak players. We use a superhuman program that generates candidate moves and evaluates how good the moves are, as well as a program that evaluates the moves' naturalness. We implement an instance for Go, which employs a superhuman program, KataGo, and a neural network trained using human games. Experiments show that the proposed method is promising for playing good-quality games with weak players.**

***Index Terms*—teaching, strength adjustment, entertaining, Go**

I. Introduction

Computer programs have attained superhuman levels of play in a wide range of games. A well-known example is AlphaZero [1] for chess, shogi, and Go. These strong programs provide humans with challenges and learning opportunities.

However, human players may not enjoy playing against or learning from these strong programs for two reasons: (1) Human players can hardly win, and losing constantly is frustrating. (2) These programs' moves are sometimes not human-like or mysterious, making it difficult for human players to learn. In contrast, many human teachers play teaching games with students to help improve the their skills. For simplicity of discussion, the following takes a board game, Go, as an example and presents critical factors in teaching games.

Go teachers usually (a) try playing balanced games with students. Although these teachers have higher skills, they (a1) do not try to always win against their students (a2) nor try to win or lose completely by a large margin in each game.

Also, (b) Go teachers usually try to reflect the goodness of the students' moves in the final outcomes. Namely, if the students play many good moves, the teachers tend to let the students win, rewarding the good moves, and *vice versa*.

To play balanced games, Go teachers give handicap stones, intentionally play some bad moves to lose advantages, or do both. Nonetheless, (c) Go teachers try to keep the quality of the move as good as possible. Thus, students can learn good stone shapes or good move flows from the teachers.

For computer programs, (d) naturalness, or more specifically human-likeness, is an important factor. A typical example is that human players tend to finish a battle in some local area and then move on to another, but strong Go programs do not.

We aim at creating programs that consider factors (a)–(d) to play good-quality games with weak players[1]. Considering all four factors at once is a challenging task. For example, to achieve (a), a naïve method is to select moves whose win rates are closest to 50%. However, the method is likely to play well or badly based on the goodness of the opponent's moves, failing to achieve (b). Also, many moves may not have good shapes and not be human-like, failing to achieve (c) and (d).

As weak players' opponents, we propose combining two programs with different roles and implement an instance for Go. First, we employ KataGo [2] to provide candidate moves along with accurate evaluations on advantages. Second, we employ a neural network trained using human games to evaluate the moves' naturalness. Experiments compare several methods and show that our method can generally play good-quality games against weak programs while looking natural.

II. Related Work

Significant effort has been put into developing suitable programs for human players to play with. Some researchers tried weakening strong programs. For programs based on Monte-Carlo tree search (MCTS), Sephton et al. [3] investigated several ways of selecting the moves to play. One of which selected moves based on a softmax policy: move i was selected with a probability of $(n_i)^z / \Sigma_j (n_j)^z$, where n_i is move i's visit count and z a parameter controlling the program's strength. Liu et al. [4] extended the method by adding a threshold, R_{th}, to avoid extremely bad moves. Let n_{max} be the highest visit count among the moves from MCTS. Only moves with $n_i \geq n_{max} \times R_{th}$ became candidates for the softmax policy.

Some researchers employed human games to tune parameters for weakening strong programs. Nakamichi and Ito [5] replaced evaluation functions with those trained using amateur players' games to create weaker but human-like programs. Rosemarin and Rosenfeld [6] modified MCTS in AlphaZero programs slightly and used human games with different rating ranges to learn three parameters to achieve different strengths.

Some researchers used human games to train neural networks to imitate human players' moves. McIlroy-Young et al. [7] trained different neural networks for different rating ranges.

[1]We target middle-level amateur players, specifically, kgs8k to kgs2d in Go. The KGS Go Server, https://www.gokgs.com, is an online platform for playing Go, which also ranks players (including humans and computer programs).

They showed that the players' moves in each rating range could be best predicted by the neural network trained for that range. They also showed that AlphaZero-based programs played quite differently from humans, including the intermediate weaker versions. Jacob et al. [8] further incorporated neural networks trained using human games into MCTS. They showed that both the accuracy of predicting human moves and the playing strengths could be improved.

Some researchers employed game-specific knowledge to create natural-looking programs. For example, Shi et al. [9]'s work on Go considered the distance to the previous move.

III. Approach

Ideas. We propose playing teaching games with weak players by combining two programs with different roles. Specifically, we need a superhuman program that generates candidate moves and can accurately evaluate the quality of each move. Additionally, we need a program that informs us how natural each candidate move looks. From the two programs' results, we attempt to select moves that play close games with weak players while being not too bad and looking natural. In contrast, most of the previous methods attempt to decide moves without considering such a division of roles.

Go Implementation. For Go, we employ an AlphaZero-based program, KataGo [2], as the superhuman player's role and a neural network (NN) trained using strong human players' games[2] as the naturalness role. The NN outputs the policy (probabilities of moves) and the estimated win rate given a position, and we use only the policy part, denoted by π_{human}.

To generate a move, (i) we let KataGo search using N_{sim} simulations to obtain candidate moves as well as the moves' statistics. We focus on each move i's visit count, n_i (i.e., the number of times this move was selected from the root node during search), and territory advantage a_i (i.e., the number of points that the current player is leading). We refer to a_i's instead of win rates because the former provides richer information to determine how advantageous a player is.

Next, (ii) we retain moves in the candidate list with visit counts n_i higher than a threshold (say 10) so that the statistics are somewhat reliable. We further obtain the maximum territory advantage of the remaining moves (i.e., $a_{max} = \max_{i \in \{j | n_j > 10\}} a_i$) and delete moves that lose too many points (i.e., $a_{max} - a_i$ higher than some other threshold, say 20). Overall, step (ii) aims to filter out obviously bad moves. Let M denote the set of remaining moves after (ii).

With the move set M, (iii) we decide the *ideal advantage loss* l^* for this position to play close games. The ideal advantage loss is defined as follows:

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

where α is a positive number deciding the range that we tune l^* finer, and β decides the degree of the loss. Namely, when we are behind with a huge margin ($a_{max} \leq -\alpha$), we do not eagerly try losing more points. When it is a close game, the ideal advantage loss l^* is about β points. Even when we lead a lot ($\alpha < a_{max}$), l^* is bounded by $2 \cdot \beta$ to avoid bad moves.

Finally, (iv) we calculate *scores* for each move in M and (v) select the move with the highest score. The score s_i of move i considers both naturalness and the ideal advantage loss l^*:

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

where p'_i is move i's probability from π_{human}, ϵ a small number to prevent the first term from being 0, e_i the error of loss between move i and the ideal loss (i.e., $|(a_{max} - a_i) - l^*|$), and γ a coefficient deciding the importance of e_i. Based on the scores calculated using Eq. (2), natural moves with advantage loss close to the ideal loss are more likely to be selected.

IV. Experiments

We compared several methods and let them play against weak programs rather than human players so that we could collect numerous games for analysis. Each pair played 500 games, altering between black and white. Japanese rules with a komi of 6.5 were used. For weak programs, we selected programs and settings with well-known strength: GNU Go[3] level 10 (~kgs8k) and Pachi[4] with settings of kgs3k and kgs2d.

We created five types of programs for comparison, and we created three types of programs based on KataGo [2]. The first (P1) was the proposed method in Section III. The second (P2) was the softmax method by Liu et al. [4]. The third (P3) was a naïve method that selected moves with territory advantages closest to zero.

For P1 and P2, we further introduced a trick, optimistic komi, because we observed that the two methods hardly lost to the weak programs. With an optimistic komi of k, the program searches with a komi of $6.5 - k$ when it plays black and $6.5 + k$ when it plays white. Consider a position where one move wins 0.5 points and another loses 0.5 points. With a normal komi of 6.5, the latter move is rarely visited; thus it is out of consideration because of the thresholds on visit counts. However, such losing moves may be good for playing teaching games. The proposed and the softmax methods considered more moves because of the dynamic komi.

In addition to the three types of programs based on KataGo, we let the weak programs (P4) play against themselves (e.g., GNU Go vs. GNU Go). Furthermore, we let π_{human} (P5) play against the two Pachi programs. P5 selected moves with the highest probability, i.e., $\text{argmax}_i\, p'_i$, which we considered as an instance of McIlroy-Young et al. [7]'s method.

In addition, we obtained and analyzed human games from the Fox Go dataset[5]. For the ranks corresponding to kgs8k, kgs3k, and kgs2d (5k, 1d, and 5d in Fox Go), we identified players that played most frequently at the corresponding ranks (who were more likely belonged to that rank) and extracted approximately 30 games between those players for each rank.

[2]https://sjeng.org/zero/best_v1.txt.zip, which was released in the Leela Zero project, https://github.com/leela-zero/leela-zero.

[3]https://www.gnu.org/software/gnugo/

[4]https://github.com/pasky/pachi

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

After games were played/collected, they were reviewed using another KataGo program to evaluate the played moves and determine the final winner. Subsections IV-A to IV-D show the results in terms of factors (a)–(d).

A. Results of Factor (a)–Balanced Games

We set N_{sim} to 1,000 for KataGo-based programs (P1–P3). The k for optimistic komi was 4 for the proposed (P1) and the softmax (P2) methods. Additionally, we tuned parameters by preliminary experiments so that P1 and P2 had win rates of approximately 50% against the weak programs, which is a way to achieve (a1). For P1, we set α to 25.0, γ to 0.4, and ϵ to 0.001 and used β to control the programs' strength: 6.0 for GNU Go, 3.0 for Pachi$_{\text{kgs3k}}$, and 2.0 for Pachi$_{\text{kgs2d}}$. For P2, we set R_{th} to 0.1 and used z to control the strength: 0.65 for GNU Go, -0.5 for Pachi$_{\text{kgs3k}}$, and 1.0 for Pachi$_{\text{kgs2d}}$.

Table I shows the win rates of P1–P5 against the weak programs, with 95% confidence intervals. P1 and P2 achieved different strengths by adjusting one parameter (β for P1 and z for P2). P3's (the naïve method) win rate was low, but it intrinsically attempted to play close games for each move[6]. P4 (the programs themselves) had, of course, balanced wins and losses. For P5 (π_{human}), the strength was between kgs3k and kgs2d. To obtain NNs with the desired strength, training on different sets of human games is required, which needs more effort than tuning programs' parameters.

TABLE I
WIN RATES AGAINST WEAK PROGRAMS

	GNU Go	Pachi$_{\text{kgs3k}}$	Pachi$_{\text{kgs2d}}$
P1 (proposed)	53.6±4.4%	48.4±4.4%	50.4±4.4%
P2 (softmax)	49.4±4.4%	47.4±4.4%	47.6±4.4%
P3 (naïve)	7.0±2.2%	7.4±2.3%	6.4±2.1%
P4 (weak program)	52.0±4.4%	47.8±4.4%	50.6±4.4%
P5 (π_{human})	–	65.8±4.2%	19.2±3.5%

To measure (a2), we calculated the ratios of games where the final territory differences were within -10 to 10. Table II shows the ratios with 95% confidence intervals. Because of the space constraint, the rest of this paper only shows the results of kgs3k. The other two ranks, kgs8k and kgs2d, generally had similar tendencies. As expected, most games against P3 ended with small territory differences. Next were P1 and P2, where P1 was slightly better than P2. For P4 and P5, as well as humans at the same ranks (Fox Go), many games ended with big territory differences.

B. Results of Factor (b)–Reflecting Goodness/Badness

To measure (b), we first define the *badness* of the weak player's moves. Based on the evaluator KataGo, we obtained the maximum territory advantage a_{max} for each position as was obtained in Section III. We then calculated the loss of the played move i by $a_{max} - a_i$. We defined the badness of a player for a game using the average loss of its moves.

[6]The reason for the low win rate is as follows: For moves with territory advantages closest to zero, it was random whether the advantages were positive or negative. In endgames, once a move with a negative advantage was selected, the program was hard to win the game.

For each set of 500 played games, we did simple supervised learning to predict the final win/loss using the badness of the target weak player (e.g., Pachi$_{\text{kgs3k}}$). Given game g's badness b_g, the weak player was predicted to win if b_g was lower than some threshold $b_{th} \in \{0, 0.01, ..., 5.99\}$, and otherwise predicted to lose. Among the threshold values, we selected the one that led to the highest Matthews correlation coefficient (MCC)[7], a measure of the quality of binary classifications. MCC ranges from -1 to 1, where 1 indicates perfect predictions. A higher MCC indicated that the target weak player's goodness/badness was better reflected in the final wins/losses.

Table II shows the results of the maximum MCC of games against Pachi$_{\text{kgs3k}}$. P1 and P2 rarely played very bad moves, and the weak player was likely to lose if it made some mistakes. Thus, the weak player's goodness/badness was properly reflected in the final wins/losses. In contrast, some moves by weak programs (P4) and programs without searches (P5) were bad. Particularly, every 2–3 out of 100 moves were very bad, as will be shown in Subsection IV-C. The weak player's goodness/badness was hardly reflected since P4 and P5 themselves made mistakes, resulting in relatively low MCCs. As for P3, who tried playing close games eagerly, when the weak player played very bad moves, P3 also tried doing so. Thus, the MCC was close to 0 (random prediction).

C. Results of Factor (c)–Move Quality

We employed both quantitative and qualitative evaluations to measure (c). For the quantitative evaluation, we calculated the frequency of very bad moves played by P1–P5 (and the extracted Fox Go games). A move i was said to be very bad if the loss $a_{max} - a_i$ was higher than 15.0. Table II shows the frequency of playing very bad moves. As expected, P1 and P2 played very bad moves the least frequently. P3 selected very bad moves when the opponent did so. Thus, it played more very bad moves than P1/P2, where the frequency was similar to P4. P5 selected moves without searches and played very bad moves the most frequently among the compared programs. As for human players, it seems that very bad moves were played more, though the number of extracted games was few and the game settings differed (e.g., thinking time limit).

For the qualitative evaluation, we asked five Go experts ($\geq$kgs3d, including a professional) to evaluate games generated/collected in our experiments, where P3 was excluded since it played many bad and obviously unnatural moves. We randomly selected games where the programs played white, as well as human games from the Fox Go dataset, and asked the experts to review the white player's moves. The games were provided randomly and blindly. Namely, the experts did not know which player played which game.

On a scale of 1 to 5, we asked the experts to score the quality of the white player's moves (how good the shapes/flows were, 5 being very good). Table II shows the results. P1 received a bit higher score than human players and was clearly better than P2 and P4. P5 received the highest score despite the

[7]https://en.wikipedia.org/wiki/Phi_coefficient

TABLE II
RESULTS OF RANK KGS3K

	(a2) Close games	(b) Rewarding	(c) Move quality		(d) Naturalness		
	(%) Games with final territory difference in [−10, 10]	MCC	(%) Moves with big loss	Expert evaluation	Distance to previous move	Average of p'	Expert evaluation
P1 (proposed)	59.4±4.3%	0.645	0.28%	3.5	3.711	0.468	3.4
P2 (softmax)	53.4±4.4%	0.720	0.22%	2.4	5.860	0.253	1.8
P3 (naïve)	97.0±1.5%	0.022	1.37%	–	5.949	0.281	–
P4 ($\text{Pachi}_{\text{kgs3k}}$)	20.4±3.5%	0.240	1.87%	3.0	3.627	0.381	2.9
P5 (π_{human})	18.4±3.4%	0.165	3.03%	4.1	3.139	0.571	4.3
Fox Go	7.1±9.5%	–	7.35%	3.3	3.426	0.364	3.6

fact that 3.03% of the moves were very bad from KataGo's perspective. We considered such a mismatch to be reasonable, which is explained as follows. Since P5 was trained using strong players' games, its playstyle might be the most strong-human-like and thus was easier for the experts to understand.

D. Results of Factor (d)–Naturalness

We also employed both quantitative and qualitative evaluations to measure (d). For the quantitative evaluation, we calculated the Euclidean distance between a move and the previous. As discussed in Section I, human players tend to finish local battles before moving on to the next, and we considered that the distance could reflect this tendency. Table II shows the average distances. There were two clearly different groups: P2/P3 and P1/P4/P5/human. Among KataGo-based programs, P1 was clearly more natural than P2 and P3.

Next, we employed π_{human} to evaluate naturalness. For each move i, we obtained its probability ${p'}_i$ from π_{human} and then calculated the average. We considered that the moves might look more natural with a higher average p'. Table II shows the results. P5, of course, had the highest value since it played the moves with the highest ${p'}_i$. P1 was the next highest, where π_{human} was involved in the move selection. P4 had a moderate value, which was close to humans. P2 and P3 had the lowest values, again showing that they were unnatural.

For the qualitative evaluation, we asked the experts to evaluate the naturalness in addition to the moves' quality. Table II shows the average scores. P5's moves appeared the most natural, for a similar reason explained in Subsection IV-C. P1 received a close score to human players, whereas P2 was clearly the worst. We concluded that employing π_{human} did help significantly improve the naturalness.

In summary, the proposed method was the most promising among the compared ones, explained as follows. In terms of naturalness, P2 and P3 were unpromising. P3–P5 played more bad moves than P1/P2 and could not reflect their opponent's goodness/badness in final wins/losses. P5 required different sets of human games to train NNs for each rank, while P1 achieved different strengths by tuning one parameter, β.

V. CONCLUSIONS AND FUTURE WORK

To play good-quality games with weak players, we proposed combining two programs with different roles: a superhuman program for generating candidate moves, which also evaluates the moves' goodness, and a program for evaluating the moves' naturalness. We implemented an instance for Go by employing KataGo as the superhuman program and a neural network trained using strong human players' games as the naturalness evaluator. Our experiments compared several methods that served as the opponents of weak players (programs). The results indicated that the proposed method could generally play balanced games, reflect weak players' goodness/badness in final outcomes, and select good and natural moves.

For future research directions, we intend to further improve the proposed method in terms of naturalness and quality of moves and deploy it on Go servers to play with human players.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Numbers JP18H03347 and JP20K12121.

REFERENCES

[1] D. Silver *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 352, no. 6419, pp. 1140–1144, 2018.

[2] D. J. Wu, "Accelerating self-play learning in Go," arXiv, 2020, abs/1902.10565. [Online]. Available: https://arxiv.org/abs/1902.10565

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

[4] A.-J. Liu *et al.*, "Strength adjustment and assessment for MCTS-based programs [research frontier]," *IEEE Comput. Intell. Mag.*, vol. 15, no. 3, pp. 60–73, 2020.

[5] T. Nakamichi and T. Ito, "Adjusting the evaluation function for weakening the competency level of a computer shogi program," *ICGA J.*, vol. 40, no. 1, pp. 15–31, 2018.

[6] H. Rosemarin and A. Rosenfeld, "Playing chess at a human desired level and style," in *the 7th Int. Conf. on Human-Agent Interact.*, 2019, pp. 76–80.

[7] R. McIlroy-Young *et al.*, "Aligning superhuman AI with human behavior," in *the 26th ACM SIGKDD Int. Conf. on Knowl. Discovery & Data Mining*, 2020, pp. 1677–1687.

[8] A. P. Jacob *et al.*, "Modeling strong and human-like gameplay with KL-regularized search," in *ICLR 2022 Workshop on Gamification and Multiagent Solutions*, 2022.

[9] Y. Shi *et al.*, "Position control and production of various strategies for game of Go using deep learning methods," *J. of Inf. Sci. and Eng.*, vol. 37, no. 3, pp. 553–573, 2021.